

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, назва факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПІДТРИМКИ ПРОКАТУ АВТОМОБІЛІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SUPPORTING CAR RENTAL

Виконав: студент(ка) 2 курсу, групи КНТ-121м
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

Хомяков А.А.

(прізвище та ініціали)

Керівник Сердюк С.М.

(прізвище та ініціали)

Рецензент Поляков М.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Хомякова Анастасія Андріївна

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів
підтримки прокату автомобілів
Research and Software Implementation of Methods for Supporting Car Rental

керівник проєкту (роботи) Сердюк Сергій Микитович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “4” листопада 2022 року № 363

2. Строк подання студентом проєкту (роботи) грудень 2022 року
3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Основні рішення щодо реалізації компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програми.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	Сердюк С.М., доцент		
Нормоконтролер	Дейнега Л.Ю., ст. викладач		

7. Дата видачі завдання “1” вересня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз та дослідження предметної області	2-5 тижні	Розділи 1, 2
3	Розробка архітектури програми	6 тиждень	Розділ 3
4	Розробка програми	7-9 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	13 тиждень	

Студент(ка)

(підпис) Хомяков А.А.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Сердюк С.М.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
99 с., 25 рис., 3 табл., 3 дод., 18 джерел.

ПРОКАТ АВТОМОБІЛЯ, СПІЛЬНЕ ВИКОРИСТАННЯ,
РЕЗЕРВУВАННЯ, ПЛАТІЖ, PAYPAL, ПРОГНОЗУВАННЯ ПРОКАТУ.

Об'єктом виконаного дослідження є процес прокату автомобілів, а предметом – методи підтримки прокату автомобілів.

Мета дослідження полягала в розробці програмного забезпечення підтримки прокату автомобілів для допомоги власникам автомобілів під час виставлення їх на прокат.

Матеріали, методи та технічні засоби: мова Python, система для роботи з базами даних PostgreSQL, фреймворк Django, застосунок Django PayPal, пакет lightgbm.

Результати. У результаті роботи отримано програмний застосунок, який може застосовуватися на практиці для підтримки взаємодії власників автомобілів і їх потенційних індивідуальних клієнтів щодо прокату автомобілів. Для цього забезпечено прогнозування кількості прокатів автомобіля на місяць, щоб власник міг визначити виставлені параметри прокату, забезпечено підтримку PayPal-платежів, щоб дозволити сплачувати за резервування.

Висновки. Науковою новизною роботи є метод прогнозування попиту на прокат автомобілів, який ґрунтується на застосуванні дерев рішень з навчанням на основі алгоритму LightGBM та параметрах автомобіля і прокату, дозволяє визначити кількість завершених резервувань прокатів автомобілів протягом наступного місяця.

Галузь використання – прокат автомобілів від власників, переважно на короткий строк.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 99 pages, 25 figures, 3 tables, 3 appendixes, 18 sources.

CAR RENTAL, SHARING, RESERVATION, PAYMENT, PAYPAL, RENTAL FORECAST.

The object of the performed research is the car rental process, and the subject is car rental support methods.

The purpose of the study was to develop car rental support software to help car owners when renting out their cars.

Materials, methods and technical means: Python language, PostgreSQL database system, Django framework, Django PayPal application, lightgbm package.

The results. As a result of the work, a software application was obtained to be used in practice to support the interaction of car owners and their potential individual customers regarding car rental. The forecasting of the number of car rentals per month is provided, so that the owner can determine the rental parameters efficiently. PayPal payment support is provided to allow bank payment for car rental reservations.

Conclusions. The scientific novelty of the work is the method of predicting the demand of car rentals, which is based on the application of decision trees with learning based on the LightGBM algorithm, rental and car parameters. The method allows determining the number of completed car rental reservations during the next month.

The field of usage is car rental from owners, mainly for a short period.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	10
1.1 Проблеми і особливості сучасного стану галузі прокату автомобілів ..	10
1.2 Аналіз програмного забезпечення підтримки прокату автомобілів	11
1.3 Висновки за розділом 1 та постановка завдань роботи	15
2 Матеріали і методи.....	17
2.1 Задача прогнозування попиту на прокат автомобілів	17
2.2 Модель LightGBM.....	18
2.3 Метод прогнозування попиту на прокат автомобілів	20
2.4 Висновки за розділом 2	22
3 Проєктування програмного забезпечення	23
3.1 Вибір базових технологій розробки програмного забезпечення	23
3.2 Визначення функціональних вимог до програми	24
3.3 Визначення структури бази даних	27
3.4 Висновки за розділом 3	38
4 Основні рішення щодо реалізації компонентів системи.....	40
4.1 Структура розробленої програмної системи.....	40
4.2 Реалізовані класи програмної системи	41
4.3 Висновки за розділом 4	51
5 Експлуатація, тестування та експериментальне дослідження програми	52
5.1 Призначення програми	52
5.2 Умови виконання програми	52
5.3 Виконання програми.....	52
5.4 Експериментальне дослідження прогнозування попиту на прокат автомобілів	56
5.5 Висновки за розділом 5	58
Висновки	60

Перелік джерел посилання	61
Додаток А Технічне завдання	64
Додаток Б Текст програми	69
Додаток В Слайди презентації.....	91

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

- IPN – Instant Payment Notification;
LightGBM – light gradient-boosting machine;
БД – база даних.

ВСТУП

Прокат автомобілів є частиною загального тренду в світовій економіці на повторне, спільне використання товарів. Одним з рушіїв цього тренду є подорожі, які можуть здійснюватися як в форматі туристичних, так і в форматі робочих поїздок. У таких випадках людина потребує знайти житло для проживання, що призвело до розвитку засобів оренди житла безпосередньо від господарів. Подібною сферою є прокат автомобілів, які можуть надаватися не централізовано компанією, а їх власниками. Перспективною є підтримка взаємодії господарів з клієнтами через створення такого програмного забезпечення та забезпечення необхідних засобів для господарів щодо прогнозування майбутньої активності стосовно їх автомобілів.

В Україні поточна ситуація характеризується тим, що через енергетичні проблеми кількість громадського транспорту значно зменшилась у всіх містах. Електротранспорт не може працювати в повній мірі через обмеження використання електроенергії. Це призводить до того, що люди мають шукати нові способи дістатися на роботу або в інші звичні місця, які розташовані на певній відстані. У підсумку люди, які не мають достатньо коштів для купівлі власного автомобіля, шукають можливості прокату автомобілів. Для зменшення вартості багато потенційних клієнтів прагнуть отримати автомобіль на прокат у власників. З іншого боку в світі ситуація також не є стабільною, адже останнім часом вартість нафти, а в підсумку і бензину значно зростала. Це з одного боку призводило до того, що люди відмовлялися від використання автомобілів на певний період, але з іншого боку це давало можливість тимчасово віддати свій автомобіль у користування іншим людям. Саме проблема підтримки прокату автомобілів через створення відповідного програмного забезпечення розглядається в роботі.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Проблеми і особливості сучасного стану галузі прокату автомобілів

Прокат автомобілів є однією з галузей економіки спільного використання, яка зараз є достатньо популярною у світі і продовжує своє розширення далі, адже вона дозволяє водіям, коли вони не користуються своїми автомобілями, здавати їх в оренду іншим водіям [1].

У 2021 році ринок оренди автомобілів становив 119,28 мільярда доларів США, і очікується, що до 2027 року він досягне 223,07 мільярда доларів США, однак пандемія COVID-19 перешкодила зростанню даного ринку через обмеження на подорожі та закриття кількох туристичних агентств і постачальників послуг з прокату автомобілів: обмеження було знято в усьому світі, і ринок оренди автомобілів почав відновлюватися до кінця 2021 року завдяки схильності споживачів до послуг прокату в основних країнах північноамериканського та європейського регіону [2]-[4].

Кількість мандрівників по роботі значно збільшилася за останні десять років, оскільки багато організацій виходять на міжнародний рівень: попит на розкішні автомобілі представницького класу на ринку оренди зріс, оскільки збільшилася кількість мандрівників по роботі, попит на автомобілі представницького класу зростає в усьому світі, оскільки вони забезпечують комфорт і легкість подорожей за прийнятною ціною, враховуючи величезні бізнес-можливості та потенціал зростання цього сектора, деякі підприємства зосереджуються виключно на корпоративному сегменті, очікується, що ринкова потреба в компаніях з прокату автомобілів зростатиме й далі через попит на послуги спільного використання автомобілів [3].

Для підтримки цієї галузі зазвичай використовується відповідне програмне забезпечення. Таке програмне забезпечення передбачає зазвичай реєстрацію власників, після чого визначається власний автомобіль, додаються фотографії та деталі, встановлюється ціна, після чого власник має відповідати на замовлення інших користувачів, які хочуть взяти його автомобіль у прокат,

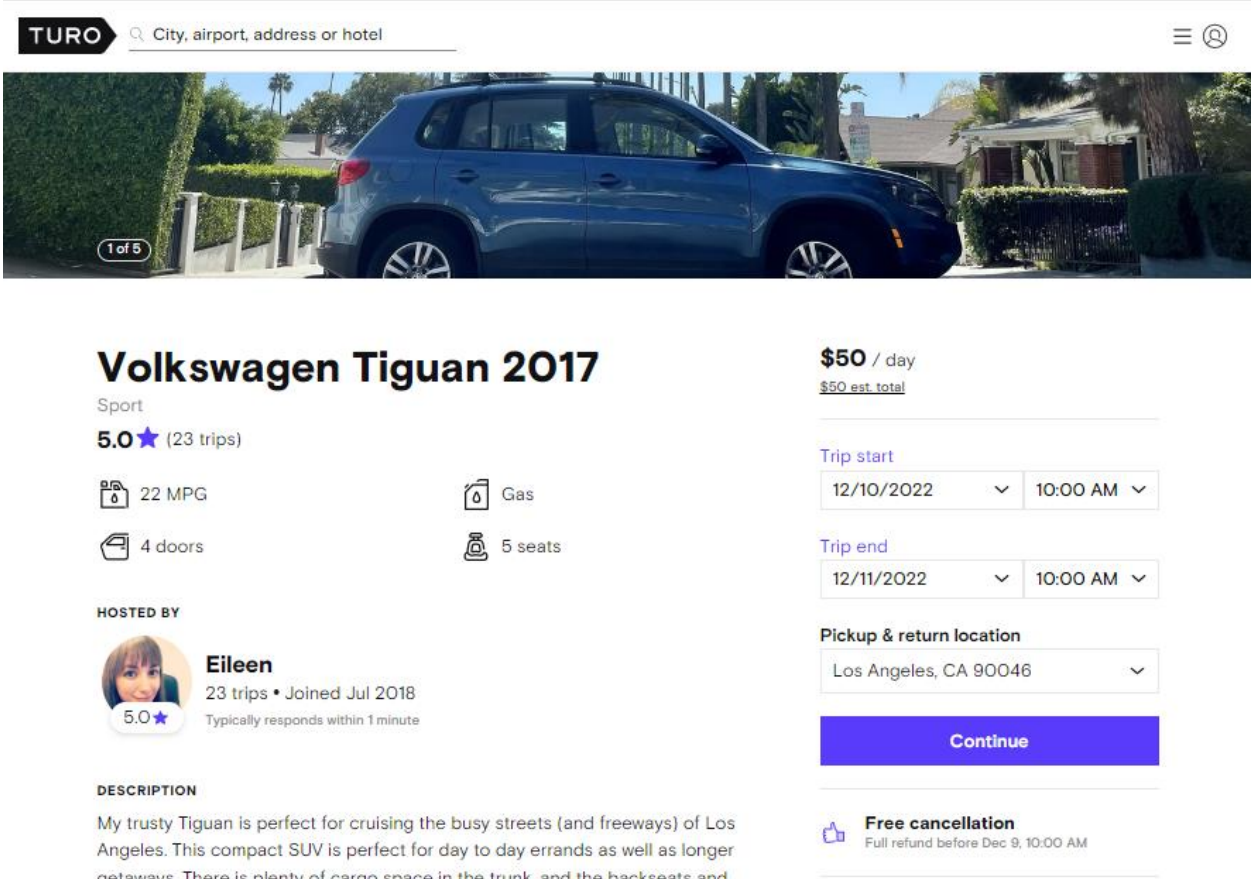
а далі зустрічають орендарів, передають їм ключі, забирають машину в кінці періоду прокату [1].

Подібні сервіси зазвичай потребують сплати меншої суми за прокат: клієнти можуть орендувати стандартну машину всього за 10 доларів на день або автомобіль класу люкс менше ніж за 100 доларів на день, у той час як традиційні компанії беруть сотні чи тисячі доларів на день за подібний прокат, але на клієнта покладається необхідність заправляти бак, інакше з нього буде стягуватися плата за низький рівень палива [1], [5].

1.2 Аналіз програмного забезпечення підтримки прокату автомобілів

Як було визначено в підрозділі 1.1, прокат автомобілів є дуже активною сучасною галуззю, яка зараз забезпечується окремими компаніями і власниками автомобілів. У даному підрозділі основна увага приділена саме програмним рішенням, у яких звичайні люди виставляють власні автомобілі на прокат клієнтам, які також є звичайними людьми, користувачами програми.

Turo (рис. 1.1) – це один з найвідоміших вебсайтів і відповідних застосунків для прокату автомобілів, де автомобілі орендуються від власників, при цьому зазвичай клієнт забирає автомобіль у місці отримання, яким може бути будинок власника або громадське місце, наприклад, парк чи торговий центр, деякі власники пропонують можливість доставити автомобіль клієнту за додаткову плату, надається також можливість для деяких автомобілів скористатися функцією Turo Go, за допомогою якої можна розблокувати автомобіль за допомогою програми, дозволяючи почати подорож, не зустрічаючись із господарем [6], [7]. Це може бути перевагою даного застосунку, але з іншого боку це не сприяє безпеці власників, адже за особистого контакту між власником і клієнтом більша ймовірність відкинути тих, хто може заподіяти шкоду.



TURO City, airport, address or hotel

Volkswagen Tiguan 2017
Sport
5.0★ (23 trips)

22 MPG Gas
4 doors 5 seats

HOSTED BY
 **Eileen**
23 trips • Joined Jul 2018
5.0★ Typically responds within 1 minute

DESCRIPTION
My trusty Tiguan is perfect for cruising the busy streets (and freeways) of Los Angeles. This compact SUV is perfect for day to day errands as well as longer getaways. There is plenty of cargo space in the trunk, and the backseats and

\$50 / day
\$50 est. total

Trip start
12/10/2022 10:00 AM

Trip end
12/11/2022 10:00 AM

Pickup & return location
Los Angeles, CA 90046

Continue

Free cancellation
Full refund before Dec 9, 10:00 AM

Рисунок 1.1 – Відображення автомобіля для прокату на вебсайті Turo

В основному для прокату потрібно мати вік не менше ніж 21, хоча деякі автомобілі можна забронювати у віці 18 років за додаткову плату за молодого водія, за правилами скасування можна безкоштовно скасувати бронювання за 24 години до поїздки, а серед недоліків є те, що оскільки прокат здійснюється від власника, якість може відрізнятись, проте зазвичай, взявши напрокат автомобіль, клієнт, як правило, отримує відносно новий, щойно вимитий автомобіль [6].

Turo надає для кожного автомобіля величезну кількість параметрів, що включають обігрів сидінь, доступність різних мультимедійних систем, наявність сплаченого доступу до платних доріг, безключового доступу тощо. Відповідна кількість фільтрів може бути встановлена і під час пошуку, тобто всі наявні в оголошеннях параметри можуть використовуватися в процесі. Також під час пошуку встановлюється період прокату, включаючи день і час, визначається варіант сортування результатів. Наявні різні варіанти

скасування, різні варіанти доставки автомобіля клієнту. Є можливість переглянути відгуки клієнтів, які також накопичуються для самого власника і таким чином визначають оцінку для автомобіля і для власника окремо.

Вебсайт MyCarYourRental (рис. 1.2) використовує подібну концепцію, одразу заохочуючи кожного користувача виставляти власний автомобіль на прокат, а всіх інших – шукати автомобіль для прокату поблизу [8].


[How it Works](#) • [Register](#) • [List Your Car](#)

[Back to search result](#)

TESLA MODEL 3



Video Preview
See the car before you rent it!



Owner Fiona L
Parked in Epsom, Auckland
Last online: 16/11 2022

- ✓ Acceptance rate: 79.37%
- ⬆ Response time: 3h, 36 min
- 📊 Views: 7484 | Bookings: 50
- ✓ Mobile & email confirmed
- ✓ Facebook verified

TESLA MODEL 3	5	Automatic
2019	Sedan	150
Electric		Extras

- ✗ Smoking not allowed
- ✗ No pets allowed
- ✓ Minimum age: 25
- ✓ Min licence: Full
- ✓ Unlimited KM, but max 250 km from pick-up
- 🕒 Vehicle owner prefers min 3 hours notice for bookings

Price - includes insurance, roadside assistance and taxes!

Day	Week	Month
225	1300	4895

Pickup

10.12.2022 | 📅 | 10:00 | ⌚

Return

11.12.2022 | 📅 | 10:00 | ⌚

Rental Price

1 Day **\$225**

[Start booking](#)

[Message owner before booking](#)



Rental is insured and includes roadside assistance.

- ✓ Full standard insurance cover
- ✓ Flexible excess options

Map - vehicle location

Рисунок 1.2 – Відображення автомобіля для прокату на вебсайті MyCarYourRental

Використовується дещо особлива система відгуків, яка представляється в формі рекомендацій, які фактично на сторінках використовуються як реклама певним користувачем певного автомобіля. Початково кожен власник автомобіля повинен вказати номер власного автомобіля, адресу, за якою він знаходиться, електронну пошту.

На сторінці автомобіля окрім фотографій, опису та форми для початку бронювання одразу виводиться інформація про вартість бронювання на день, на тиждень та на місяць, надається можливість написати власнику і поставити йому запитання до бронювання. За власником виводиться його дата останньої появи на сайті, відсоток запитів, яку він підтвердив, час відповіді, кількість переглядів сторінок власника і кількість самих резервувань.

Безпосередньо за автомобілем визначаються його модель, кількість сидінь, тип коробки передач, рік виробництва, тип автомобіля, тип пального, можливість палити, перевозити домашніх тварин, мінімальний вік водія, дозволена відстань для всього часу прокату, також надається календар з чітким визначенням дат, коли автомобіль не доступний, доступний або частково доступний. Усі інші особливості автомобіля задаються власником переліком, що унеможливорює потім пошук за цими параметрами. Пошук відбувається фактично за тими параметрами, які було описано вище, їх кількість невелика. Не надається інформація про можливість забрати автомобіль в інших місцях окрім адреси власника, доставити його клієнту.

Після того, як було проведено огляд можливостей різних програмних аналогів, було виконано відповідний аналіз з забезпеченням їх порівняння. Результати подані в таблиці 1.1. Виконувана розробка має основним прототипом Туго, але до того ж підтримує власників автомобілів, прогножуючи кількість прокатів автомобілів, таким чином дозволяючи власникам приймати важливі рішення, зокрема спочатку, коли встановити вартість прокату, визначити інші параметри може бути достатньо складно, а цим характеризуються всі власники, адже вони не мають спеціалізованого досвіду на відміну від компаній, які займаються орендою.

Таблиця 1.1 – Порівняння аналогів з програмою, що розробляється

Критерій	Turo	MyCarYourRental	Розробка
Пошук автомобілів для прокату	+	+	+
Розширений пошук автомобілів	+	—	+
Вибір варіанту доставки автомобіля	+	—	+
Можливість внести власний автомобіль	+	+	+
Прогнозування кількості прокатів	—	—	+
Різні пакети дальності подорожей	+	—	+
Відображення найкращих автомобілів	+	—	+
Підтримка короткотривалого/довготривалого прокату	+	+	+

1.3 Висновки за розділом 1 та постановка завдань роботи

Прокат автомобілів динамічно розвивається і набирає оберти, збільшуючи обсяги у світі. Прокат автомобілів від власників має свої переваги і свої недоліки, але є перспективним напрямком цього розвитку. Розглянуто, проаналізовано і порівняно аналоги, включаючи Turo і MyCarYourRental.

Завдання роботи включають:

- проаналізувати програмне забезпечення підтримки прокату автомобілів;
- виконати дослідження методів підтримки прокату автомобілів;

- виконати проєктування програмного забезпечення підтримки прокату автомобілів;
- реалізувати програмне забезпечення підтримки прокату автомобілів.

2 МАТЕРІАЛИ І МЕТОДИ

З метою визначення того, як компанія працюватиме в майбутні періоди часу, багато компаній наймають аналітиків для створення аналітичних рішень, які передбачають ефективність бізнесу: цей аналіз, як правило, зосереджений на усередненні історичних показників і екстраполяції його на майбутні результати [9].

2.1 Задача прогнозування попиту на прокат автомобілів

У даній роботі розглядається задача прогнозування попиту на прокат автомобілів. Це задача, яка виникає перед власниками, які бажають виставити свій власний автомобіль на прокат, дозволяючи іншим людям використовувати його, сплачуючи за це кошти. У такому випадку власника цікавить звичайно, яку суму він може заробити за своїм автомобілем. Для того щоб охарактеризувати дану ситуацію використовується цільовий показник, який визначається кількістю прокатів даного автомобіля, яку за місяць здійснюють через програму користувачі. Кількість прокатів є показником, який виражає попит на автомобіль. У випадку, якщо прокат не є тривалим в середньому, тобто займає приблизно декілька діб, то за значенням даного показника можна приблизно визначити кошти, на які може розраховувати власник за місяць.

Фактично в якості цільового показника використовується кількість прокатів автомобіля за місяць, що визначається як кількість резервувань прокатів автомобіля в програмі, за якими було надано і повернуто автомобіль. Для випадків, коли прокат розпочато в одному місяці, а потім він триває в іншому, такий прокат враховується в обидва місяці.

У даній задачі передбачається, що прокат може здійснюватися і на короткий період (наприклад, на декілька годин), але переважно прокат здійснюється на більш тривалі періоди (зазвичай на добу).

Варто відзначити, що існують також задачі, в яких виконується подібне прогнозування, але вони розглядаються в розрізі сервісів каршерингу [10]. Ідея каршерингу полягає в тому, що клієнт може в будь-який момент взяти автомобіль у легкодоступному йому місці, використовуючи застосунок, а після повернути його, просто залишивши на парковці. Тобто в такому випадку власник автомобіля не контактує з клієнтом взагалі. У розрізі даної роботи розглядається саме контактний спосіб передачі автомобілів, адже автомобілі надаються своїми власниками, а ті в свою чергу, забезпечуючи безпеку використання автомобілів, повинні впевнитися в тому, кому вони віддають свої автомобілі. У випадку каршерингу автомобіль належить певній компанії і перевірка, а також особиста зустріч не потребується. Це також важлива практична задача, але в даній роботі розглядається саме менш динамічний варіант прокату автомобілів, коли автомобіль віддається у прокат саме своїм власником.

Задача прогнозування попиту на прокат автомобілів у підсумку полягає у визначенні на майбутній місяць кількості прокатів за значеннями характеристик автомобіля і прокатів цього автомобіля за минулі періоди. У процесі розв'язання цієї задачі необхідно встановити набір характеристик, які мають значний вплив на вихідну ознаку. При цьому використання даних про минулі періоди має бути обмеженим, оскільки дана задача має практичне значення зокрема для власників автомобілів, які виставляють свій автомобіль на прокат вперше. Тоді історії автомобіля в програмі немає і відповідно потрібно за всіма іншими ознаками визначити, яка саме кількість прокатів буде у автомобіля в наступний місяць.

2.2 Модель LightGBM

Метод підсилених градієнтом дерев рішень light gradient-boosting machine (LightGBM) є традиційним деревом рішень із посиленням градієнта і базується на трьох важливих принципах:

– слабе навчання: дерева рішень, які представлені в моделі в великій кількості;

– оптимізація градієнта;

– техніка посилення [11].

При цьому дерева будуються послідовно:

– перше дерево дізнається, як відповідати цільовій змінній;

– друге дерево вчиться відповідати залишку (різниці) між передбаченнями першого дерева та очікуваним результатом;

– третє дерево вивчає, як відповідати залишкам другого дерева тощо, відповідно всі ці дерева навчаються шляхом поширення градієнтів помилок по системі [11].

LightGBM використовує два типи технік:

– одностороння вибірка на основі градієнта: хоча в підсилених градієнтом деревах рішень немає власної ваги для екземпляра даних, екземпляри даних з різними градієнтами відіграють різні ролі в обчисленні отримання інформації, зокрема, згідно з визначенням отримання інформації, екземпляри з більшими градієнтами (тобто недостатньо навчені екземпляри) більше сприятимуть отриманню інформації, таким чином, під час зменшення вибірки екземплярів даних, щоб зберегти точність оцінки приросту інформації, слід краще зберігати ці екземпляри з великими градієнтами (наприклад, більшими за попередньо визначене порогове значення або серед верхніх процентилів) і лише випадковим чином відкинути екземпляри з малими градієнтами, в результаті така обробка може призвести до більш точної оцінки посилення, ніж однорідна випадкова вибірка з тією самою цільовою частотою вибірки, особливо коли значення притоку інформації має великий діапазон;

– ексклюзивний пакет функцій: зазвичай у реальних програмах існує велика кількість ознак, хоча простір ознак досить розріджений, що дає можливість розробити підхід майже без втрат, щоб зменшити кількість ефективних функцій, так зокрема у розрідженому просторі функцій багато

функцій є ексклюзивними, тобто вони рідко приймають ненульові значення одночасно, приклади включають одноразові функції (наприклад, представлення одного гарячого слова в аналізі тексту), тому можна об'єднувати такі ексклюзивні функції і в алгоритмі зводиться проблема оптимального групування до проблеми розфарбовування графа (взявши ознаки як вершини та додаючи ребра для кожних двох ознак, якщо вони не є взаємовиключними), а розв'язується вона за допомогою жадібного алгоритму з постійним коефіцієнтом апроксимації [12]-[13].

Даний алгоритм є достатньо вимогливим щодо витрат часу, проте варто відзначити, що для даної задачі немає надто високих вимог часу. Навіть у випадку необхідності кожного разу навчати модель заново, ця операція буде здійснюватися тільки раз на місяць.

2.3 Метод прогнозування попиту на прокат автомобілів

Метод прогнозування попиту на прокат автомобілів розпочинається зі збору даних. Спочатку необхідно зібрати всі дані, які характеризують автомобіль загалом на поточний момент. Фактично це статичні показники для автомобіля. Це модель автомобіля, тип автомобіля, це тип палива, рейтинг автомобіля за весь час його існування в програмі, який має обчислюватися заново кожен раз, коли створюється новий відгук, це місто, де автомобіль знаходиться, і рік його виробництва, який одразу перетворюється на вік автомобіля.

Після цього мають бути зібрані дані, що характеризують вартість прокату автомобіля на наступний місяць. Це значення отримані не за фактичними резервуваннями, а встановлені самим власником у програмі. Тобто виходячи з подової оренди обчислюється середня вартість доби прокату автомобіля.

Далі має бути реалізовано збір даних про всі резервування, які було виконано протягом кожного місяця, і визначено кількість резервувань, які

знаходяться в стані фактично виконаних прокатів, за місяць. На основі цих даних має бути обчислено відносну зміну кількості прокатів протягом останніх 6 місяців та середню кількість прокатів автомобіля за місяць протягом останніх 6 місяців.

Після цього всі ці дані мають бути зібрані у навчальну вибірку. До вхідних ознак навчальної вибірки належать:

- тип палива;
- середня оцінка (рейтинг) автомобіля;
- кількість відгуків;
- середня кількість прокату автомобіля за місяць протягом останніх 6 місяців;
- відносна зміна кількості прокатів протягом останніх 6 місяців – відношення різниці між кількістю прокатів протягом минулого місяця та кількістю прокатів протягом місяця півроку тому, поділена на кількість прокатів протягом місяця півроку тому;
- місто, де відбувався прокат;
- середня вартість доби прокату автомобіля;
- модель автомобіля;
- тип автомобіля;
- рік виробництва автомобіля.

У свою чергу вихідною ознакою є кількість прокатів автомобіля у наступний місяць.

Сукупність отриманих значень вхідних ознак і вихідної ознаки формує навчальну вибірку.

На наступному етапі навчальна вибірка використовується для навчання дерев рішень на основі алгоритму LightGBM. Після того, як навчання завершено, навчена модель зберігається в файл для подальшого використання на наступних етапах, коли необхідність прогнозування виникне. Стандартно це має відбуватися хоча би раз на місяць. У цей же момент може виконуватися повторне навчання моделі за наявності відповідних нових обсягів даних.

Також прогнозування може відбуватися в момент, коли власник змінив значення описаних характеристик, які є вхідними ознаками. Звичайно, що на більшість з них вплинути власник не може, але власник може змінити вартість прокату автомобіля на наступний місяць. Тоді в такий момент має бути виконано прогноз заново, уточнюючи його результат.

Відповідно за описаних моментів запускається наступний етап, на якому переобчислюється середня вартість доби прокату автомобіля на наступний місяць, витягаються статичні характеристики автомобіля. Далі обчислюються результати за попередні 6 місяців для автомобіля. Кожен раз, коли таке обчислення виконується, дані мають зберігатися в таблиці, щоб повторне приведення статистики не виконувалось. Тоді у випадку повторного виникнення цієї задачі дані мають бути витягнуті з таблиці.

Зі всіх ознак формується новий екземпляр даних. Новий екземпляр даних подається на вхід моделі, яка виконує прогнозування кількості прокатів автомобіля у наступний місяць.

2.4 Висновки за розділом 2

Розглянуто особливості задачі прогнозування попиту на прокат автомобілів, відмінності її постановки для прокату від власників порівняно з каршерингом. Проаналізовано алгоритм LightGBM.

Науковою новизною роботи є метод прогнозування попиту на прокат автомобілів, який ґрунтуючись на застосуванні дерев рішень з навчанням на основі алгоритму LightGBM та параметрах автомобіля і прокату, дозволяє визначити кількість завершених резервувань прокатів автомобілів протягом наступного місяця.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір базових технологій розробки програмного забезпечення

Аналіз аналогів вказує на те, що програмні інструменти в цій галузі стандартно представлені вебсайтами, тому в основі подальших рішень використовувалась ідея створення вебзастосунку. Одними з мов програмування, які зараз найактивніше застосовуються для веброзробки, є мови Python і PHP. Проте зараз веброзробка виконується з використанням фреймворків, що дозволяє швидше створювати готові інструменти і більше мати включених інструментів для цього. Тому основний вибір виконувався між фреймворками Symfony мовою PHP та Symfony мовою Python, при цьому враховуючи особливості самих мов і необхідність реалізувати підтримку методу, описаного в розділі 2.

Сукупність технологій і інструментів, які обрано під час проєктування, включає мову Python, фреймворк Django (табл. 3.1) і всі інструменти, які вбудовані в Django, оскільки цей фреймворк має більше готових засобів перевірки безпеки роботи, вищу простоту роботи з моделями даних і кращу підтримку бібліотек для роботи з інтелектуальними методами.

Таблиця 3.1 – Порівняння фреймворків

Показник	Symfony	Django
Мова програмування	PHP	Python
Надання засобів перевірки безпеки роботи	Нижча	Вища
Об'єктно-реляційне відображення	Doctrine	Django ORM
Простота роботи з моделями даних	Нижча	Вища
Підтримка спільноти	Активна	Активна
Підтримка бібліотек для роботи з інтелектуальними методами	Середня	Максимально висока на даний момент

3.2 Визначення функціональних вимог до програми

Неавторизований користувач використовує програму шляхом наступних сценаріїв:

- пошук автомобіля для прокату за вимогами;
- відображення найбільш популярних автомобілів у заданому місті;
- відображення автомобілів з найвищим рейтингом у заданому місті;
- відображення автомобілів власника;
- відображення відгуків про автомобіль;
- відображення прокатного оголошення автомобіля, що включає сценарій обчислення вартості прокату;
- відображення профіля з відгуками про клієнта;
- відображення профіля з відгуками про власника автомобіля;
- авторизація облікового запису або як альтернативний сценарій – реєстрація облікового запису.

Клієнт використовує програму шляхом наступних сценаріїв:

- авторизація облікового запису;
- пошук автомобіля для прокату за вимогами;
- резервування автомобіля для прокату;
- формулювання відгуку про прокат автомобіля, що включає сценарії визначення усередненої рейтингової оцінки власника автомобіля та визначення усередненої рейтингової оцінки автомобіля;
- відображення прокатного оголошення автомобіля, що включає сценарій обчислення вартості прокату;
- збереження автомобіля до переліку улюблених;
- відображення переліку улюблених автомобілів;
- обмін повідомленнями з власником автомобіля;
- відображення найбільш популярних автомобілів у заданому місті;
- відображення автомобілів з найвищим рейтингом у заданому місті;
- відображення резервувань клієнта;

- відображення майбутніх прокатів автомобілів клієнтом;
- відображення минулих прокатів автомобілів клієнтом;
- скасування власного резервування автомобіля;
- відображення автомобілів власника;
- відображення відгуків про автомобіль;
- відображення профіля з відгуками про клієнта, що включає сценарій відображення профіля з відгуками про власника автомобіля.

Власник автомобілів (або використовується скорочений термін власник) використовує програму шляхом наступних сценаріїв (рис. 3.1):

- авторизація облікового запису і як альтернативний сценарій – реєстрація облікового запису;
- виставлення власного автомобіля на прокат, а як альтернативний сценарій – змінення параметрів прокату власного автомобіля;
- встановлення часових інтервалів, коли автомобіль доступний для прокату;
- встановлення варіантів вартості прокату на період за тривалістю;
- встановлення власником локацій для видачі автомобіля в прокат;
- встановлення доступних відстаней на час прокату;
- відображення відгуків клієнтів про власні автомобілі;
- реагування на відгук клієнта;
- формулювання відгуку про клієнта, що включає сценарії визначення усередненої рейтингової оцінки клієнта;
- обмін повідомленнями з клієнтами;
- відображення найбільш популярних автомобілів у заданому місті;
- відображення автомобілів з найвищим рейтингом у заданому місті;
- відображення запланованих прокатів автомобілів, а як альтернативний сценарій – скасування резервування за власним автомобілем;
- відображення минулих прокатів автомобілів.

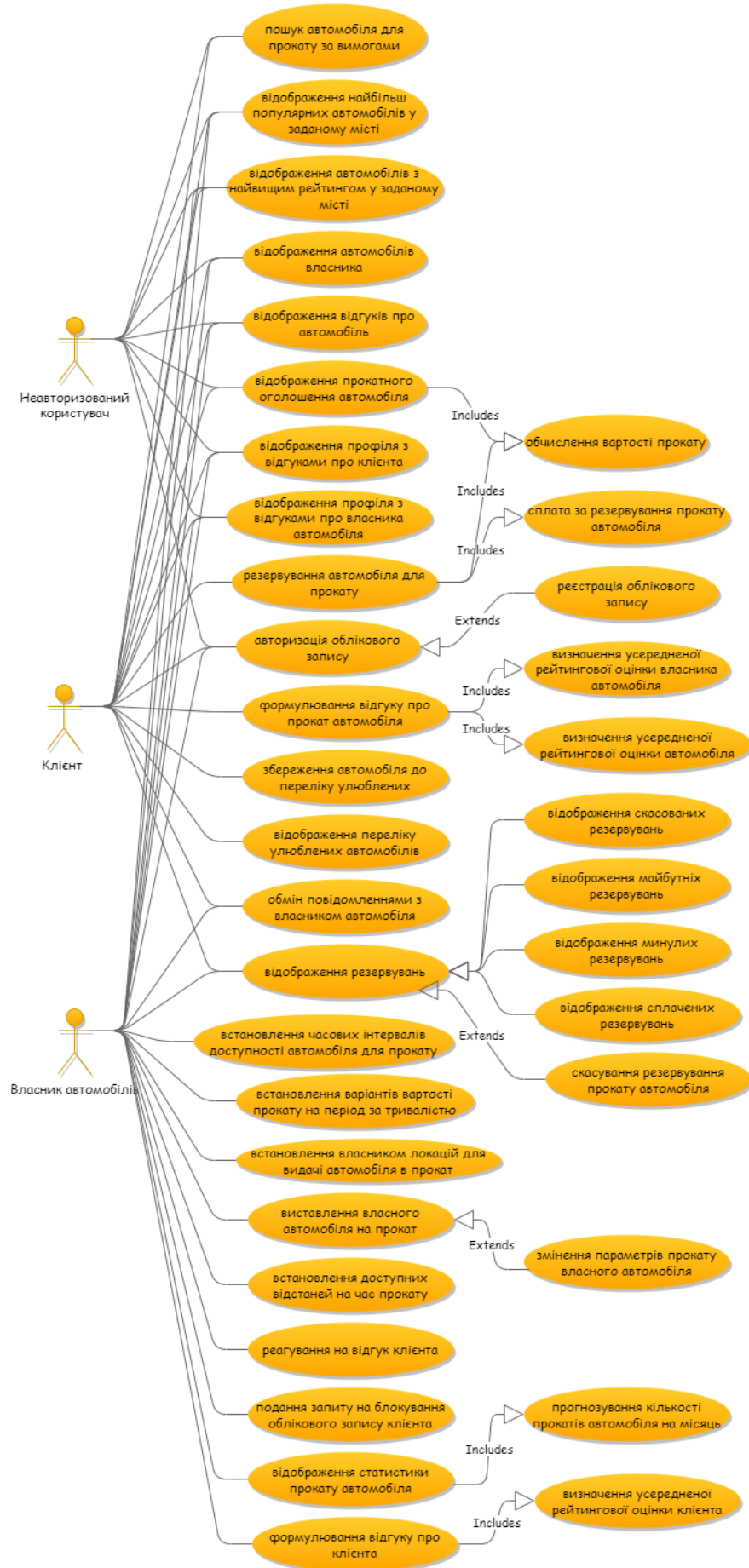


Рисунок 3.1 – Діаграма варіантів використання програмної системи

Також сценаріями власника є:

- відображення автомобілів власника;
- відображення профіля з відгуками про клієнта;
- відображення профіля з відгуками про власника;
- подання запиту на блокування облікового запису клієнта;
- відображення статистики прокату автомобіля за минулі періоди;
- прогнозування кількості прокатів автомобіля на місяць.

3.3 Визначення структури бази даних

Визначаючи структуру бази даних (БД), спочатку необхідно було виконати вибір системи управління. Цей вибір виконувався, враховуючи, що розробка виконується для програмного забезпечення, яке має використовуватися на практиці потенційно великою кількістю користувачів, та враховуючи використання фреймворку Django. Фреймворк Django має обмеження щодо можливостей роботи з такими системами. У результаті вирішено використовувати PostgreSQL, зважаючи на його широкі можливості, потужність та наявність актуального інтерфейсу для Django.

Автомобілі, які власниками виставляються на прокат через програму, зберігаються в БД у таблиці `carentsoft_vehicle`, яка має поля:

- `id` – ідентифікатор прокатного автомобілю;
- `proprietor` – власник автомобілю, що є зовнішнім ключем з таблиці `carentsoft_proprietor`, з якою встановлено зв'язок типу один-до-багатьох;
- `vehiclemodel` – модель прокатного автомобілю, що є зовнішнім ключем з таблиці `carentsoft_vehiclemodel`, з якою встановлено зв'язок типу один-до-багатьох;
- `vinnumber` – ідентифікаційний номер транспортного засобу, який ідентифікує даний автомобіль (не використовується як ключ, адже автомобіль може належати різним власникам за весь час і ця таблиця визначає записи, які

стосуються конкретного автомобіля в тому варіанті, в якому він надається в прокат власником, у конкретного власника, а не автомобіля загалом);

- `vehetype` – тип транспортного засобу;
- `rating` – середнє значення рейтингу, що за результатами усереднення даних з відгуків клієнтів отримав автомобіль;
- `city` – місто, в якому знаходиться автомобіль;
- `reviewsnumber` – кількість відгуків від клієнтів про цей автомобіль;
- `producedyear` – рік виробництва автомобіля;
- `state` – стан автомобіля щодо прокату на даний момент (доступний, в ремонті, тимчасово недоступний, знятий з прокату на постійній основі);
- `state_detail` – деталі технічного стану автомобіля;
- `enginevolume` – обсяг двигуна автомобіля;
- `allwheeled` – позначка про те, чи має автомобіль повний привід, чи ні;
- `transmission` – тип коробки передач;
- `fueltype` – тип пального;
- `gpsssystem` – наявність системи GPS-координації;
- `passengers` – кількість пасажирів разом з водієм, які можуть одночасно перевозитися в автомобілі;
- `doors` – кількість дверей у автомобілі;
- `airsystem` – система керування температурою повітря, що використовується в автомобілі;
- `childseat` – доступність дитячого крісла в автомобілі;
- `sunroof` – наявність люка в автомобілі;
- `usbinput` – наявність USB-входів у автомобілі;
- `usbcharger` – можливість заряджання пристроїв від USB у автомобілі;
- `tiresseason` – тип шин, які встановлені на автомобілі, за сезоном;
- `keyless` – можливість безключового доступу і керування автомобілем;
- `blindspotwarning` – наявність системи повідомлення про сліпі зони;
- `backupcamera` – наявність камери заднього виду;
- `musicsystem` – наявність музичної системи в автомобілі;

- longterm – можливість довгострокового прокату автомобіля (більше одного місяця);
- heatedseats – наявність обігріву сидінь;
- bluetooth – можливість керування через Bluetooth;
- cancellationpolicy – політика скасування резервування прокату, що використовується для даного автомобіля;
- autocontrol – тип автоматичного керування автомобілем (відсутній, круїз-контроль, автопілот);
- photo – фотографія автомобіля;
- detalization – детальний опис автомобіля, що має демонструватися у вигляді відповідного оголошення про прокат;
- petfriendly – можливість перевезення домашніх тварин у салоні автомобіля;
- smoking – можливість палити у салоні автомобіля;
- bodycolor – колір кузова автомобіля;
- tripstotal – сумарна кількість прокатів автомобіля, яка була здійснена з підтримкою програми на даний момент;
- minimumage – мінімальний вік клієнта, який може взяти автомобіль на прокат;
- deliveryprice – вартість доставки автомобіля самим власником клієнту за його адресою;
- maxdelivdistance – максимальна відстань, на яку готовий доставити автомобіль клієнту власник.

Від власника залежить те, де він готовий віддати свій автомобіль для прокату клієнту. Локації, з яких готовий віддати власник свій автомобіль зберігаються в БД у таблиці `carentsoft_vehiclepickup`, яка має поля:

- id – ідентифікатор локації, доступної для надання автомобіля в прокат;

- vehicle – автомобіль, для якого доступна ця локація, що є зовнішнім ключем з таблиці `arentsoft_vehicle`, з якою встановлено зв'язок типу один-до-багатьох;

- city – місто, в якому знаходиться локація, що є зовнішнім ключем з таблиці `arentsoft_city`, з якою встановлено зв'язок типу один-до-багатьох;

- place – адреса локації, в районі якої можна забрати автомобіль;

- location1 – довгота координати локації, в районі якої можна забрати автомобіль;

- location2 – широта координати локації, в районі якої можна забрати автомобіль;

- freedistance – максимальна відстань відносно даної локації, куди готовий без додаткової плати доставити автомобіль власник;

- price – вартість основної плати за доставку від розташування автомобіля на дану локацію;

- state – визначає, чи доступна ця локація на даний момент.

Автомобіль надається в прокат власником не з обмеженими правами, а зокрема з обмеженням відстані, яку на ньому можна проїхати за період, інакше така експлуатація буде порушенням правил. Дозволена відстань, яку за певний період може подолати клієнт, зберігається в таблиці `arentsoft_distance`, яка має поля:

- id – ідентифікатор правила щодо дозволеної відстані прокату;

- vehicle – автомобіль, для якого встановлено правило, що є зовнішнім ключем з таблиці `arentsoft_vehicle`, з якою встановлено зв'язок типу один-до-багатьох;

- maxduration – максимальна тривалість прокату, за якої можна проїхати цю відстань;

- maxdistance – максимальна відстань, на яку можна подорожувати за цей період.

Власники можуть мати декілька автомобілів, які вони виставляють на прокат, а дані про них самих зберігаються в таблиці `arentsoft_proprietor`, яка має поля:

- `id` – ідентифікатор облікового запису власника;
- `appname` – ім'я власника, що відображається в програмі;
- `joined` – дата додавання власника до програми, тобто фактично дата реєстрації;
- `trips` – кількість прокатів, які сумарно було повністю виконано за автомобілем цього власника;
- `password` – пароль облікового запису власника;
- `phonenumber` – номер телефону власника;
- `rating` – середнє значення рейтингу, що за результатами усереднення даних з відгуків клієнтів отримав власник за всіма його автомобілями;
- `reviewsnumber` – кількість відгуків, які було створено клієнтами за автомобілями цього власника;
- `photo` – фотографія власника;
- `respondtime` – середній час відповіді власником на початкове резервування прокату автомобіля.

Бренди виробників автомобілів зберігаються в БД у таблиці `arentsoft_manufacturer`, яка має поля:

- `id` – ідентифікатор бренду виробника автомобілів;
- `shortname` – назва бренду.

Моделі автомобілів зберігаються в БД у таблиці `arentsoft_vehiclemodel`, яка має поля:

- `id` – ідентифікатор моделі автомобіля;
- `manufacturer` – бренд, який виготовляє цю модель автомобіля, що є зовнішнім ключем з таблиці `arentsoft_manufacturer`, з якою встановлено зв'язок типу один-до-багатьох;
- `modelname` – назва моделі автомобіля.

Самі резервування і відповідно в подальшому прокати автомобілів зберігаються в таблиці `carentsoft_vehiclebooking`, яка має поля:

- `id` – ідентифікатор резервування прокату автомобіля;
- `client` – клієнт, який зарезервував автомобіль, що є зовнішнім ключем з таблиці `carentsoft_client`, з якою встановлено зв'язок типу один-до-багатьох;
- `vehicle` – автомобіль, який був зарезервований для прокату, що є зовнішнім ключем з таблиці `carentsoft_vehicle`, з якою встановлено зв'язок типу один-до-багатьох;
- `booktime` – час оформлення резервування;
- `responvertime` – час, який пройшов до моменту, коли власник відповів на резервування (підтвердив, скасував його або написав повідомлення у відповідь);
- `checkindate` – дата і час початку резервування прокату;
- `checkoutdate` – дата і час закінчення резервування прокату;
- `price` – підсумкова вартість прокату автомобіля;
- `paid` – сплачена вартість прокату автомобіля;
- `deliveryprice` – вартість доставки автомобіля на локацію, з якої забирає клієнт автомобіль на прокат;
- `deliverylocation` – адреса локації, з якої забирає автомобіль клієнт, що є зовнішнім ключем з таблиці `carentsoft_vehiclepickup`, з якою встановлено зв'язок типу один-до-багатьох;
- `checkoutprice` – вартість доставки автомобіля з локації, на якій повертає клієнт автомобіль з прокату;
- `checkoutlocation` – адреса локації, на якій повертає автомобіль клієнт, що є зовнішнім ключем з таблиці `carentsoft_vehiclepickup`, з якою встановлено зв'язок типу один-до-багатьох;
- `state` – стан резервування прокату автомобіля;
- `cancellationpolicy` – політика скасування прокату, яка застосовується до прокату;
- `instruction` – інструкції щодо прокату автомобіля від власника;

– payment_email – адреса PayPal-акаунта, за якою було сплачено за резервування прокату автомобіля.

Статистика прокату автомобіля за деякий період разом з результатами прогнозування кількості прокатів зберігається в таблиці carentsoft_vehiclestatistics, яка має поля:

- id – ідентифікатор запису про статистику прокату автомобіля;
- vehicle – автомобіль, статистику прокату якого подано, є зовнішнім ключем з таблиці carentsoft_vehicle, з якою встановлено зв'язок типу один-до-багатьох;
- periodtype – тип інтервалу, за яким обчислено статистику (місяць, квартал, рік);
- period – початок періоду, з якого сформовано статистику;
- trips – фактична кількість прокатів автомобіля за цей період, виконаних через програму;
- predictedtrips – прогнозована кількість прокатів автомобіля за цей період, які мають бути виконані через програму;

Усі автомобілі з короткого переліку клієнта зберігаються в таблиці carentsoft_savedvehicles, яка має поля:

- id – ідентифікатор збереження автомобіля в короткому переліку клієнта;
- vehicle – автомобіль, який було збережено, що є зовнішнім ключем з таблиці БД carentsoft_vehicle, з якою встановлено зв'язок типу один-до-багатьох;
- client – клієнт, який зберіг цей автомобіль у власному переліку, є зовнішнім ключем з таблиці carentsoft_client, з якою встановлено зв'язок типу один-до-багатьох.

Автомобіль не обов'язково надається в прокат самостійно. Можуть бути періоди, коли автомобіль доступний, і періоди, коли він не може бути орендований. Періоди, коли автомобіль доступний, зберігаються в таблиці carentsoft_rentalready, яка має поля:

- id – ідентифікатор періоду доступності автомобіля для прокату;
- vehicle – автомобіль, який доступний для прокату, є зовнішнім ключем з таблиці `carentsoft_vehicle`, з якою встановлено зв'язок типу один-до-багатьох;
- periodstart – початок періоду, в який доступний автомобіль для прокату;
- periodend – закінчення періоду, в який доступний автомобіль для прокату.

Вартість прокату може встановлюватися гнучко. Тобто на деякий період вона може бути однією, на інший – змінена, до того ж в межах періоду може визначатися, що прокат на тиждень буде коштувати дешевше ніж вартість щоденного прокату, помножена на 7 днів.

Варіанти вартості прокату зберігаються в таблиці `carentsoft_price`, яка має поля:

- id – ідентифікатор варіанту вартості прокату автомобіля;
- vehicle – автомобіль, вартість якого задана, є зовнішнім ключем з таблиці `carentsoft_vehicle`, з якою встановлено зв'язок типу один-до-багатьох;
- periodstart – початок періоду, з якого ця вартість діє;
- periodend – закінчення періоду, до якого ця вартість діє;
- maxduration – максимальна кількість днів, для якої діє ця вартість прокату;
- minduration – тривалість періоду, для якого визначена вартість (наприклад, день);
- price – вартість мінімальної одиниці тривалості (наприклад, вартість дня оренди).

Комунікація щодо резервувань і прокатів між власниками і клієнтами зберігається в таблиці `carentsoft_request`, яка має поля:

- id – ідентифікатор комунікації;

– vehicle – автомобіль, щодо якого відбулась комунікація з клієнтом, є зовнішнім ключем з таблиці carentsoft_vehicle, з якою встановлено зв'язок типу один-до-багатьох;

– client – клієнт, який комунікував з власником щодо прокату автомобіля, є зовнішнім ключем з таблиці carentsoft_client, з якою встановлено зв'язок типу один-до-багатьох.

Самі повідомлення з відповідної комунікації зберігаються в таблиці carentsoft_requestanswer, яка має поля:

- id – ідентифікатор повідомлення з комунікації;
- request – комунікація щодо резервування, за якою було відправлено повідомлення, є зовнішнім ключем з таблиці carentsoft_request, з якою встановлено зв'язок типу один-до-багатьох;
- requesttime – час надсилання повідомлення;
- respond – текст у цьому повідомленні;
- respondauthor – автором повідомлення є клієнт або власник автомобіля.

Відгуки клієнтів про автомобілі і їх власників зберігаються в таблиці carentsoft_clientreview, яка має поля:

- id – ідентифікатор відгуку клієнта про автомобіль, прокат якого він виконував;
- vehicle – автомобіль, прокат якого виконувався, є зовнішнім ключем з таблиці БД carentsoft_vehicle, з якою встановлено зв'язок типу один-до-багатьох;
- client – клієнт, який брав цей автомобіль на прокат, є зовнішнім ключем з таблиці carentsoft_client, з якою встановлено зв'язок типу один-до-багатьох;
- rating – рейтингова оцінка, виставлена клієнтом;
- detalization – текст, який залишив у відгуку про автомобіль клієнт;
- revdate – час публікації відгуку клієнта про автомобіль.

Відгуки власників автомобілів про клієнтів зберігаються в таблиці `carentsoft_propreview`, яка має поля:

- `id` – ідентифікатор відгуку власника автомобіля про клієнта;
- `client` – клієнт, з яким взаємодіяв власник автомобіля, надавши йому власний автомобіль на прокат, є зовнішнім ключем з таблиці `carentsoft_client`, з якою встановлено зв'язок типу один-до-багатьох;
- `proprietor` – власник автомобіля, який створив цей відгук, є зовнішнім ключем з таблиці `carentsoft_proprietor`, з якою встановлено зв'язок типу один-до-багатьох;
- `rating` – рейтингова оцінка, виставлена клієнту у відгуку власником;
- `detalization` – текст, який залишив у відгуку про клієнта власник;
- `revdate` – час публікації відгуку власника про клієнта.

Відповіді власників на відгуки клієнтів про автомобілі зберігаються в таблиці `carentsoft_reviewanswer`, яка має поля:

- `id` – ідентифікатор відповіді власника автомобіля на відгук клієнта про його автомобіль;
- `review` – відгук клієнта про автомобіль, на який відреагував його власник, є зовнішнім ключем з таблиці `carentsoft_clientreview`, з якою встановлено зв'язок типу один-до-одного;
- `proprietor` – власник автомобіля, який створив цю відповідь, є зовнішнім ключем з таблиці `carentsoft_proprietor`, з якою встановлено зв'язок типу один-до-багатьох;
- `answer` – текст відповіді власника на відгук клієнта про його автомобіль;
- `ansdate` – дата публікації відповіді в програмі.

Самі ж клієнти представлені в БД своїми обліковими записами і накопиченими даними про свою роботу, що зберігаються в таблиці `carentsoft_client`, яка має поля:

- `id` – ідентифікатор облікового запису клієнта в програмі прокату автомобілів;

- name – ім'я клієнта, що використовується для відображення в програмі;
- full_name – прізвище клієнта, що використовується для відображення в програмі;
- joined – дата приєднання клієнта до програми, тобто дата його реєстрації;
- trips – кількість разів, коли клієнт через програму брав автомобіль на прокат;
- birthday – дата народження клієнта;
- password – пароль, який використовується для роботи з цим обліковим записом клієнта;
- phonenumber – номер телефону клієнта;
- rating – усереднена рейтингова оцінка, виставлена клієнту за всіма відгуками власників автомобілів, які він брав на прокат, про нього;
- reviewsnumber – кількість відгуків, які були написані про клієнта власниками автомобілів, які він брав на прокат;
- state – можливість доступу клієнтом до власного облікового запису (заблокований чи ні);
- photo – фотографія клієнта.

Для зручності пошуку також було виділено окрему таблицю `carrentsoft_city`, де зберігається перелік міст, в яких може відбуватися прокат, до складу таблиці входить:

- поле ідентифікатора `id`;
- поле `name` з назвою міста.

На основі сформульованої структури була визначена схема БД (рис. 3.2), яка використовувалась у подальшому при розробці самого застосунку.



Рисунок 3.2 – Схема БД

3.4 Висновки за розділом 3

Сукупність технологій і інструментів, які обрано під час проектування програмного забезпечення, включає мову Python, фреймворк Django, систему керування БД PostgreSQL.

У даному розділі для неавторизованого користувача, клієнта та власника автомобілів визначено сценарії їх використання програми з

представленням відповідної діаграми, обрана структура БД, що повністю враховує функціональні вимоги, з якими відповідні основні сутності предметної області мають працювати.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Структура розробленої програмної системи

Фреймворк Django визначає базову структуру вебзастосунку, створеного на його основі. Проте враховуючи функціональність програмного забезпечення, реалізацію методу, було внесено певні додаткові особливості до базової структури (рис. 4.1).

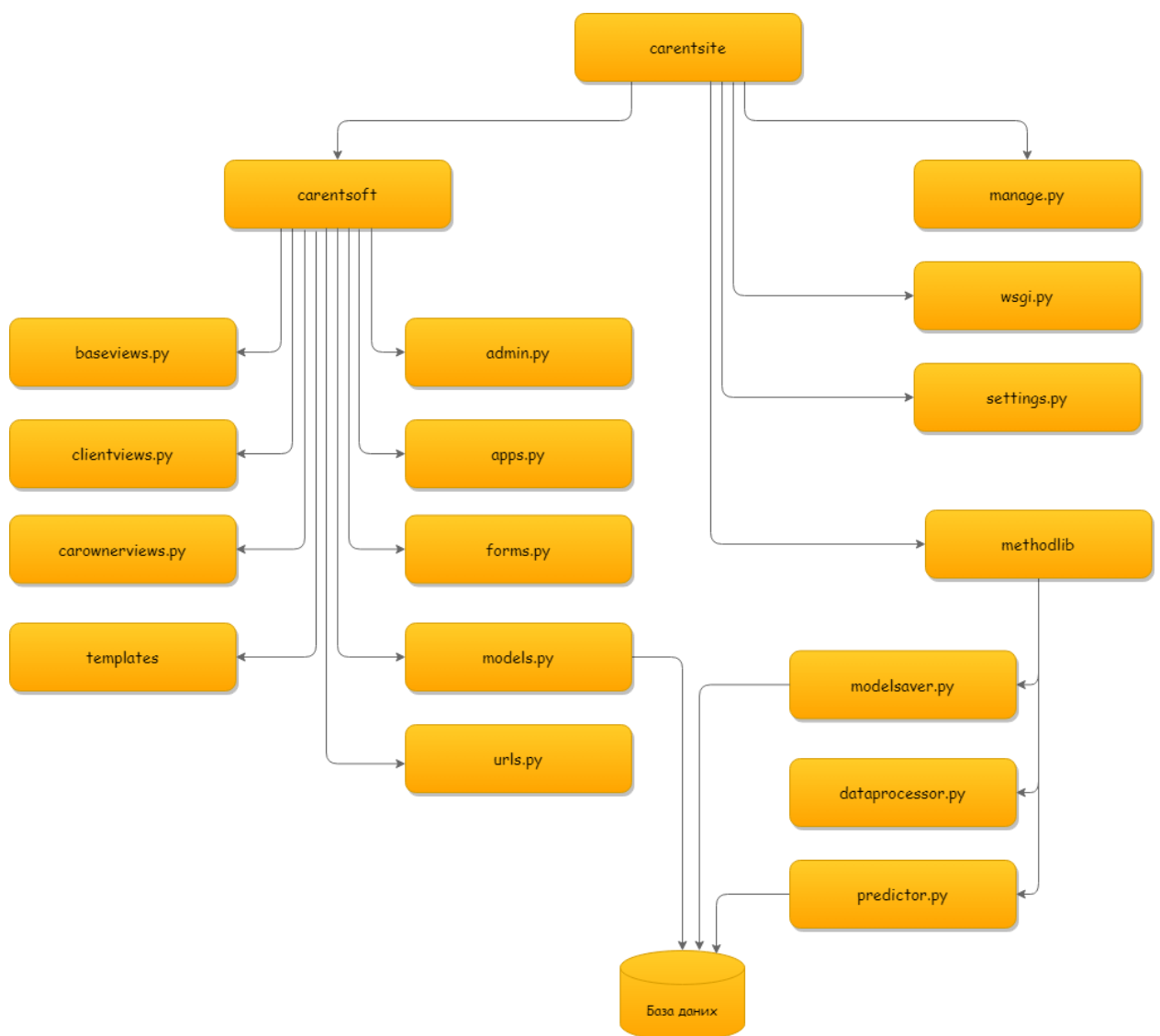


Рисунок 4.1 – Структура розробленої програмної системи

По-перше, в складі загального проєкту `carentsite` було окрім самого застосунку `carentsoft`, що реалізує основну функціональність предметної області, визначено бібліотеку `methodlib`, яка реалізує метод, описаний в розділі 2. В цій структурі створено файли `dataprocessor.py`, який обробляє і підготовляє дані для роботи методу, `modelsaver.py`, який виконує навчання і зберігання моделі в файл, `predictor.py`, який виконує завантаження моделі з файлу, звернення до БД з метою отримання даних і прогнозування на їх основі. Для реалізації цієї функціональності використано пакети `lightgbm` для застосування алгоритму LightGBM, `numpy` для базової роботи з масивами, в яких представлено дані, `pandas` для читання даних з файлу вибірки та `scipy` для оброблення даних.

По-друге, представлення вебзастосунку було розбито на три файли, кожен з яких визначає представлення для відповідного користувача: `baseviews.py` містить представлення для неавторизованого користувача, `clientviews.py` – для клієнта, `managerviews.py` – для власника.

4.2 Реалізовані класи програмної системи

Дані для профіля клієнта разом з відгуками про нього формуються в класі представлення `ClientProfileView`, що є реалізацією на основі стандартного класу `DetailView` через його наслідування. Клас `DetailView` дозволяє одразу заповнити даними профіля клієнта з класу моделі `Client` контекст даних. У випадку, якщо обліковий запис авторизованого користувача є обліковим записом клієнта і він не співпадає з обліковим записом, деталізація його виконується на сторінці, тобто ця сторінка не є власним профілем авторизованого клієнта, то номер телефону зачищається з даних об'єкту клієнтського профіля. Для всіх випадків зачищаються також дані пароля облікового запису. За датою народження обчислюється поточний повний вік клієнта і вказується в атрибуті `years` клієнтського облікового запису. Додатково до цих даних у контексті даних для вебсторінки за ключем `reviews`

розташовуються всі відгуки про даного клієнта, написані власниками, у яких брав на прокат автомобіль цей клієнт.

За подібною логікою дані для профіля власника разом з відгуками про нього формуються в класі представлення `ProprietorProfileView`, що є реалізацією на основі стандартного класу `DetailView` через його наслідування. Дані профіля власника отримуються з класу моделі `Proprietor`. У випадку, якщо обліковий запис авторизованого користувача є обліковим записом власника автомобіля і він не співпадає з обліковим записом, деталізація його виконується на сторінці, тобто ця сторінка не є власним профілем авторизованого власника, то номер телефону зачищається з даних об'єкту клієнтського профіля. Для всіх випадків зачищаються також дані пароля облікового запису. Додатково до цих даних у контексті даних для вебсторінки за ключем `vehicles` розташовуються всі автомобілі даного власника, а за ключем `reviews` розташовуються всі відгуки про даного власника і його автомобілі, написані його клієнтами.

Також на основі класу `DetailView` побудоване представлення `VehicleDetailView`, в якому для сторінки детального перегляду даних автомобіля, прокат якого пропонується, витягаються дані, які окрім даних самого об'єкта відповідної моделі `Vehicle` включають також:

- перелік локацій, з яких готовий віддати або після прокату забрати власник свій автомобіль, які обираються з множини відповідних об'єктів через відфільтрування всіх, які не стосуються даного автомобіля, доступний за ключем `locations`;
- перелік дозволених варіантів відстані, яку за певний період може подолати клієнт на цьому автомобілі, доступний за ключем `distances`;
- перелік дат, коли автомобіль готовий до прокату, доступний за ключем `ready`, що використовується для формування календаря, з якого клієнт може обирати дати і потім відповідно час прокату;
- перелік 5 останніх відгуків про даний автомобіль від клієнтів, які його орендували, доступний за ключем `client_reviews`;

- сумарна вартість прокату на встановлений інтервал дат, що визначаються через параметри bookstarttime GET-запиту, доступна за ключем rental_price;
- вартість доставки автомобіля для прокату клієнту на локацію, що визначається через параметр delivlocation GET-запиту, доступна за ключем доступний за ключем deliveryprice;
- вартість доставки автомобіля після прокату з локації, що визначається через параметр outlocation GET-запиту, доступна за ключем доступний за ключем checkoutprice.

За цими ключами відбувається доступ до даних з шаблону відповідної вебсторінки vehicle.html. Шаблоном для побудови макетів сторінок виступив шаблон [14].

Для того щоб визначити вартість прокату використовується наступний алгоритм у даному класі. Спочатку обчислюється інтервал між датою і часом початку прокату і датою і часом завершення прокату, після чого це значення переводиться у кількість годин, яка знаходиться між цими миттєвостями. Далі формується перелік з усіх варіантів вартості прокату, які стосуються даного автомобіля, мають початок інтервалу раніше дати і часу початку прокату, його закінчення пізніше дати і часу завершення прокату та максимальну тривалість прокату більшу за кількість днів між необхідними датами для прокату. Усі варіанти сортуються за максимальною тривалістю і варіант з найменшою тривалістю обирається в якості джерела даних, тобто це варіант, який за тривалістю прокату є найближчим варіантом, що переважає заданий період. З цього варіанта визначається вартість одиниці і тривалість одиниці часу в годинах. Тоді визначається кількість годин у заданому періоді прокату і на основі цих даних обчислюється вартість прокату, до якої додається вартість доставки до необхідної локації та вартість доставки з локації після завершення прокату.

У свою чергу клас представлення ProprietorVehiclesView побудований на основі іншого базового класу – ListView, який підтримує роботу з

переліками об'єктів, що належать моделі `Vehicle`. Цей клас готує перелік об'єктів, які представляють автомобілі заданого власника. Ідентифікатор власника задається через параметр `prop` GET-запиту. Усі автомобілі сортуються за ідентифікатором, починаючи з найбільшого.

Клас представлення `FullVehicleReviews` також виконує підготовку переліку об'єктів, але ці об'єкти є всіма відгуками, які стосуються заданого через параметр `vehicle` GET-запиту ідентифікатора автомобіля. Усі відгуки клієнтів сортуються, починаючи з останніх опублікованих.

Клас представлення `CityBestVehiclesView` формує перелік автомобілів, які слід вважати найкращими в місті, ідентифікатор якого передається через параметр `city` GET-запиту. Перелік обмежується 100 автомобілями з найвищим рейтингом у місті.

Клас представлення `CityPopularVehiclesView` формує перелік автомобілів, які мають найбільшу кількість прокатів у місті, ідентифікатор якого передається через параметр `city` GET-запиту. Перелік обмежується 100 автомобілями з найбільшою кількістю оформлених резервувань прокатів.

Підготовка результатів пошуку автомобілів за потребами клієнта виконується в класі представлення `VehicleFindView`. Формування контексту даних з результатами забезпечується через метод `get_queryset`. Під час пошуку використовуються наступним чином наступні параметри, отримані через GET-запит:

- параметр `vmodel`, який задає за необхідності конкретну модель автомобіля, задану в формі пошуку `VehicleSearchForm` прямим вибором або вибором спочатку виробника автомобіля, а потім самої моделі;
- параметр `brand`, який задає за необхідності конкретний бренд, тобто конкретного виробника автомобілів, заданого в формі пошуку вибором з переліку;
- параметр `vehetypes`, який задає типи транспортних засобів: типи представлені відповідними числовими значеннями, переліченими через кому,

цей рядок розбирається і в результатах пошуку залишаються тільки ті автомобілі, які відповідають одному з перелічених типів;

- параметр `minrating`, який задає мінімальне прийнятне для клієнта середнє значення рейтингу автомобіля: залишаються автомобілі з рейтингом, який не нижче за заданий;

- параметр `maxrating`, який задає максимальне прийнятне для клієнта середнє значення рейтингу автомобіля: залишаються автомобілі з рейтингом, який не вище за заданий;

- параметр `city`, який задає ідентифікатор міста, в якому клієнт хоче взяти в прокат автомобіль;

- параметр `minprodyear`, який задає найстарший можливий рік виробництва автомобіля: залишаються автомобілі, які були вироблені не раніше заданого року;

- параметр `maxprodyear`, який задає найновіший можливий рік виробництва автомобіля: залишаються автомобілі, які були вироблені не пізніше заданого року;

- параметр `minengine`, який задає найменший допустимий обсяг двигуна автомобіля: залишаються автомобілі, які мають двигун обсягом не менше за заданий;

- параметр `maxengine`, який задає найбільший допустимий обсяг двигуна автомобіля: залишаються автомобілі, які мають двигун обсягом не більше за заданий;

- параметр `allwheeled`, який задає потребу в автомобілі з повним приводом: нульове значення визначає, що автомобіль може мати будь-який варіант приводу, значення 1 призводить до того, що залишаються тільки повноприводні автомобілі, а значення 2 – автомобілі без повного приводу;

- параметр `transmission`, який задає тип коробки передач: нульове значення не призводить до відбору, тобто залишаються всі автомобілі, а інші значення встановлюють відповідний тип коробки передач, залишаючи в результаті тільки автомобілі з цим типом коробки передач;

– параметр `fueltypes`, який задає типи пального, з яким готовий взяти автомобіль на прокат клієнт: типи представлені відповідними числовими значеннями, переліченими через кому, цей рядок розбирається і в результатах пошуку залишаються тільки ті автомобілі, які використовують один з типів переліченого пального;

– параметр `gpssystem`, який визначає наявність системи GPS-координації: нульове значення не призводить до відбору, тобто залишаються всі автомобілі, значення 1 встановлює обов'язкову наявність системи GPS-навігації, а значення 2 – обов'язкову відсутність;

– параметр `minpassengers`, який визначає мінімальну необхідну кількість місць в автомобілі: залишаються автомобілі, які мають не меншу кількість місць за задану;

– параметр `maxpassengers`, який визначає максимальну необхідну кількість місць в автомобілі: залишаються автомобілі, які мають не більшу кількість місць за задану;

– параметр `mindoors`, який визначає мінімальну необхідну кількість дверей у автомобілі: залишаються автомобілі, які мають не меншу кількість дверей за задану;

– параметр `maxdoors`, який визначає максимальну необхідну кількість дверей у автомобілі: залишаються автомобілі, які мають не більшу кількість дверей за задану;

– параметр `airsystem`, який задає систему керування температурою повітря, що використовується в автомобілі: нульове значення не призводить до відбору, тобто залишаються всі автомобілі, а інші значення встановлюють відповідний тип системи керування температурою повітря, залишаючи в результатах тільки автомобілі з цим типом системи керування температурою повітря;

– параметр `childseat`, який визначає доступність дитячого крісла в автомобілі: нульове значення або значення `False` не призводить до відбору,

тобто залишаються всі автомобілі, значення True встановлює обов'язкову доступність такого крісла;

- параметр sunroof, який визначає наявність люка в автомобілі: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову наявність люка;

- параметр usbinput, який визначає наявність USB-входів у автомобілі: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову наявність USB-входів;

- параметр usbcharger, який визначає можливість заряджання пристроїв від USB у автомобілі: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову можливість USB-зарядки;

- параметр tiresseason, який задає тип шин, які встановлені на автомобілі, за сезоном: нульове значення або значення False залишає всі автомобілі, а інші значення встановлюють відповідний тип шин;

- параметр keyless, який визначає можливість безключового доступу і керування автомобілем: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову можливість безключового доступу;

- параметр blindspotwarning, який визначає наявність системи повідомлення про сліпі зони: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову наявність системи повідомлення про сліпі зони;

- параметр backupcamera, який визначає наявність камери заднього виду: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову наявність камери заднього виду;

- параметр musicsystem, який задає наявність музичної системи в автомобілі: нульове значення залишає всі автомобілі, а інші значення встановлюють наявність відповідної музичної системи;

- параметр longterm, який визначає можливість довгострокового прокату автомобіля: нульове значення або значення False залишає всі

автомобілі, значення True встановлює обов'язкову можливість довгострокового прокату в майбутньому;

- параметр `heatedseats`, який визначає наявність обігріву сидінь: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову наявність обігріву сидінь;

- параметр `bluetooth`, який визначає можливість керування через Bluetooth: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову можливість керування через Bluetooth;

- параметр `cancellationpolicy`, який задає політику скасування резервування прокату: нульове значення залишає всі автомобілі, а інші значення встановлюють точну політику скасування (повністю, з поверненням 10 % або 50 %);

- параметр `autocontrol`, який задає тип автоматичного керування автомобілем: нульове значення залишає всі автомобілі, а інші значення встановлюють тип автоматичного керування (відсутній, круїз-контроль, автопілот);

- параметр `petfriendly`, який визначає можливість перевезення домашніх тварин у салоні автомобіля: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову можливість перевезення домашніх тварин у салоні автомобіля;

- параметр `smoking`, який визначає можливість палити у салоні автомобіля: нульове значення або значення False залишає всі автомобілі, значення True встановлює обов'язкову можливість палити у салоні автомобіля;

- параметр `bodycolor`, який задає колір кузова автомобіля: нульове значення не призводить до відбору, тобто залишаються всі автомобілі, а інші значення встановлюють відповідний колір кузова автомобіля, залишаючи в результатах тільки автомобілі з цим кольором кузова;

- параметр `mintrips`, який визначає мінімальну сумарну кількість прокатів автомобіля: залишаються автомобілі, які мають не меншу кількість прокатів за задану;
- параметр `maxtrips`, який визначає максимальну сумарну кількість прокатів автомобіля: залишаються автомобілі, які мають не більшу кількість прокатів за задану;
- параметр `minprice`, який визначає мінімальну вартість прокату автомобіля: залишаються автомобілі, вартість прокату яких не менше заданого значення;
- параметр `maxprice`, який визначає максимальну вартість прокату автомобіля: залишаються автомобілі, вартість прокату яких не більше заданого значення;
- параметр `bookstarttime`, який визначає дату і час початку прокату;
- параметр `bookendtime`, який визначає дату і час закінчення прокату.

Вартість прокату під час пошуку розраховується тільки на основі сплати за саме використання автомобіля. У класі представлення `VehicleDetailView` розрахунок виконується з врахуванням вартості доставки і повернення автомобіля для видачі клієнту.

Класи представлень, які забезпечують підтримку процесів змінювання або створення відповідних об'єктів реалізовані через наслідування стандартного класу представлення `View` та класу `LoginRequiredMixin`, що вимагає авторизації для доступу до відповідної сторінки. Такими класами представлень є класи `VehicleEditView`, що підтримує змінювання параметрів прокату автомобіля, `VehiclePublishView`, що підтримує виставлення власного автомобіля на прокат, `ClientreviewPublishView`, що підтримує публікацію відгуку клієнта про прокат автомобіля і інші подібні класи. У процесі публікації відгуку клієнта одразу переобчислюється рейтинг автомобіля, рейтинг власника, кількість відгуків про автомобіль і кількість відгуків за всі автомобілі власника.

Для підтримки і реалізації платежів через платіжну систему PayPal було створено ряд класів представлень.

Клас представлення `RentalPaymentView` забезпечує підтримку базового платежу для сплати вартості прокату, організований через наслідування класу `DetailView`.

У методі `get` задається словник, який передається в клас `PayPalPaymentsForm`, який запускає на виконання сам платіж. Словник включає набір параметрів, зокрема параметр `amount`, що визначає суму, яка сплачується в платежі, параметр `business`, що визначає електронну адресу PayPal-рахунку, на яку сплачується сума, параметр `invoice`, що містить ідентифікатор резервування, за який виконується сплата. Резервування завантажується за його ідентифікатором, значення якого передається через параметр `pk` даного методу.

Для оброблення результатів виконання платежу використовуються сигнали `Instant Payment Notification (IPN)` [15]-[17].

Коли IPN-сигнал отримується за результатами платежу, то його оброблення здійснюється через функцію `payment_result_processor`. У цій функції відбувається спочатку перевірка того, що стан платежу дорівнює `ST_PP_COMPLETED`, тобто він був виконаний успішно. У разі, якщо це правда, перевіряється, що електронна адреса облікового запису PayPal, на яку було виконано платіж, дорівнює тій, що в налаштуваннях виставлена базовою для застосунку. Якщо це все так, то далі відбувається внесення на рахунок суми, яка була фактично отримана після платежу і яка міститься в атрибуті `mc_gross` результату. Окрім того відбувається перевірка поточного стану і він змінюється для фіксування виконання платежу. Аналогічні дії відбувається в представленнях, пов'язаних з поверненням коштів у випадку скасування резервування.

4.3 Висновки за розділом 4

Структура розробленого вебзастосунку відповідає структурі Django-застосунків. У ній для кожної складової шаблону проєктування MVT було розроблено відповідні представлення, основна частина з яких була детально описана в розділі, моделі, які дублюють структуру БД, шаблони сторінок, менеджер URL.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма надає можливість кожному конкретному власнику отримати доступ до ринку прокату автомобілів, на якому він може тимчасово віддати власний автомобіль іншим користувачам у прокат за плату, яку він вважає адекватною для даної послуги.

Користувачу, який бажає взяти автомобіль на прокат, програма надає можливість знайти такий варіант автомобіля, який відповідає його вимогам, і зарезервувати його для подальшого прокату.

5.2 Умови виконання програми

Сервер потребує обчислювальну потужність не меншу ніж процесор, який має не менше ніж 4 ядра, частоту від 2 ГГц, оперативну пам'ять, що має бути не меншою від 4 Гб, жорсткий диск, де має бути доступно не менше ніж 2 Гб.

5.3 Виконання програми

Для виконання програми обов'язково має бути спочатку запущено серверну частину, що включає сервер БД та вебсервер. Якщо вони не запущені на виконання, то ні клієнт, ні неавторизований користувач, ні власник не зможуть взаємодіяти з вебзастосунком.

Після запуску серверної частини за звернення користувача до вебзастосунку за адресою вебсервера відображається спочатку домашня сторінка, яка в меню має назву «Сервіс». Вона містить опис загалом ідеї застосунку та відображення найбільш популярних автомобілів у місті, яке можна обрати з випадającego переліку, або найрейтинговіших автомобілів.

Також неавторизованого користувачу доступний розділ «Пошук», який дозволяє задати параметри прокату, які зокрема були детально описані в розділі 3, та отримати інформацію про можливість прокату тих автомобілів, які відповідають цим параметрам. Аналогічний розділ доступний і клієнту після авторизації (рис. 5.1).

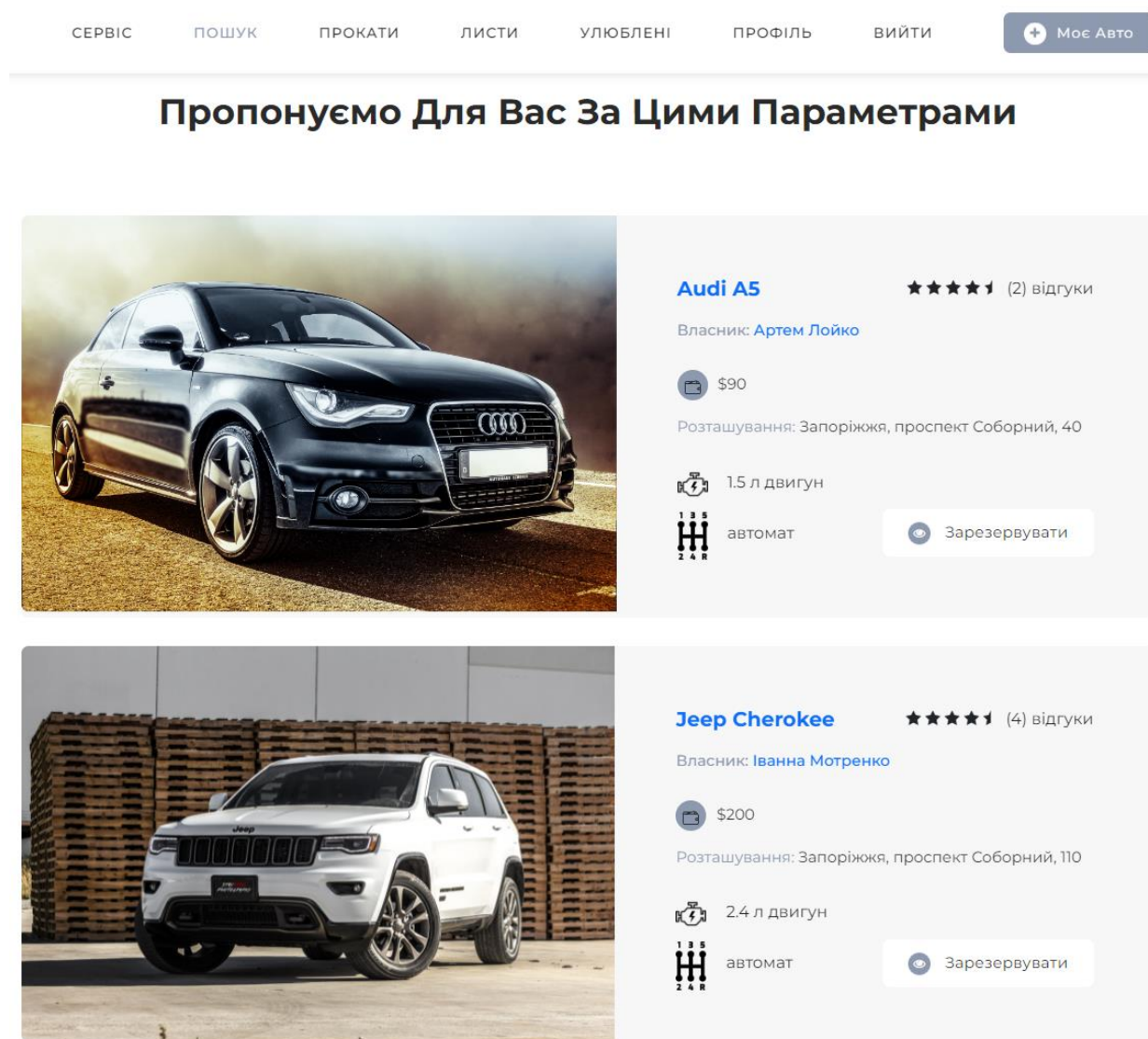


Рисунок 5.1 – Вебсторінка результатів пошуку автомобіля для прокату

За натискання на назву бренду і моделі автомобіля відкривається сторінка з інформацією про автомобіль (рис. 5.2). За натискання на ім'я власника відкривається сторінка профіля власника. Сторінка з інформацією про автомобіль містить чотири кнопки, які перемикають відображення

інформації з основної, де можна зберегти автомобіль улюблених, написати власнику, визначити період прокату та перейти до резервування прокату (рис. 5.2), на детальну (відображає значення всіх параметрів, відомих для автомобіля), інформацію про власника (відображає основну інформацію і можливість переходу до профіля) та перелік відгуків про цей автомобіль разом з оцінкою автомобіля.

СЕРВІС

ПОШУК

ПРОКАТИ

ЛИСТИ

УЛЮБЛЕНІ

ПРОФІЛЬ

ВИЙТИ

+

Моє Авто

🏠

Основне

🚗

Детально

👤

Власник

👜

Відгуки



Range Rover

Evoque (2020)

Зберегти

Автомобіль в ідеальному стані. Дозволить подорожувати без проблем по будь-якій місцевості. Ви зможете подолати і бездоріжжя, і прекрасно почувати себе в місті. Шкіряний інтер'єр, презентабельний вигляд - допоможуть вам у будь-якій ситуації.

Я гнучко вирішую додаткові проблеми і питання. Якщо вони є, просто напишіть мені.

✍️

Написати власнику

Початок прокату

20.12.2022

📅

Кінець прокату

22.12.2022

📅

10:00

🕒

10:00

🕒

Місце

у власника

▼

Місце

у власника

▼

Доставка до: 0 \$

Доставка від: 0 \$

Сумарна вартість: 100 \$

Зарезервувати

Рисунок 5.2 – Вебсторінка інформації про автомобіль

Зарезервувати, написати власнику або зберегти автомобіль в улюблених неавторизований користувач не зможе.

Після вибору резервування прокату потрібно ввести дані PayPal-рахунку і сплатити за резервування.

Розділ «Прокати» містить перелік всіх резервувань прокатів автомобілів авторизованого користувача, де за допомогою кнопок можна переходити до минулих, майбутніх, сплачених або скасованих резервувань. У випадку, якщо прокат ще не розпочався, клієнт може скасувати свої резервування. Після того, як прокат завершився, клієнт може написати відгук. Цей розділ не доступний неавторизованим користувачам, як і розділ «Листи», де відображаються комунікації клієнта з власниками, «Улюблені», де відображається перелік збережених автомобілів.

За допомогою кнопки «Моє авто» можна перейти до вебсторінки реєстрації облікового запису власника, а звідти до виставлення першого автомобіля. Також можна зі сторінки авторизації клієнта перейти за кнопкою до авторизації або реєстрації облікового запису власника.

Розділ «Автомобілі» для власника містить перелік всіх автомобілів, які він виставив на прокат. Звідси можна для кожного автомобіля встановлювати різні параметри (період, коли можна виконувати прокат, вартість тощо), змінювати загалом параметри автомобіля.

Розділ «Резервування» підтримує відповідні резервування за всіма автомобілями власника одразу. Власник може скасовувати резервування, може змінювати стан резервування, коли автомобіль відданий клієнту або повернутий. Після завершення прокату стає доступною кнопка для написання відгуку про клієнта.

Розділ «Листи» власника містить відповідну комунікації з клієнтами.

Розділ «Статистика» (рис. 5.3) містить інформацію про кількість виконаних прокатів автомобілів власника за кожен місяць останнього року. Останнє значення відповідно представляє результат прогнозування. За кожний минулий місяць відображається кількість прокатів, які в цей місяць були здійснені фактично, тобто клієнт забрав і повернув автомобіль. Якщо прокат

здійснювався на межі місяця, то він рахувався і в минулий, і в наступний місяць.

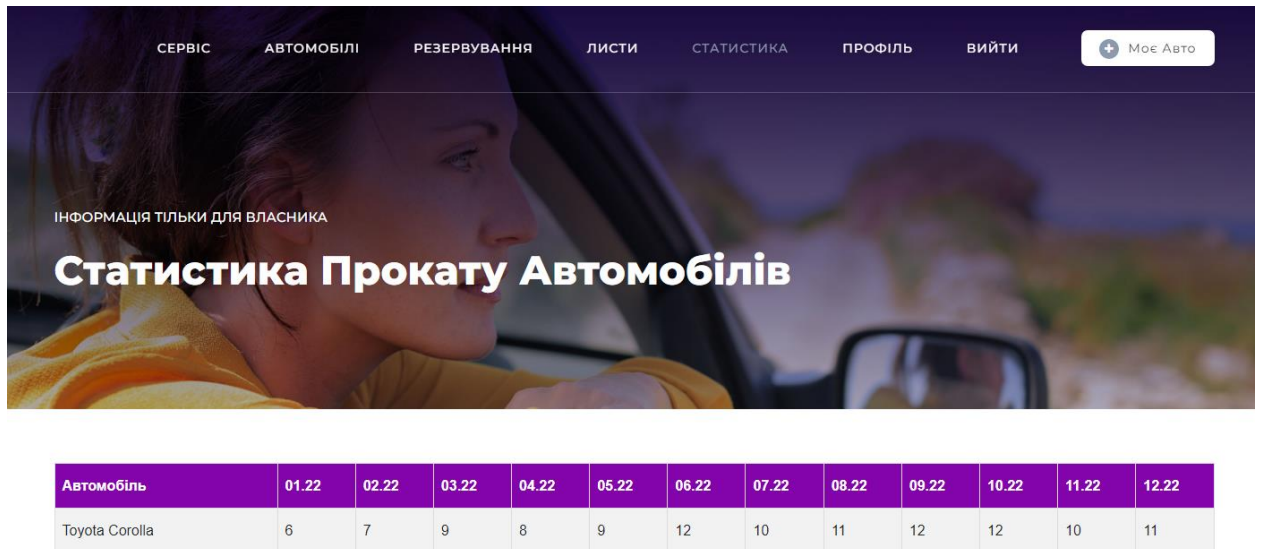


Рисунок 5.3 – Статистика і прогнозування кількості прокатів автомобілів

5.4 Експериментальне дослідження прогнозування попиту на прокат автомобілів

Виконання експериментального дослідження прогнозування попиту на прокат автомобілів було забезпечено на основі використання вибірки даних [18], яка містить дані прокату автомобілів у містах у Сполучених Штатах Америки. Дані були накопичені шляхом витягання з відповідних вебсайтів, а тому під час проведення дослідження вибірка була таким саме шляхом розширена.

Вибірка даних містить зокрема ознаки, що включають тип палива автомобіля, середню оцінку, кількість прокатів, місто і країну, середню добову вартість прокату, модель і бренд автомобіля, тип автомобіля, рік виробництва.

Вибірка була спочатку докладно проаналізована з вивченням впливу різних ознак вибірки на попит, тобто кількість прокатів автомобілів.

На діаграмі розсіювання для залежності кількості прокатів від середньої оцінки автомобілів за вибіркою даних (рис. 5.4) помітно, що

автомобілі з вищою середньою оцінкою загалом частіше беруться на прокат. На діаграмі також помітно, що невелика кількість прокатів може бути за середньої оцінки у 5 балів (правий нижній кут). Це випадки, коли автомобіль отримав тільки декілька відгуків, а кількість прокатів на поточний момент ще є невеликою. Дані агреговані за всі періоди, тому деякі екземпляри характеризують ситуації, коли автомобіль ще не набрав популярності, проте на поточний момент має максимальну середню оцінку, яку наприклад, виставив один клієнт.

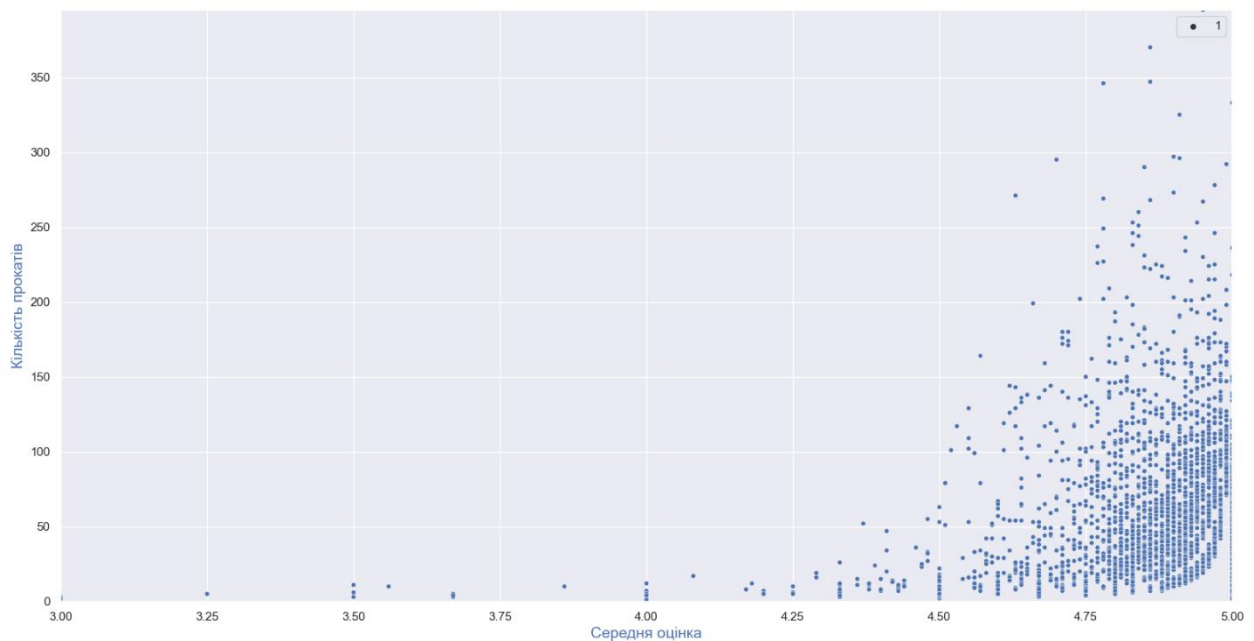


Рисунок 5.4 – Діаграма розсіювання для залежності кількості прокатів від середньої оцінки автомобілів за вибіркою даних

Діаграма розсіювання для залежності кількості прокатів від вартості доби прокату автомобіля за вибіркою даних (рис. 5.5) демонструє, що автомобілі з меншою вартості доби за оренду загалом мають більшу ймовірність високого попиту. Зі зростанням вартості кількість прокатів загалом зменшується.

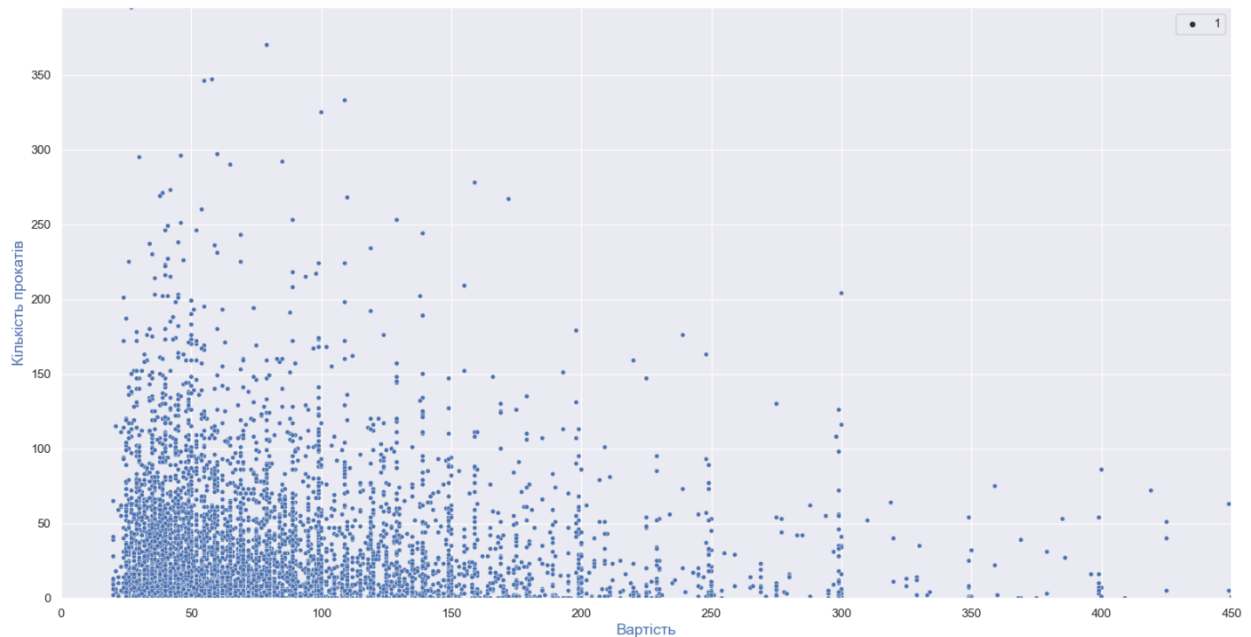


Рисунок 5.5 – Діаграма розсіювання для залежності кількості прокатів від вартості доби прокату автомобіля за вибіркою даних

Метод прогнозування попиту на прокат автомобілів дозволив отримати значення коефіцієнта R^2 рівне 0,819, що є достатньо точним, адже це значення близьке до 1 і одночасно на 21,7 % перевищує результат методу опорних векторів (табл. 5.1).

Таблиця 5.1 – Результати експериментального дослідження

Метод	Коефіцієнт R^2
Метод опорних векторів	0,673
Метод прогнозування попиту на прокат автомобілів	0,819

5.5 Висновки за розділом 5

Логіка взаємодії як неавторизованих користувачів, так і клієнтів, і власників автомобілів з програмою є достатньо простою: вона була детально описана в розділі. За експериментальним дослідженням метод прогнозування попиту на прокат автомобілів виявився достатньо точним для його

рекомендування для застосування на практиці. Попередньо було проаналізовано вибірку даних, результати якого було покладено в процес розроблення даного методу.

ВИСНОВКИ

Дипломна кваліфікаційна робота магістра направлена на розробку програмного забезпечення підтримки прокату автомобілів для допомоги власникам автомобілів під час виставлення їх на прокат. Така підтримка включає як підтримку виконання резервувань прокату, так і підтримку прийняття рішень власниками щодо параметрів прокату, які принесуть їм достатню користь.

Розроблене програмне забезпечення дозволяє як виставляти власній автомобілі на прокат, знаходити адекватні варіанти автомобілів для прокату на основі розширеного набору параметрів пошуку, так і виконувати резервування цих автомобілів. Резервування забезпечується не тільки через створення певного графіку прокату, але і за допомогою підтримки даного процесу через сплату за послуги з використанням платіжної системи PayPal. Впроваджено як функціональність сплати за резервування, так і функціональність повернення внесених коштів за різними правилами через скасування резервувань, якщо вони ще не почали виконуватися. Для підтримки власників і клієнтів створено можливості залишити відгуки про клієнтів власникам, про автомобілі – клієнтам, переглянути ці відгуки, обчислити середню оцінку автомобіля, клієнта та власника на основі цих відгуків. Відгуки створюються тільки після того, як було повністю завершено прокат. Також підтримується можливість комунікації між власниками і орендарями, що за необхідності дозволяє запитувати обом сторонам те, що їх цікавить або потребує додаткової перевірки. Окрім реалізації програмної системи прийнято рішення з її проектування, які описано та обґрунтовано.

Науковою новизною роботи є метод прогнозування попиту на прокат автомобілів, який ґрунтуючись на застосуванні дерев рішень з навчанням на основі алгоритму LightGBM та параметрах автомобіля і прокату, дозволяє визначити кількість завершених резервувань прокатів автомобілів протягом наступного місяця.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Lobell, K. O. The Airbnb of Car Renting [Electronic resource] : How to Legally Rent Your Car to a Stranger – or from One / K. O. Lobell. – Access mode : <https://www.legalzoom.com/articles/the-airbnb-of-car-renting-how-to-legally-rent-your-car-to-a-stranger-from-one>.
2. The global car rental market is projected to grow from \$82.54 billion in 2022 to \$137.25 billion by 2029, at a CAGR of 7.5% in forecast period 2022-2029 [Electronic resource]. – Access mode : <https://www.fortunebusinessinsights.com/car-rental-market-105117>.
3. \$126+ Billion Global Car Rental Market is Expected to Grow at a CAGR of over 4.3% During 2022-2028. Vantage Market Research [Electronic resource]. – Access mode : <https://www.globenewswire.com/en/news-release/2022/08/02/2490510/0/en/126-Billion-Global-Car-Rental-Market-is-Expected-to-Grow-at-a-CAGR-of-over-4-3-During-2022-2028-Vantage-Market-Research.html>.
4. Car Rental Market – Growth, Trends, COVID-19 Impact, and Forecasts [Electronic resource]. – Access mode : <https://www.mordorintelligence.com/industry-reports/car-rental-market>.
5. Helling, B. Peer-to-Peer Car Rental [Electronic resource] : What Are Your Options? / B. Helling. – Access mode : <https://www.ridester.com/peer-to-peer-car-rental/>.
6. French, S. 5 Smart Rental Car Alternatives You Might Not Know About [Electronic resource] / S. French. – Access mode : <https://www.nerdwallet.com/article/travel/rental-car-alternatives>.
7. Turo Car Sharing Marketplace. Rent the Perfect Car [Electronic resource]. – Access mode : <https://turo.com/>.
8. MyCarYourRental [Electronic resource]. – Access mode : <https://www.mycaryourrental.com/>.
9. Ankarstad, N. How to use LightGBM and boosted decision trees to

forecast sales [Electronic resource] / N. Ankarstad. – Access mode : <https://towardsdatascience.com/how-to-use-lightgbm-and-boosted-decision-trees-forecast-sales-cf65ce8ab645>.

10. Modelling on Car-Sharing Serial Prediction Based on Machine Learning and Deep Learning [Electronic resource] / Nihad Brahimi, Huaping Zhang, Lin Dai and Jianzi Zhang // The Complexity of Mobility and Human Dynamics in Cities. – 2022. – Volume 2022. – Access mode : <https://www.hindawi.com/journals/complexity/2022/8843000/>.

11. Understanding LightGBM Parameters (and How to Tune Them) [Electronic resource]. – Access mode : <https://neptune.ai/blog/lightgbm-parameters-guide>.

12. LightGBM [Electronic resource] : A Highly Efficient Gradient Boosting Decision Tree / Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, Tie-Yan Liu // 31st Conference on Neural Information Processing Systems (NIPS 2017). – Long Beach, CA : 2017. – Access mode : <https://papers.nips.cc/paper/2017/hash/6449f44a102fde848669bdd9eb6b76fa-Abstract.html>.

13. Mondal, A. Complete guide on how to Use LightGBM in Python [Electronic resource] / A. Mondal. – Access mode : <https://www.analyticsvidhya.com/blog/2021/08/complete-guide-on-how-to-use-lightgbm-in-python/>.

14. Plot Listing Free Website Template [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page286/plot-listing>.

15. Using PayPal Standard IPN [Electronic resource]. – Access mode : <https://django-paypal.readthedocs.io/en/stable/standard/ipn.html>.

16. NVP and SOAP API reference [Electronic resource]. – Access mode : <https://developer.paypal.com/api/nvp-soap/>.

17. IPN and PDT variables [Electronic resource]. – Access mode : <https://developer.paypal.com/api/nvp-soap/ipn/IPNandPDTVariables/>.

18. Cornell Car Rental Dataset [Electronic resource]. – Access mode :

<https://www.kaggle.com/datasets/kushleshkumar/cornell-car-rental-dataset?select=CarRentalData.csv>.

ДОДАТОК А
Технічне завдання

Вступ

Бізнесом, який достатньо активно розвивається в інтернеті, останнім часом є бізнес, пов'язаний з орендою. Одним з рушіїв цього розвитку є подорожі, які можуть здійснюватися як в форматі туристичних, так і в форматі робочих поїздок. У таких випадках людина потребує знайти житло для проживання, що призвело до розвитку засобів оренди житла безпосередньо від господарів. Подібною сферою є прокат автомобілів, які можуть надаватися не централізовано компанією, а їх власниками. Перспективною є підтримка взаємодії господарів з клієнтами через створення такого програмного забезпечення та забезпечення необхідних засобів для господарів щодо прогнозування майбутньої активності стосовно їх автомобілів.

А.1 Підстави для розробки

Розробка кваліфікаційної роботи магістра виконується на основі завдання за темою «Дослідження та програмна реалізація методів підтримки прокату автомобілів», затвердженою наказом № 363 від 4 листопада 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма повинна надавати можливість кожному конкретному власнику автомобіля отримати доступ до ринку прокату автомобілів, на якому він може тимчасово віддати власний автомобіль іншим користувачам у прокат за плату, яку він вважає адекватною для даної послуги. Користувачу, який бажає взяти автомобіль на прокат, програма повинна надавати можливість знайти такий варіант автомобіля, який відповідає його вимогам, і зарезервувати його для подальшого прокату.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Функціональні можливості, які мають підтримуватися програмою, включають наступні:

- реєстрація і авторизація власників автомобілів і клієнтів;
- виставлення власного автомобіля на прокат;
- змінення параметрів прокату власного автомобіля;
- пошук автомобіля для прокату за вимогами клієнта;
- встановлення часових інтервалів, коли автомобіль доступний для прокату;
- встановлення варіантів вартості прокату на період за тривалістю;
- встановлення власником локацій для видачі автомобіля в прокат;
- встановлення доступних відстаней на час прокату;
- резервування автомобіля для прокату;
- формулювання відгуку про прокат автомобіля клієнтом;
- реагування власником автомобіля на відгук клієнта;
- формулювання відгуку про клієнта власником автомобіля;
- визначення усередненої рейтингової оцінки клієнта, власника автомобіля та автомобіля;
- відображення прокатного оголошення автомобіля, що включає сценарій обчислення вартості прокату;
- збереження автомобіля до переліку улюблених;
- обмін повідомленнями між власником автомобіля і клієнтом;
- відображення найбільш популярних автомобілів у заданому місті;
- відображення автомобілів з найвищим рейтингом у заданому місті;
- відображення резервувань клієнта, його майбутніх та минулих прокатів;
- відображення переліку улюблених автомобілів;
- скасування власного резервування автомобіля;

- скасування резервування за власним автомобілем власником;
- відображення запланованих і минулих прокатів автомобіля власника;
- відображення автомобілів власника;
- відображення відгуків клієнтів про власні автомобілі;
- відображення профіля з відгуками про клієнта;
- відображення профіля з відгуками про власника автомобіля;
- подання запиту на блокування облікового запису клієнта;
- підтвердження або скасування запиту на блокування облікового запису клієнта;
- відображення статистики прокату автомобіля за минулі періоди;
- прогнозування кількості прокатів автомобіля на місяць.

A.3.2 Вимоги до складу та параметрів технічних засобів

Сервер потребує обчислювальну потужність не меншу ніж процесор, який має не менше ніж 4 ядра, частоту від 2 ГГц, оперативну пам'ять, що має бути не меншою від 4 Гб, жорсткий диск, де має бути доступно не менше 2 Гб.

A.3.3 Умови експлуатації

Для експлуатації програмного забезпечення необхідні вміння керування вебсервером та сервером БД. У ролі вебсервера може застосовуватися тільки вебсервер, сумісний з системою, яка буде контролювати вебзастосунок і яка буде обрана на етапі проєктування. Такі самі вимоги мають враховуватися під час вибору системи управління БД.

A.3.4 Вимоги до програмної документації

Основним представленням результатів виконання роботи є пояснювальна записка. Вона повинна у своєму складі містити опис основних

рішень щодо реалізації компонентів системи у вигляді розділу 4, прийняті під час проєктування рішення у вигляді розділу 3, дане технічне завдання і текст програми з відповідним фрагментом у вигляді додатків А і Б.

А.4 Порядок контролю та приймання

Порядок контролю та приймання роботи визначається календарним планом, який входить до складу завдання на дипломну кваліфікаційну роботу магістра.

ДОДАТОК Б
Текст програми

Б.1 Текст файла baseviews.py

```

import datetime
from carentsoft.forms import VehicleBookForm, VehicleSearchForm,
ClientProfileForm
from carentsoft.models import Client, Propreview, Clientreview, Proprietor,
Vehicle, Vehiclepickup, Distance, Rentalready, Price, Vehiclemodel, City,
Reviewanswer, Vehiclebooking
from django.views.generic import ListView, DetailView
from django.shortcuts import render, get_object_or_404

class ClientProfileView (DetailView):
    model = Client
    template_name = 'carentsoft/client_profile.html'

    def get(self, request, pk):
        client_user = get_object_or_404(Client, id = pk)
        client = self.request.user
        client_profile = Client.objects.get(user = client)
        client_user.password = None
        if client_profile and client_profile != client_user:
            client_user.phonenumber = None
        birthday = client_user.birthday
        current = datetime.datetime.now()
        client_user.years = current.year - birthday.year
        date_to_compare = datetime.datetime(birthday.year, current.month,
current.day, current.hour, current.minute)
        if date_to_compare < birthday:
            client_user.years = client_user.years - 1
        client_reviews = Propreview.objects.filter(client = client_user)
        client_reviews = client_reviews.order_by('-revdate')
        client_profile_context = {}
        client_profile_context['client_profile'] = client_user
        client_profile_context['reviews'] = client_reviews
        return render(request, self.template_name, client_profile_context)

class ProprietorProfileView (DetailView):
    model = Proprietor
    template_name = 'carentsoft/proprietor_profile.html'

```

```

def get(self, request, pk):
    proprietor_user = get_object_or_404(Proprietor, id = pk)
    proprietor_user.password = None
    proprietor_user.phonenumber = None
    proprietor_reviews = Clientreview.objects.filter(vehicle_proprietor =
proprietor_user)
    proprietor_vehicles = Vehicle.objects.filter(proprietor =
proprietor_user)[0:3]
    proprietor_vehicles = proprietor_vehicles.order_by('-rating')
    proprietor_profile_context = {}
    proprietor_profile_context['proprietor_profile'] = proprietor_user
    proprietor_profile_context['reviews'] = proprietor_reviews
    proprietor_profile_context['vehicles'] = proprietor_vehicles
    return render(request, self.template_name, proprietor_profile_context)

```

```

class VehicleDetailView (DetailView):

```

```

    model = Vehicle

```

```

    template_name = 'carentsoft/vehicle.html'

```

```

def get(self, request, pk):

```

```

    vehicle_full = get_object_or_404(Vehicle, id = pk)

```

```

    pickup_locations = Vehiclepickup.objects.filter(vehicle = vehicle_full)

```

```

    distances = Distance.objects.filter(vehicle = vehicle_full)

```

```

    current = datetime.datetime.now()

```

```

    ready = Rentalready.objects.filter(vehicle = vehicle_full, periodend__gte =
current)

```

```

    ready = ready.order_by('periodend')

```

```

    client_reviews = Clientreview.objects.filter(vehicle = vehicle_full)

```

```

    client_reviews = client_reviews.order_by('revdate')[0:5]

```

```

    delivlocation = int(self.request.GET.get('delivlocation', '-1'))

```

```

    pickup_location = get_object_or_404(Vehiclepickup, id = delivlocation)

```

```

    deliveryprice = pickup_location.price

```

```

    out = int(self.request.GET.get('outlocation', '-1'))

```

```

    out_location = get_object_or_404(Vehiclepickup, id = out)

```

```

    checkoutprice = out_location.price

```

```

    bookstarttime = str(self.request.GET.get('bookstarttime', ''))

```

```

    bookstartdatetime = datetime.strptime(bookstarttime, '%d.%m.%y
%H:%M')

```

```

    bookendtime = str(self.request.GET.get('bookendtime', ''))

```

```

    bookenddatetime = datetime.strptime(bookendtime, '%d.%m.%y %H:%M')

```

```

    time_interval = bookenddatetime - bookstarttime

```

```

    days = time_interval.days

```

```

time_interval = time_interval.total_seconds()
hours = time_interval / 3600
if time_interval % 3600 != 0:
    hours = hours + 1
prices = Price.objects.filter(vehicle = vehicle_full, periodstart__lte =
bookstartdatetime, periodend__gte = bookenddatetime, maxduration__gte = days)
prices = prices.order_by('maxduration')
minprice = prices[0].price
minduration = prices[0].minduration
duration = hours / minduration
if hours % minduration != 0:
    duration = duration + 1
rental_price = minprice * duration
rental_price = rental_price + deliveryprice + checkoutprice
booked = Vehiclebooking.objects.filter(vehicle = vehicle_full,
checkindate__gte = current, state_in = [0, 1, 2, 7])
booked = booked.order_by('checkindate')
for b in booked:
    b.respondtime = None
    b.price = None
    b.paid = None
    b.deliveryprice = None
    b.deliverylocation = None
    b.checkoutprice = None
    b.checkoutlocation = None
    b.cancellationpolicy = None
    b.instruction = None
    b.payment_email = None
proprietor_profile_context = {}
vehicle_detail_context['booked'] = booked
vehicle_detail_context['vehicle'] = vehicle_full
vehicle_detail_context['locations'] = pickup_locations
vehicle_detail_context['distances'] = distances
vehicle_detail_context['ready'] = ready
vehicle_detail_context['client_reviews'] = client_reviews
vehicle_detail_context['rental_price'] = rental_price
vehicle_detail_context['deliveryprice'] = deliveryprice
vehicle_detail_context['checkoutprice'] = checkoutprice_price
return render(request, self.template_name, vehicle_detail_context)

```

```

def get_context_data(self, **kwargs):
    vehicle_search_form = super(VehicleDetailView,
self).get_context_data(**kwargs)
    vehicle_search_form ['form_settings'] = VehicleBookForm(initial={
        'delivlocation': self.request.GET.get('delivlocation', '-1'),
        'outlocation': self.request.GET.get('outlocation', ""),
        'bookstarttime': self.request.GET.get('bookstarttime', ""),
        'bookendtime': self.request.GET.get('bookendtime', "")
    })
    return vehicle_search_form

class ProprietorVehiclesView (ListView):
    model = Vehicle
    template_name = 'carentsoft/proprietor_vehicles.html'

    def get_queryset(self, *args, **kwargs):
        proprietor = int(self.request.GET.get('prop', '-1'))
        proprietor_vehicles = Vehicle.objects.filter(proprietor = proprietor_user)
        return proprietor_vehicles.order_by('-id')

class FullVehicleReviews (ListView):
    model = Clientreview
    template_name = 'carentsoft/vehicle_reviews.html'
    def get_queryset(self, *args, **kwargs):
        vehicle = int(self.request.GET.get('vehicle', '-1'))
        vehicle = get_object_or_404(Vehiclemodel, id = vehicle)
        reviews = Clientreview.objects.filter(vehicle = vehicle)
        reviews = reviews.order_by('-revdate')
        for rev in reviews:
            answers = Reviewanswer.objects.filter(review = rev)
            rev.answer = answers
        return reviews

class CityBestVehiclesView (ListView):
    model = Vehicle
    template_name = 'carentsoft/best_vehicle_rating.html'

    def get_queryset(self, *args, **kwargs):
        city_id = int(self.request.GET.get('city', '-1'))
        vehicles = Vehicle.objects.filter(city_id = city_id)
        best_vehicles = vehicles.order_by('-rating')[0:100]
        return best_vehicles

```

```

class CityPopularVehiclesView (ListView):
    model = Vehicle
    template_name = 'carentsoft/popular_vehicle_rating.html'

    def get_queryset(self, *args, **kwargs):
        city_id = int(self.request.GET.get('city', '-1'))
        vehicles = Vehicle.objects.filter(city_id = city_id)
        popular_vehicles = vehicles.order_by('-tripstotal')[0:100]
        return popular_vehicles

class VehicleFindView(ListView):
    model = Vehicle
    template_name = 'carentsoft/find_vehicles.html'

    def get_queryset(self, *args, **kwargs):
        vehicles = super().get_queryset(*args, **kwargs)
        vmodel = int(self.request.GET.get('vmodel', '0'))
        vmodel = get_object_or_404(Vehiclemodel, id = vmodel)
        vehicles = vehicles.filter(vehiclemodel = vmodel)
        brand = int(self.request.GET.get('brand', '0'))
        brand = get_object_or_404(Manufacturer, id = brand)
        vehicles = vehicles.filter(vehiclemodel_manufacturer = brand)
        vehtypes = str(self.request.GET.get('vehtypes', ''))
        if vehtypes:
            vehtypes_ids = vehtypes.split(',')
            for vt in vehtypes_ids:
                vt = int(vt)
                vehicles = vehicles.filter(vehtype__in = vehtypes_ids)
        minrating = int(self.request.GET.get('minrating', '0'))
        if minrating:
            vehicles = vehicles.filter(rating__gte = minrating)
        maxrating = int(self.request.GET.get('maxrating', '0'))
        if maxrating:
            vehicles = vehicles.filter(rating__lte = maxrating)
        city = int(self.request.GET.get('city', '-1'))
        city = get_object_or_404(City, id = city)
        if city:
            vehicles = vehicles.filter(city = city)
        minprodyear = int(self.request.GET.get('minprodyear', '0'))
        if minprodyear:
            vehicles = vehicles.filter(producedyear__gte = minprodyear)
        maxprodyear = int(self.request.GET.get('maxprodyear', '0'))

```

```

if maxprodyear:
    vehicles = vehicles.filter(producedyear__lte = maxprodyear)
minengine = int(self.request.GET.get('minengine', '0'))
if minengine:
    vehicles = vehicles.filter(enginevolume__gte = minengine)
maxengine = int(self.request.GET.get('maxengine', '0'))
if maxengine:
    vehicles = vehicles.filter(enginevolume__lte = maxengine)
allwheeled = int(self.request.GET.get('allwheeled', '0'))
if allwheeled:
    vehicles = vehicles.filter(allwheeled = allwheeled)
fueltypes = str(self.request.GET.get('fueltypes', ''))
if fueltypes:
    fueltypes_ids = vehtypes.split(',')
    for ft in fueltypes_ids:
        ft = int(ft)
        vehicles = vehicles.filter(fueltype__in = fueltypes_ids)
gpssystem = int(self.request.GET.get('gpssystem', '0'))
if gpssystem == 1:
    vehicles = vehicles.filter(gpssystem = True)
elif gpssystem == 2:
    vehicles = vehicles.filter(gpssystem = False)
minpassengers = int(self.request.GET.get('minpassengers', '0'))
if minpassengers:
    vehicles = vehicles.filter(passengers__gte = minpassengers)
maxpassengers = int(self.request.GET.get('maxpassengers', '0'))
if maxpassengers:
    vehicles = vehicles.filter(passengers__lte = maxpassengers)
mindoors = int(self.request.GET.get('mindoors', '0'))
if mindoors:
    vehicles = vehicles.filter(doors__gte = mindoors)
maxdoors = int(self.request.GET.get('maxdoors', '0'))
if maxdoors:
    vehicles = vehicles.filter(doors__lte = maxdoors)
airsystem = int(self.request.GET.get('airsystem', '0'))
if airsystem:
    vehicles = vehicles.filter(airsystem = airsystem)
childseat = int(self.request.GET.get('childseat', '0'))
if childseat:
    vehicles = vehicles.filter(childseat = True)
sunroof = int(self.request.GET.get('sunroof', '0'))
if sunroof:

```

```

    vehicles = vehicles.filter(sunroof = True)
usbinput = int(self.request.GET.get('usbinput', '0'))
if usbinput:
    vehicles = vehicles.filter(usbinput = True)
usbcharger = int(self.request.GET.get('usbcharger', '0'))
if usbcharger:
    vehicles = vehicles.filter(usbcharger = True)
tiresseason = int(self.request.GET.get('tiresseason', '0'))
if tiresseason:
    vehicles = vehicles.filter(tiresseason = tiresseason)
keyless = int(self.request.GET.get('keyless', '0'))
if keyless:
    vehicles = vehicles.filter(keyless = True)
blindspot = int(self.request.GET.get('blindspot', '0'))
if blindspot:
    vehicles = vehicles.filter(blindspotwarning = True)
backupcamera = int(self.request.GET.get('backupcamera', '0'))
if backupcamera:
    vehicles = vehicles.filter(backupcamera = True)
musicsystem = int(self.request.GET.get('musicsystem', '0'))
if musicsystem:
    vehicles = vehicles.filter(musicsystem= musicsystem)
longterm = int(self.request.GET.get('longterm', '0'))
if longterm:
    vehicles = vehicles.filter(longterm = longterm)
heatedseats = int(self.request.GET.get('heatedseats', '0'))
if heatedseats:
    vehicles = vehicles.filter(heatedseats = True)
bluetooth = int(self.request.GET.get('bluetooth', '0'))
if bluetooth:
    vehicles = vehicles.filter(bluetooth = True)
autocontrol = int(self.request.GET.get('autocontrol', '0'))
if autocontrol:
    vehicles = vehicles.filter(autocontrol = autocontrol)
petfriendly = int(self.request.GET.get('petfriendly', '0'))
if petfriendly:
    vehicles = vehicles.filter(petfriendly = True)
smoking = int(self.request.GET.get('smoking', '0'))
if smoking:
    vehicles = vehicles.filter(smoking = True)
bodycolor = int(self.request.GET.get('bodycolor', '-1'))
if bodycolor:

```

```

        vehicles = vehicles.filter(bodycolor = bodycolor)
    mintrips = int(self.request.GET.get('mintrips', '0'))
    if mintrips:
        vehicles = vehicles.filter(tripstotal__gte = mintrips)
    maxtrips = int(self.request.GET.get('maxtrips', '0'))
    if maxtrips:
        vehicles = vehicles.filter(tripstotal__lte = maxtrips)
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)
    current = datetime.datetime.now()
    birthday = client_profile.birthday
    client_profile.years = current.year - birthday.year
    date_to_compare = datetime.datetime(birthday.year, current.month,
current.day, current.hour, current.minute)
    if date_to_compare < birthday:
        client_profile.years = client_profile.years - 1
    age = client_profile.years
    if age:
        vehicles = vehicles.filter(minimumage__lte = age)
    transmission = int(self.request.GET.get('transmission', '0'))
    if transmission:
        vehicles = vehicles.filter(transmission = transmission)
    bookstarttime = str(self.request.GET.get('bookstarttime', ''))
    bookstartdatetime = datetime.strptime(bookstarttime, '%d.%m.%y
%H:%M')
    bookendtime = str(self.request.GET.get('bookendtime', ''))
    bookenddatetime = datetime.strptime(bookendtime, '%d.%m.%y %H:%M')
    time_interval = bookenddatetime - bookstarttime
    days = time_interval.days
    time_interval = time_interval.total_seconds()
    hours = time_interval / 3600
    if time_interval % 3600 != 0:
        hours = hours + 1
    vehicles = vehicles.order_by('-rating')
    maxprice = int(self.request.GET.get('maxprice', '0'))
    minprice = int(self.request.GET.get('minprice', '0'))
    vehicles_with_price = []
    for vehicle in vehicles:
        prices = Price.filter(vehicle = vehicle, periodstart__lte =
bookstartdatetime, periodend__gte = bookenddatetime, maxduration__gte = days)
        prices = prices.order_by('maxduration')
        mprice = prices[0].price

```

```

minduration = prices[0].minduration
duration = hours / minduration
if hours % minduration != 0:
    duration = duration + 1
rental_price = mprice * duration
booked = Vehiclebooking.objects.filter(vehicle = vehicle,
checkindate__lte = bookstartdatetime, checkoutdate__gte = bookenddatetime,
state__in = [0, 1, 2, 7])
    if rental_price >= minprice and rental_price <= maxprice and not booked:
        vehicle.price = rental_price
        vehicles_with_price.append(vehicle)
return vehicles_with_price

def get_context_data(self, **kwargs):
    vehicle_search_form = super(VehicleFindView,
self).get_context_data(**kwargs)
    vehicle_search_form['form_settings'] = VehicleSearchForm(initial={
        'vmodel': self.request.GET.get('vmodel', '0'),
        'brand': self.request.GET.get('brand', '0'),
        'vehetypes': self.request.GET.get('vehetypes', '0'),
        'minrating': self.request.GET.get('minrating', '0'),
        'maxrating': self.request.GET.get('maxrating', '0'),
        'bookstarttime': self.request.GET.get('bookstarttime', ''),
        'bookendtime': self.request.GET.get('bookendtime', '0'),
        'minprice': self.request.GET.get('minprice', '0'),
        'maxprice': self.request.GET.get('maxprice', '0'),
        'city': self.request.GET.get('city', '-1'),
        'minprodyear': self.request.GET.get('minprodyear', '0'),
        'maxprodyear': self.request.GET.get('maxprodyear', '0'),
        'minengine': self.request.GET.get('minengine', '0'),
        'maxengine': self.request.GET.get('maxengine', '0'),
        'allwheeled': self.request.GET.get('allwheeled', '0'),
        'fueltypes': self.request.GET.get('fueltypes', ''),
        'gpssystem': self.request.GET.get('gpssystem', '0'),
        'minpassengers': self.request.GET.get('minpassengers', '0'),
        'maxpassengers': self.request.GET.get('maxpassengers', '0'),
        'mindoors': self.request.GET.get('mindoors', '0'),
        'maxdoors': self.request.GET.get('maxdoors', '0'),
        'airsystem': self.request.GET.get('airsystem', '0'),
        'childseat': self.request.GET.get('childseat', '0'),
        'sunroof': self.request.GET.get('sunroof', '0'),
        'usbinput': self.request.GET.get('usbinput', '0'),

```

```

        'usbcharger': self.request.GET.get('usbcharger', '0'),
        'tiresseason': self.request.GET.get('tiresseason', '0'),
        'keyless': self.request.GET.get('keyless', '0'),
        'blindspot': self.request.GET.get('blindspot', '0'),
        'backupcamera': self.request.GET.get('backupcamera', '0'),
        'musicsystem': self.request.GET.get('musicsystem', '0'),
        'longterm': self.request.GET.get('longterm', '0'),
        'heatedseats': self.request.GET.get('heatedseats', '0'),
        'bluetooth': self.request.GET.get('bluetooth', '0'),
        'autocontrol': self.request.GET.get('autocontrol', '0'),
        'petfriendly': self.request.GET.get('petfriendly', '0'),
        'smoking': self.request.GET.get('smoking', '0'),
        'bodycolor': self.request.GET.get('bodycolor', '-1'),
        'mintrips': self.request.GET.get('mintrips', '0'),
        'maxtrips': self.request.GET.get('maxtrips', '0'),
        'transmission': self.request.GET.get('transmission', '0')
    })
    return vehicle_search_form

```

Б.2 Текст файла clientviews.py

```

from carentsoft.forms import ClientreviewForm, VehicleBookingForm
from paypal.standard.forms import PayPalPaymentsForm
from carentsoft.models import Client, Clientreview, Vehicle, Savedvehicle,
Vehiclebooking, Request, Requestanswer, Proprietor, Vehiclepickup, Price
from paypal.standard.models import ST_PP_COMPLETED
from django.views.generic import ListView, TemplateView, DetailView
from django.urls import reverse, reverse_lazy
from django.shortcuts import render, redirect, get_object_or_404
from django.views.decorators.csrf import csrf_exempt
from django.utils.decorators import method_decorator
from paypal.standard.ipn.signals import valid_ipn_received
from django.db.utils import IntegrityError
from django.contrib.auth.mixins import LoginRequiredMixin
from django.http import HttpResponse

class ClientProfileEditView(LoginRequiredMixin, View):
    template_name = 'carentsoft/client_edit.html'

    def get(self, request, pk = None):
        auser = self.request.user

```

```

    client_profile = get_object_or_404(Client, user = auser)
    cpf = ClientProfileForm (instance = client_profile)
    crf_context = {}
    crf_context['form'] = crf
    return render(request, self.template_name, cpf_context)

def post(self, request, pk = None):
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)
    cpf = ClientProfileForm (request.POST, request.FILES or None, instance =
client_profile)
    if not cpf.is_valid():
        crf_context = {}
        crf_context['form'] = crf
        return render(request, self.template_name, cpf_context)
    cl = cpf.save()
    cl.save()
    return redirect(reverse_lazy('carentsoft:clientprofile'))

class ClientreviewPublishView(LoginRequiredMixin, View):
    template_name = 'carentsoft/add_client_review.html'

    def get(self, request, pk):
        crf = ClientreviewForm()
        vehicle = get_object_or_404(Vehicle, id = pk)
        crf_context = {}
        crf_context['form'] = crf
        crf_context['vehicle'] = vehicle
        return render(request, self.template_name, crf_context)

    def post(self, request, pk = None):
        crf = ClientreviewForm (request.POST, None)
        auser = self.request.user
        client_profile = get_object_or_404(Client, user = auser)
        if not crf.is_valid():
            vehicle = get_object_or_404(Vehicle, id = pk)
            crf_context = {}
            crf_context['form'] = crf
            crf_context['vehicle'] = vehicle
            return render(request, self.template_name, crf_context)
        clientreview = ClientreviewForm.save(commit = False)
        clientreview.client = client_profile

```

```

clientreview.save()
vehicle = get_object_or_404(Vehicle, id = clientreview.vehicle.id)
clreviews = Clientreview.objects.filter(vehicle = vehicle)
rating = 0
reviews_count = clreviews.count()
for clreview in clreviews:
    rating = rating + clreview.rating
rating = rating / reviews_count
vehicle.rating = rating
vehicle.reviewsnumber = reviews_count
vehicle.save()
proprietor = vehicle.proprietor
clreviews = Clientreview.objects.filter(vehicle_proprietor = proprietor)
rating = 0
reviews_count = clreviews.count()
for clreview in clreviews:
    rating = rating + clreview.rating
rating = rating / reviews_count
proprietor.rating = rating
proprietor.reviewsnumber = reviews_count
proprietor.save()
return redirect(reverse('carentsoft:clientreviews', args=[client_profile.id]))

```

```

@method_decorator(csrf_exempt, name='dispatch')
class VehicleSaveView(LoginRequiredMixin, View):
    def post(self, request, pk):
        auser = self.request.user
        client_profile = get_object_or_404(Client, user = auser)
        veh = get_object_or_404(Vehicle, id = pk)
        save_veh = Savedvehicle(client = client_profile, vehicle = veh)
        try:
            save_veh.save()
        except IntegrityError as e:
            pass
        return HttpResponse()

```

```

@method_decorator(csrf_exempt, name='dispatch')
class VehicleUnsaveView(LoginRequiredMixin, View):
    def post(self, request, pk):
        auser = self.request.user
        client_profile = get_object_or_404(Client, user = auser)
        vehicle = get_object_or_404(Vehicle, id = pk)

```

```

    try:
        save_veh = Savedvehicle.objects.get(client = client_profile, vehicle =
veh).delete()
    except save_veh.DoesNotExist as err:
        pass
    return HttpResponse()

```

```

class VehiclesSavedView (ListView):
    model = Vehicle
    template_name = 'carentsoft/saved_vehicles.html'

```

```

def get_queryset(self, *args, **kwargs):
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)
    saves = Savedvehicle.objects.get(client = client_profile)
    vehicles = []
    for sv in saves:
        vehicles.append(sv.vehicle)
    return vehicles

```

```

class OldVehicleBookingsView(ListView):
    model = Vehiclebooking
    template_name = 'carentsoft/old_bookings.html'

```

```

def get_queryset(self, *args, **kwargs):
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)
    current = datetime.datetime.now()
    vehbooks = Vehiclebooking.objects.get(client = client_profile,
checkoutdate__lt = current)
    vehbooks = vehbooks.order_by('-checkoutdate')
    for vehbook in vehbooks:
        vehbook.payment_email = "
    return vehbooks

```

```

class FutureVehicleBookingsView(ListView):
    model = Vehiclebooking
    template_name = 'carentsoft/future_bookings.html'

```

```

def get_queryset(self, *args, **kwargs):
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)

```

```

        current = datetime.datetime.now()
        vehbooks = Vehiclebooking.objects.get(client = client_profile,
checkoutdate__gte = current, state__in = [0, 1, 2, 7])
        vehbooks = vehbooks.order_by('checkindate')
        for vehbook in vehbooks:
            vehbook.payment_email = ""
        return vehbooks

```

```

class FuturePayedVehicleBookingsView(ListView):
    model = Vehiclebooking
    template_name = 'carentsoft/future_payed_bookings.html'

    def get_queryset(self, *args, **kwargs):
        auser = self.request.user
        client_profile = get_object_or_404(Client, user = auser)
        current = datetime.datetime.now()
        vehbooks = Vehiclebooking.objects.get(client = client_profile,
checkoutdate__gte = current, state = 2)
        vehbooks = vehbooks.order_by('checkindate')
        for vehbook in vehbooks:
            vehbook.payment_email = ""
        return vehbooks

```

```

class CancelledVehicleBookingsView(ListView):
    model = Vehiclebooking
    template_name = 'carentsoft/cancelled_bookings.html'

    def get_queryset(self, *args, **kwargs):
        auser = self.request.user
        client_profile = get_object_or_404(Client, user = auser)
        current = datetime.datetime.now()
        vehbooks = Vehiclebooking.objects.get(client = client_profile,
checkoutdate__gte = current, state__in = [3, 4, 5, 6])
        vehbooks = vehbooks.order_by('-checkindate')
        for vehbook in vehbooks:
            vehbook.payment_email = ""
        return vehbooks

```

```

class RequestanswerCreateView(LoginRequiredMixin, View):

    def post(self, request, pk = None):
        auser = self.request.user

```

```

client_profile = Client.objects.get(user = auser)
if not pk:
    veh = int(request.POST['vehicle'])
    vehicle = get_object_or_404(Vehicle, id = veh)
    request = Request(vehicle = vehicle, client = client_profile)
    request.save()
else:
    request = get_object_or_404(Request, id = pk)
if not client_profile:
    proprietor = Proprietor.objects.get(user = auser)
    answer = Requestanswer(respond = request.POST['message'], request =
request, respondauthor = 0)
else:
    answer = Requestanswer(respond = request.POST['message'], request =
request, respondauthor = 1)
    answer.save()
    return redirect(reverse('carentsoft:reqanswers', args=[pk]))

```

```

class RequestAnswersView(ListView):

```

```

    model = Requestanswer

```

```

    template_name = 'carentsoft/request_answers.html'

```

```

    def get_queryset(self, *args, **kwargs):

```

```

        req = self.request.GET.get('req', '-1')

```

```

        auser = self.request.user

```

```

        client_profile = Client.objects.get(user = auser)

```

```

        proprietor = Proprietor.objects.get(user = auser)

```

```

        request = get_object_or_404(Request, id = req)

```

```

        answers = []

```

```

        if client_profile and request.client == client_profile:

```

```

            answers = Requestanswer.objects.filter(request = request)

```

```

            answers = answers.order_by('-requesttime')

```

```

        elif proprietor and request.vehicle.proprietor == proprietor:

```

```

            answers = Requestanswer.objects.filter(request = request)

```

```

            answers = answers.order_by('-requesttime')

```

```

        return answers

```

```

class RequestsListView(ListView):

```

```

    model = Request

```

```

    template_name = 'carentsoft/requests.html'

```

```

def get_queryset(self, *args, **kwargs):
    client_profile = Client.objects.get(user = auser)
    requests = Request.objects.filter(client = client_profile)
    requests_displayed = []
    reqanswers = []
    for req in requests:
        ras = Requestanswer.objects.filter(request = req)
        ras = ras.order_by('-requesttime')[0]
        reqanswers.append(ras)
    reqanswers = reqanswers.order_by('-requesttime')
    for req in reqanswers:
        requests_displayed.append(req.request)
    return requests_displayed

```

```

class RentalPaymentView (DetailView):

```

```

    model = VehicleBooking
    template_name = 'paypal_payment.html'

```

```

    def get(self, request, pk):

```

```

        vehiclebooking = get_object_or_404(Vehiclebooking, id = pk)
        paypal_values = {}
        paypal_values["business"] = settings.PAYPAL_RECEIVER_EMAIL
        paypal_values["amount"] = vehiclebooking.price
        paypal_values["currency_code "] = "USD"
        paypal_values["item_name"] = 'Car rental'
        paypal_values["invoice"] = vehiclebooking.id
        paypal_values["notify_url"] = vehiclebooking.id
        self.request.build_absolute_uri(reverse('carentsoft:paymentipn'))
        paypal_values["return_url"] =
        self.request.build_absolute_uri(reverse('carentsoft:paymentreturn'))
        paypal_values["cancel_return"] =
        self.request.build_absolute_uri(reverse('carentsoft:paymentcancel'))
        paypal_values["lc"] = 'EN'
        paypal_values["no_shipping"] = '1'
        pppf = PayPalPaymentsForm(initial = paypal_values)
        return render(request, self.template_name, pppf)

```

```

class RentalPaymentSuccessView(TemplateView):

```

```

    template_name = 'success_payment_result.html'

```

```

class RentalPaymentCancelView(TemplateView):

```

```

    template_name = 'cancelled_payment_result.html'

```

```

@receiver(valid_ipn_received)
def payment_result_processor(sender, **kwargs):
    result_for_payment = sender
    if result_for_payment.payment_status == ST_PP_COMPLETED:
        if result_for_payment.receiver_email !=
settings.PAYPAL_RECEIVER_EMAIL:
            return
            vehiclebooking_id = result_for_payment.invoice
            vehiclebooking = Vehiclebooking.objects.get(pk = vehiclebooking_id)
            if result_for_payment.mc_currency == 'USD'
                vehiclebooking.paid = vehiclebooking.paid +
result_for_payment.mc_gross
                if vehiclebooking.paid == vehiclebooking.price and vehiclebooking.state
== 0:
                    vehiclebooking.state = 2
                    elif vehiclebooking.paid != vehiclebooking.price and vehiclebooking.paid
> 0 and vehiclebooking.state == 0:
                        vehiclebooking.state = 1
                        elif vehiclebooking.paid == vehiclebooking.price and
vehiclebooking.state == 7:
                            vehiclebooking.state = 8
                            vehiclebooking.payment_email = result_for_payment.payer_email
                            vehiclebooking.save()
            else:
                return

class VehicleBookingCreateView(LoginRequiredMixin, View):
    template_name = 'carentsoft/make_booking.html'

    def get(self, request, pk):
        vbf = VehicleBookingForm ()
        vehicle = get_object_or_404(Vehicle, id = pk)
        vbf_context = {}
        vbf_context['form'] = vbf
        vbf_context['vehicle'] = vehicle
        locations = Vehiclepickup.objects.filter(vehicle = vehicle, state = 1)
        locations = locations.order_by('place')
        vbf_context['locations'] = locations
        delivlocation = int(self.request.GET.get('delivlocation', '-1'))
        pickup_location = get_object_or_404(Vehiclepickup, id = delivlocation)
        deliveryprice = pickup_location.price
        out = int(self.request.GET.get('outlocation', '-1'))

```

```

out_location = get_object_or_404(Vehiclepickup, id = out)
checkoutprice = out_location.price
bookstarttime = str(self.request.GET.get('bookstarttime', ''))
bookstartdatetime = datetime.strptime(bookstarttime, '%d.%m.%y
%H:%M')

bookendtime = str(self.request.GET.get('bookendtime', ''))
bookenddatetime = datetime.strptime(bookendtime, '%d.%m.%y %H:%M')
booked = Vehiclebooking.objects.filter(vehicle = vehicle, checkindate__lte
= bookstartdatetime, checkoutdate__gte = bookenddatetime, state__in = [0, 1, 2])
if not booked:
    return
time_interval = bookenddatetime - bookstarttime
days = time_interval.days
time_interval = time_interval.total_seconds()
hours = time_interval / 3600
if time_interval % 3600 != 0:
    hours = hours + 1
prices = Price.objects.filter(vehicle = vehicle_full, periodstart__lte =
bookstartdatetime, periodend__gte = bookenddatetime, maxduration__gte = days)
prices = prices.order_by('maxduration')
minprice = prices[0].price
minduration = prices[0].minduration
duration = hours / minduration
if hours % minduration != 0:
    duration = duration + 1
rental_price = minprice * duration
rental_price = rental_price + deliveryprice + checkoutprice
vbf_context['deliveryprice'] = deliveryprice
vbf_context['checkoutprice'] = checkoutprice
vbf_context['rental'] = rental_price
return render(request, self.template_name, vbf_context)

def post(self, request, pk):
    vbf = VehicleBookingForm(request.POST, None)
    auser = self.request.user
    client_profile = get_object_or_404(Client, user = auser)
    vehicle = get_object_or_404(Vehicle, id = pk)
    if not vbf.is_valid():
        delivlocation = int(self.request.GET.get('delivlocation', '-1'))
        out = int(self.request.GET.get('outlocation', '-1'))
        bookstarttime = str(self.request.GET.get('bookstarttime', ''))
        bookendtime = str(self.request.GET.get('bookendtime', ''))

```

```

        pickup_location = get_object_or_404(Vehiclepickup, id = delivlocation)
        out_location = get_object_or_404(Vehiclepickup, id = out)
        bookstartdatetime = datetime.strptime(bookstarttime, '%d.%m.%y
%H:%M')

        bookenddatetime = datetime.strptime(bookendtime, '%d.%m.%y
%H:%M')

    else:
        vehiclebooking = VehicleBookingForm.save(commit = False)
        pickup_location = vehiclebooking.deliverylocation
        out_location = vehiclebooking.checkoutlocation
        bookstartdatetime = vehiclebooking.checkindate
        bookenddatetime = vehiclebooking.checkoutdate
        booked = Vehiclebooking.objects.filter(vehicle = vehicle, checkindate__lte
= bookstartdatetime, checkoutdate__gte = bookenddatetime, state__in = [0, 1, 2])
        if booked:
            return
        deliveryprice = pickup_location.price
        checkoutprice = out_location.price
        time_interval = bookenddatetime - bookstarttime
        days = time_interval.days
        time_interval = time_interval.total_seconds()
        hours = time_interval / 3600
        if time_interval % 3600 != 0:
            hours = hours + 1
        prices = Price.objects.filter(vehicle = vehicle_full, periodstart__lte =
bookstartdatetime, periodend__gte = bookenddatetime, maxduration__gte = days)
        prices = prices.order_by('maxduration')
        minprice = prices[0].price
        minduration = prices[0].minduration
        duration = hours / minduration
        if hours % minduration != 0:
            duration = duration + 1
        rental_price = minprice * duration
        rental_price = rental_price + deliveryprice + checkoutprice
        if not vbf.is_valid():
            vbf_context = {}
            vbf_context['form'] = vbf
            vbf_context['vehicle'] = vehicle
            locations = Vehiclepickup.objects.filter(vehicle = vehicle, state = 1)
            locations = locations.order_by('place')
            vbf_context['locations'] = locations
            vbf_context['deliveryprice'] = deliveryprice

```

```

        vbf_context['checkoutprice'] = checkoutprice
        vbf_context['rental'] = rental_price
        return render(request, self.template_name, vbf_context)
    vehiclebooking.client = client_profile
    vehiclebooking.deliveryprice = deliveryprice
    vehiclebooking.checkoutprice = checkoutprice
    vehiclebooking.price = rental_price
    vehiclebooking.cancellationpolicy = vehicle.cancellationpolicy
    vehiclebooking.state = 0
    vehiclebooking.save()
    return return redirect(reverse('carentsoft:rentalpayment',
args=[vehiclebooking.id]))

```

```

class VehicleBookingClientCancelView(LoginRequiredMixin, View):

    def post(self, request, pk):
        vehiclebooking = get_object_or_404(Vehiclebooking, id = pk)
        auser = self.request.user
        client_profile = Client.objects.get(user = auser)
        if vehiclebooking.client == client_profile and vehiclebooking.state in [0, 1,
2]:
            vehiclebooking.state = 5
            vehiclebooking.save()
            return redirect(reverse('carentsoft:paymentreturn',
args=[vehiclebooking.id]))

```

```

class RentalPaymentReturnView(DetailView):
    model = VehicleBooking
    template_name = 'paypal_return_payment.html'

    def get(self, request, pk):
        vehiclebooking = get_object_or_404(Vehiclebooking, id = pk)
        paypal_values = {}
        paypal_values["business"] = vehiclebooking.payment_email
        if vehiclebooking.cancellationpolicy == 0:
            paypal_values["amount"] = vehiclebooking.price
        elif vehiclebooking.cancellationpolicy == 1:
            paypal_values["amount"] = vehiclebooking.price * 0.9
        elif vehiclebooking.cancellationpolicy == 2:
            paypal_values["amount"] = vehiclebooking.price * 0.5
        else:
            return

```

```

        paypal_values["currency_code "] = "USD"
        paypal_values["item_name"] = 'Car rental'
        paypal_values["invoice"] = vehiclebooking.id
        paypal_values["notify_url"] =
self.request.build_absolute_uri(reverse('carentsoft:paymentretipn'))
        paypal_values["return_url"] =
self.request.build_absolute_uri(reverse('carentsoft:paymentretreturn'))
        paypal_values["cancel_return"] =
self.request.build_absolute_uri(reverse('carentsoft:paymentretcancel'))
        paypal_values["lc"] = 'EN'
        paypal_values["no_shipping"] = '1'
        pppf = PayPalPaymentsForm(initial = paypal_values)
        return render(request, self.template_name, pppf)

    @receiver(valid_ipn_received)
    def payment_return_result_processor(sender, **kwargs):
        result_for_payment = sender
        if result_for_payment.payment_status == ST_PP_COMPLETED:
            payment_client = get_object_or_404(Client, payment_email =
result_for_payment.receiver_email)
            vehiclebooking = get_object_or_404(Vehiclebooking, client =
payment_client, id = result_for_payment.invoice)
            if result_for_payment.mc_currency == 'USD' and
result_for_payment.mc_gross > 0 and vehiclebooking.paid >=
result_for_payment.mc_gross:
                vehiclebooking.paid = vehiclebooking.paid -
result_for_payment.mc_gross
            if vehiclebooking.state == 3:
                vehiclebooking.state = 4
            elif vehiclebooking.state == 5:
                vehiclebooking.state = 6
            vehiclebooking.save()
        else:
            return

```

ДОДАТОК В
Слайди презентації



Рисунок В.1 – Титульний слайд



Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

3

- проаналізувати програмне забезпечення підтримки прокату автомобілів;
- виконати дослідження методів підтримки прокату автомобілів;
- виконати проектування програмного забезпечення підтримки прокату автомобілів;
- реалізувати програмне забезпечення підтримки прокату автомобілів.

Рисунок В.3 – Завдання роботи

Порівняння аналогів з програмою, що розроблялась в роботі

4

Критерій	Turo	MyCarYourRental	Розробка
Пошук автомобілів для прокату	+	+	+
Розширений пошук автомобілів	+	–	+
Вибір варіанту доставки автомобіля	+	–	+
Можливість внести власний автомобіль	+	+	+
Прогнозування кількості прокатів	–	–	+
Різні пакети дальності подорожей	+	–	+
Відображення найкращих автомобілів	+	–	+
Підтримка короткотривалого/ довготривалого прокату	+	+	+

Рисунок В.4 – Порівняння аналогів з програмою, що розроблялась в роботі



Рисунок В.5 – Метод прогнозування попиту на прокат автомобілів

Порівняння фреймворків

Показник	Symfony	Django
Мова програмування	PHP	Python
Надання засобів перевірки безпеки роботи	Нижча	Вища
Об'єктно-реляційне відображення	Doctrine	Django ORM
Простота роботи з моделями даних	Нижча	Вища
Підтримка спільноти	Активна	Активна
Підтримка бібліотек для роботи з інтелектуальними методами	Середня	Максимально висока на даний момент

Рисунок В.6 – Порівняння фреймворків



Рисунок В.7 – Діаграма варіантів використання програмної системи



Рисунок В.8 – Схема БД



Рисунок В.9 – Структура розробленої програмної системи

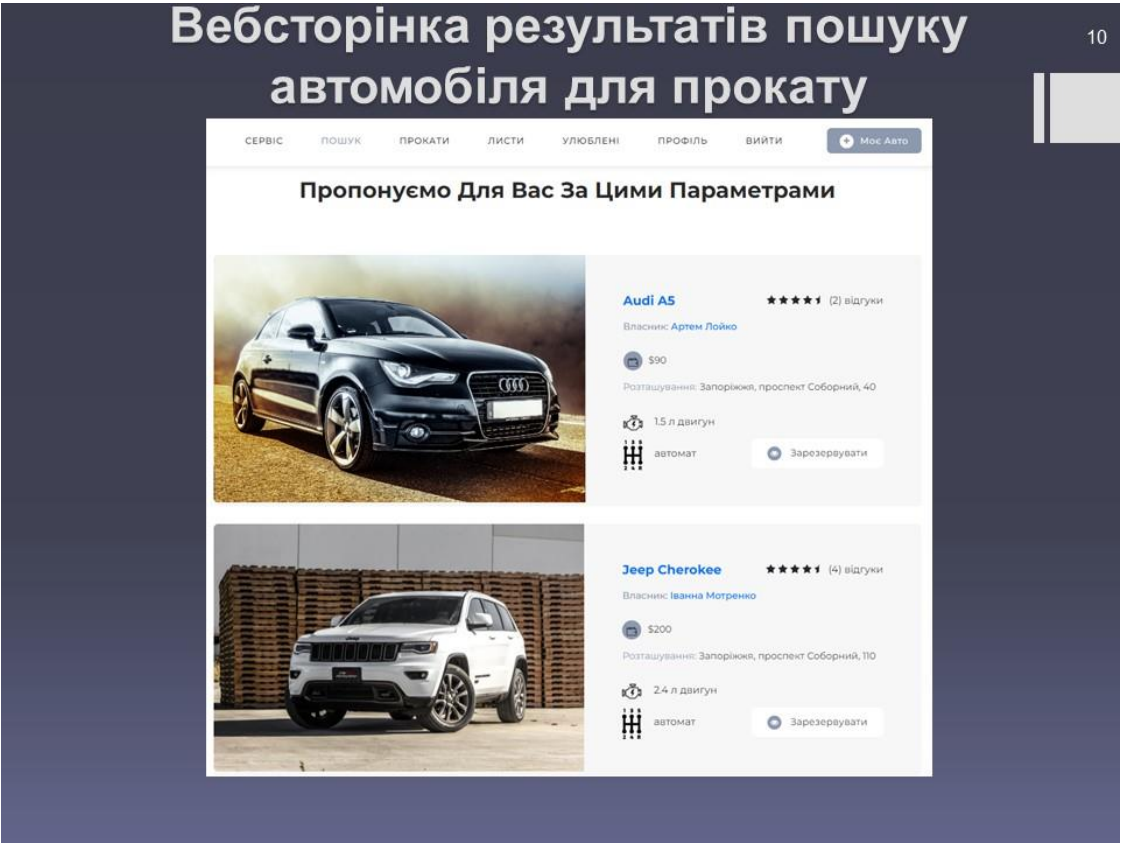


Рисунок В.10 – Вебсторінка результатів пошуку автомобіля для прокату

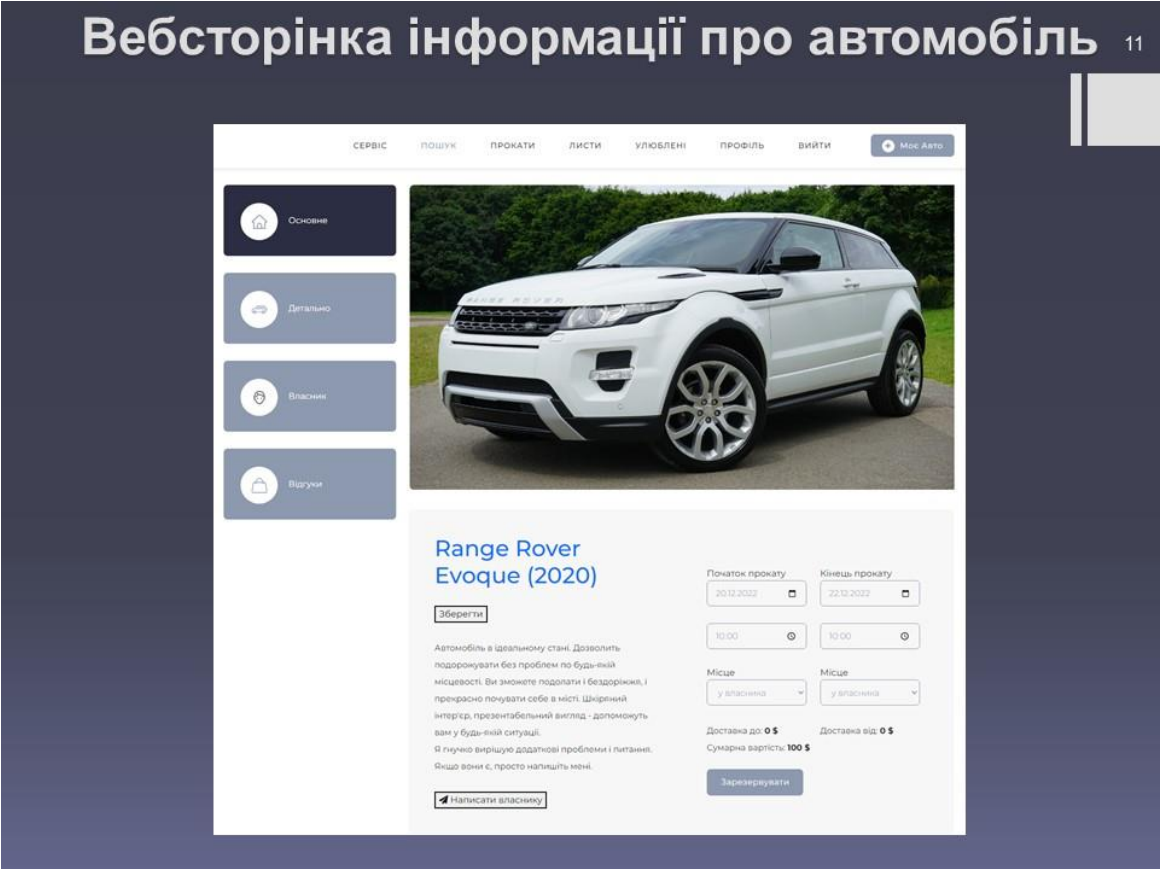


Рисунок В.11 – Вебсторінка інформації про автомобіль

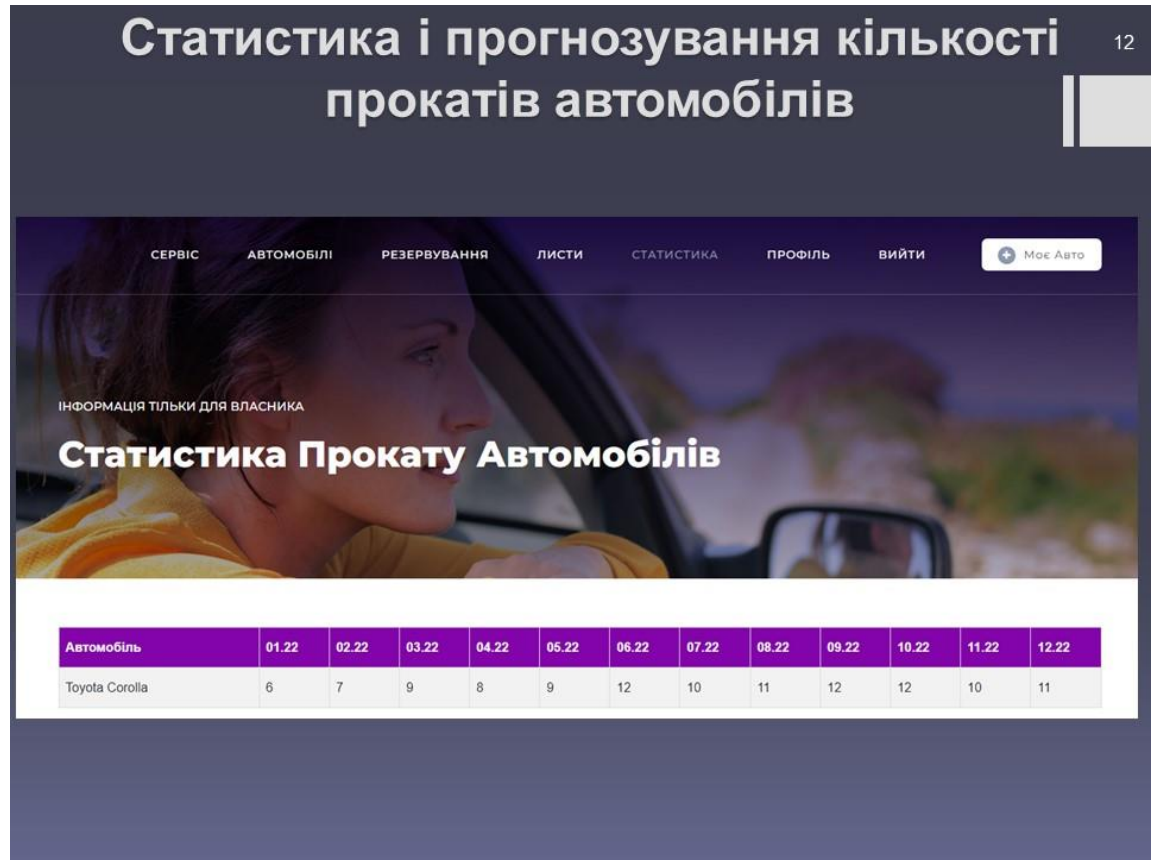


Рисунок В.12 – Статистика і прогнозування кількості прокатів автомобілів

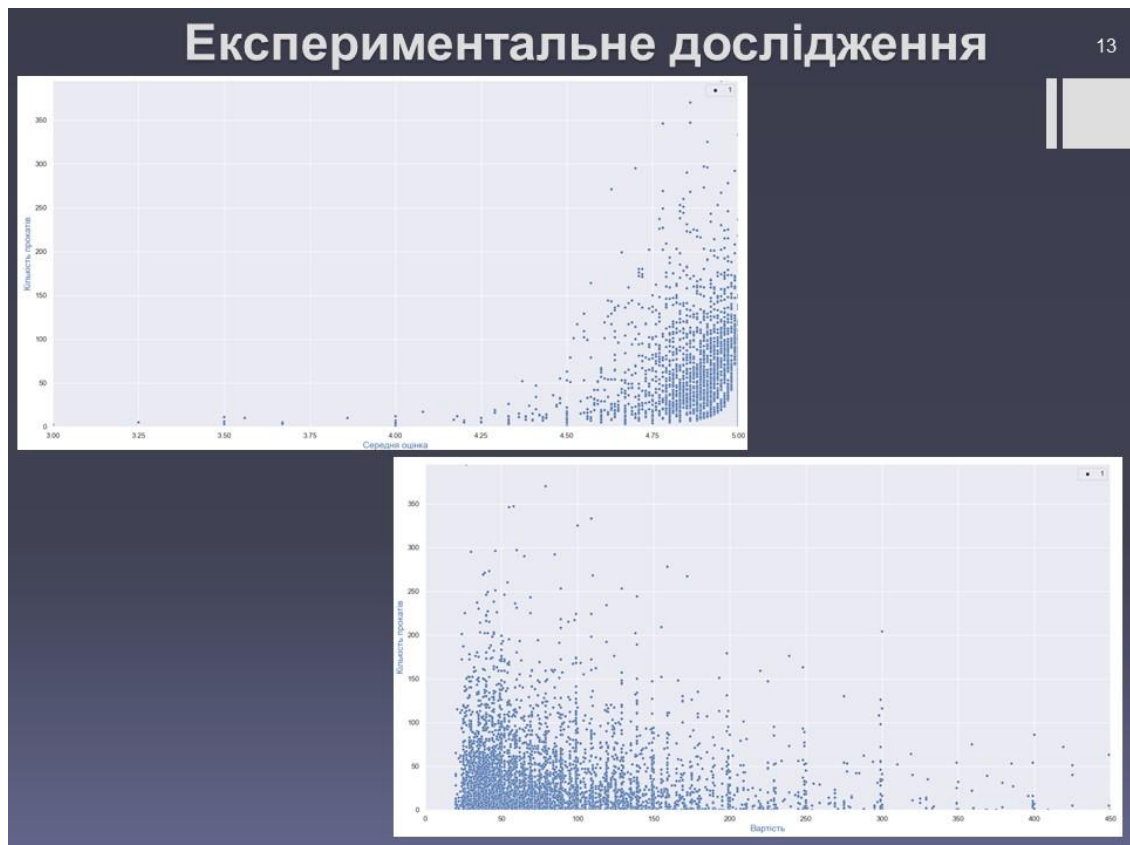


Рисунок В.13 – Експериментальне дослідження

Експериментальне дослідження

14

Метод	Коефіцієнт R^2
Метод опорних векторів	0,673
Метод прогнозування попиту на прокат автомобілів	0,819

Рисунок В.14 – Експериментальне дослідження

Висновки

15

Дипломна кваліфікаційна робота магістра направлена на розробку програмного забезпечення підтримки прокату автомобілів для допомоги власникам автомобілів під час виставлення їх на прокат. Така підтримка включає як підтримку виконання резервувань прокату, так і підтримку прийняття рішень власниками щодо параметрів прокату, які принесуть їм достатню користь.

Розроблене програмне забезпечення дозволяє як виставляти власній автомобілі на прокат, знаходити адекватні варіанти автомобілів для прокату на основі розширеного набору параметрів пошуку, так і виконувати резервування цих автомобілів. Резервування забезпечується не тільки через створення певного графіку прокату, але і за допомогою підтримки даного процесу через сплату за послуги з використанням платіжної системи PayPal.

Науковою новизною роботи є метод прогнозування попиту на прокат автомобілів, який ґрунтується на застосуванні дерев рішень з навчанням на основі алгоритму LightGBM та параметрах автомобіля і прокату, дозволяє визначити кількість завершених резервувань прокатів автомобілів протягом наступного місяця.

Рисунок В.15 – Висновки