

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Інститут інформатики та радіoeлектроніки,
Факультет комп'ютерних наук і технологій
(повне найменування інституту, факультету)

Кафедра програмних засобів
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістра

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
ПОШУКУ ВЖИВАНИХ АВТОМОБІЛІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS FOR
SEARCHING USED CARS

Виконав: студент(ка) 2 курсу, групи КНТ-210м
Спеціальності 122 Комп'ютерні науки
(код і найменування спеціальності)

Освітня програма (спеціалізація)
Системи штучного інтелекту

Тарасов В.С.
(прізвище та ініціали)

Керівник Степаненко О.О.
(прізвище та ініціали)

Рецензент Гофман Є.О.
(прізвище та ініціали)

2021

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

Інститут, факультет ІРЕ, ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)

Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2021 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Тарасова Владислава Сергійовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів пошуку вживаних автомобілів

Research and Software Implementation of Methods for Searching Used Cars

керівник проєкту (роботи) Степаненко Олександр Олексійович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від “5” листопада 2021 року №435

2. Строк подання студентом проєкту (роботи) 13 грудень 2021 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. Проектування 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	Степаненко О.О., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання “ 7 ” вересня 2021 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	7-12 тижні	Розділ 4
5	Тестування та експериментальне дослідження програми	12-14 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	15 тиждень	Додатки
7	Захист роботи	16 тиждень	

Студент(ка)

(підпис) Тарасов В.С.
(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис) Степаненко О.О.
(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи магістра: 90 с., 25 рис., 3 табл., 3 дод., 23 джерела.

ВЖИВАНІ АВТОМОБІЛІ, ПОШУК АВТОМОБІЛІВ, ПРОДАЖ АВТОМОБІЛІВ, КЛАСТЕРИЗАЦІЯ ДАНИХ, ВЕБЗАСТОСУНОК.

Об'єкт роботи – процес продажу вживаних автомобілів.

Предмет роботи – методи пошуку вживаних автомобілів.

Мета роботи – розробити програмне забезпечення пошуку вживаних автомобілів для забезпечення створення відповідного вимогам покупця портфолію пропозицій про придбання вживаних автомобілів для подальшого вибору найкращого.

Матеріали, методи та технічні засоби: мова програмування Python, система керування базою даних MySQL, вебфреймворк Django, бібліотека машинного навчання scikit-learn, метод кластеризації k-середніх.

Результати. Наукова новизна роботи полягає в запропонованому методі пошуку вживаних автомобілів, який ґрунтується на застосуванні методу k-середніх для кластеризації пропозицій, що дозволяє на основі опитувальника або обраної моделі автомобіля обирати моделі автомобілів і оголошення, які відповідають вимогам покупця.

Висновки. Розроблена програма призначена для організації процесу продажу автомобілів і комунікації в даному процесі між продавцями та покупцями. Задля цього програма надає засоби для підтримки роботи з оголошеннями, пошуку вживаних автомобілів як на основі пошуку за параметрами, так і на основі заповнення опитувальника або вибору подібної моделі автомобіля та спілкування між продавцями і покупцями.

Галузь використання – платформа для взаємодії між продавцями і покупцями вживаних автомобілів.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 90 pages, 25 figures, 3 tables, 3 appendixes, 23 sources.

USED CARS, SEARCH OF CARS, SALE OF CARS, CLUSTERIZATION OF DATA, WEB APPLICATIONS.

The object of work is the process of selling used cars.

Subject of work is methods of searching used cars.

The aim of the work is to develop software for searching used cars to enable the creation of a portfolio of proposals for the purchase of used cars to meet the requirements of the buyer to choose the best propositions on the following stages.

Materials, methods and technical means: Python programming language, MySQL database management system, Django web framework, scikit-learn machine learning library, k-means clustering method.

Results. The scientific novelty of the work is in the proposed method of searching used cars, which is based on the use of k-means method for clustering proposals, which allows to choose car models and ads which meet customer requirements based on a questionnaire or selected car model.

Conclusions. The developed program is designed to organize the process of selling cars and communication during this process between sellers and buyers. The program provides tools to support the work with ads, search for used cars both on the basis of search by parameters and on the basis of filling out a questionnaire or choosing a similar car model and communication between sellers and buyers for solving this task.

Scope is a platform for interaction between sellers and buyers of used cars.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	8
Вступ	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Сучасний стан проблеми продажу вживаних автомобілів	11
1.2 Аналіз існуючих програмних засобів пошуку вживаних автомобілів ..	12
1.3 Висновки за розділом 1	16
2 Матеріали і методи	18
2.1 Задача пошуку вживаних автомобілів	18
2.2 Метод кластеризації k-середніх	18
2.3 Метод пошуку вживаних автомобілів	20
2.4 Висновки за розділом 2	22
3 Проєктування програмного забезпечення	24
3.1 Вибір мови програмування	24
3.2 Функціональні вимоги до програмного забезпечення	25
3.3 Проєктування бази даних	27
3.4 Структура програмного забезпечення	37
3.5 Висновки за розділом 3	39
4 Основні рішення щодо реалізації компонентів системи	40
4.1 Алгоритм функціонування програми	40
4.2 Опис основних розроблених класів і шаблонів	41
4.3 Висновки за розділом 4	47
5 Експлуатація, тестування та експериментальне дослідження програми	48
5.1 Призначення програми	48
5.2 Умови виконання програми	48
5.3 Робота з програмою	48
5.4 Експериментальне дослідження	54
5.5 Висновки за розділом 5	55
Висновки	56

Перелік джерел посилання	58
Додаток А Технічне завдання.....	61
Додаток Б Текст програми.....	66
Додаток В Слайди презентації	83

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

VIN – vehicle identification number;

ПЗ – програмне забезпечення.

ВСТУП

Ринок вживаних автомобілів у сучасному світі має достатньо великий обсяг, що постійно підживлюється виробництвом нових автомобілів, які через певний час мають отримати наступного власника. Наявність такого циклу призводить до постійного збільшення обсягів ринку.

Водночас окрім цієї особливості варто також відзначити і ще одну: ринок вживаних автомобілів не тільки сегментований, але і відзначається тим, що в багатьох країнах реалізується дрібними торговцями, кожен з яких відзначається своїми особливостями опису автомобілів, підходу до продажу тощо. Цей фактор призводить до того, що пошук вживаних автомобілів для покупців є достатньо складним процесом. Подальші кроки з огляду автомобіля, вибору на практиці того екземпляру, який найбільше відповідає вимогам покупця, є окремим процесом. Проте для підтримки даного процесу потрібно спочатку створити адекватний набір пропозицій, адже неякісне виконання такого підбору призводить до того, що покупець даремно витрачає свій час на читання оголошень, які його не цікавлять, на даремний огляд на місці автомобілів.

Програмне забезпечення пошуку вживаних автомобілів повинно підтримувати не тільки процес внесення оголошень про продаж автомобілів, але і допомагати комунікації між продавцями і покупцями на наступних підетапах першого етапу їх взаємодії. Для цього зокрема мають надаватися засоби комунікації між сторонами, засоби створення системи відгуків про продавців та їх автомобілі, та засоби пошуку вживаних автомобілів. Саме об'єднання двох останніх факторів є критично важливим для того, щоб покупці користувались програмою і отримували від цього користь щодо досягнення мети всіх заходів.

ПЗ має виступати посередником для покупця, певним менеджером його інтересів, смаків, бажань і вимог. У такому випадку покупець, формулюючи свої вимоги достатньо гнучко (не в формі зазначення значень

критеріїв пошуку), зможе отримати пропозиції для придбання автомобілів, які складуть портфоліо для нього на наступних етапах. Якісно створене портфоліо є дуже важливим для раціонального використання власного часу покупцем, для підвищення його задоволеності, а значить залучення до ринку за допомогою саме розробленого ПЗ.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Сучасний стан проблеми продажу вживаних автомобілів

Компанія Fortune Business Insight у своєму звіті за результатами аналізу ринку вживаних автомобілів зазначає, що обсяг світового ринку вживаних автомобілів у 2020 році становив 1 710,75 мільярда доларів США і за прогнозами (протягом прогнозованого періоду з 2021 по 2028 рік) обсяг ринку зросте на 1,79% і, за оцінками, досягне 1 969,72 мільярда доларів США до 2028 року [1]-[3].

Щодо стану ринку вживаних автомобілів на даний момент можна відзначити наступне:

- за даними аналітиків-дослідників продажі вживаних автомобілів зросли, оскільки їхнє володіння зменшилося приблизно до 3-4 років, більше того, зменшення кількості власників і зростання цифрових технологій покращили онлайн покупку автомобілів у поєднанні з розширеними можливостями різних засобів підписки на використання автомобілів, тому очікується, що це підвищить попит на вживані транспортні засоби протягом прогнозного періоду 2021-2028 років;

- зростання схильності до приватного транспорту на тлі COVID-19, що у свою чергу також сприяє підтримці зростання: спалах пандемії коронавірусу поставив автомобільну промисловість у стан великого заміщення, після спалаху клієнти стали прихильниками того, щоб використовувати приватний транспорт через стандарти захисту та дистанції, до того ж експоненційна фінансова рецесія, ймовірно, завадить придбанню нових транспортних засобів, що призведе до того, що зокрема мандрівники віддадуть перевагу вживаним автомобілям, отже, за оцінками, це прискорить зростання ринку вживаних автомобілів під час пандемії;

- фактор девальвації автомобілів також позитивно впливає на збільшення попиту на вживані автомобілі: девальвація є серйозним фактором, який спрацьовує на користь стабільного схилення певної категорії населення до вживаних транспортних засобів порівняно зі свіжими, так

девальвація становить приблизно 30% за перший рік купівлі автомобіля, тому можна говорити про те, що високий рівень зниження вартості нових автомобілів впливає на ставлення клієнтів, які вибирають автомобілі високої вартості, що робить покупку вживаних транспортних засобів додатковим варіантом співвідношення ціни та якості, тому схвалення ринку вживаних автомобілів у суспільстві також зростає;

- ринок Європи класифікується зростаючим попитом на вживані легкові та комерційні транспортні засоби;

- сегментація: залежно від типу транспортного засобу ринок поділяється на легкові та комерційні транспортні засоби, у 2020 році на ринку домінував сегмент легкових автомобілів, уживані автомобілі з додатковою надійністю та якістю змінили підходи клієнтів і, таким чином, збільшили продажі легкових автомобілів, за віком транспортних засобів ринок поділяється на категорії від 0 до 5 років, від 6 до 10 років та від 10 років і вище, за типом продавця ринок поділяється на організований і неорганізований, за типом двигуна ринок поділяється на двигуни внутрішнього згоряння та електричні, за типом ринку світовий ринок поділяється на офлайн і онлайн, у свою чергу географічно глобальний ринок розділений на Північну Америку, Європу, Азіатсько-Тихоокеанський регіон та решту світу [1]-[3].

1.2 Аналіз існуючих програмних засобів пошуку вживаних автомобілів

Auto Ria (рис. 1.1) є найбільшим майданчиком в Україні для продажу вживаних автомобілів [4]. Проте на даний момент майданчик надає також засоби пошуку нових автомобілів, нерухомості, автотоварів, автозапчастин тощо.

Програмою надається великий набір інструментів щодо роботи з вживаними автомобілями, зокрема створення оголошень про продаж автомобілів, пошуку автомобілів з великою кількістю фільтрів пошуку,

керування власними оголошеннями, створення власного нотатника з оголошеннями, доступу до звітів про перевірку автомобіля, створення відгуків про автомобілі, перегляду таких відгуків.

RIA.com
Автомобілі
Нерухомість
Автотовари
Автозапчастини

☆

🔔 1

👤

Особистий кабінет

Рус

Укр

auto

RIA

Вживані авто

Нові авто

Новини

Все для авто

+

Продати авто

Додавання оголошення

Основна інформація

Вкажіть характеристики вашого авто

1

Додайте 2-3 фото з відкритим держ. номером

щоб автоматично заповнити технічні дані авто

?

Не переживайте, ви можете

📄

замалювати держ. номер

2

Основна інформація

?

тип транспорту, марка, модель, регіон

*

Поля обов'язкові для заповнення

Тип транспорту

*

Оберіть

Марка авто

*

Оберіть

Модель авто

*

Оберіть

Рік випуску

*

Оберіть

Модифікація

Модифікація

Тип кузова

*

Оберіть

Пробіг

*

тис.км

Регіон

*

Оберіть

Місто

*

Оберіть

Авто зі статусом «Перевірений VIN-код» продаються удвічі швидше

Вкажіть VIN-код і ми безкоштовно перевіримо авто на предмет розшуку й дати останньої операції (за реєстрами МВС), а також звіримо інформацію з порталом відкритих даних і дилерськими базами.

Рисунок 1.1 – Вебсайт Auto Ria

Кожен користувач після реєстрації на вебсайті та внесення свого номеру телефону може створювати оголошення про продаж автомобілів.

Американський вебсайт Cars.com також надає засоби пошуку вживаних автомобілів (рис. 1.2) [5].

Доступний пошук автомобілів за параметрами у скороченій та розширеній версії. При цьому розширена версія Auto Ria має більшу кількість параметрів відносно Cars.com.


[Cars for Sale](#)
[Research & Reviews](#)
[News & Videos](#)
[Sell Your Car](#)
[Service & Repair](#)
[Sign In](#) / [Sign Up](#)

[Home](#) / [Cars for Sale](#) / [Search Results](#) / Used 2020 Subaru Ascent Limited 7-Passenger

[< Previous car](#)



\$42,442



\$44,987



\$41,010



\$39,500



\$44,077



\$44,915



\$40,975

[Next car >](#)



1/15










Used

2020 Subaru Ascent Limited 7-Passenger

25,114 mi.

\$42,442

[Good Deal](#)
[Home Delivery](#)
[Virtual Appointments](#)

Contact seller

Call (866) 779-4742

Subject

Check availability

Comments

I'd like to know if the Used 2020 Subaru Ascent Limited 7-Passenger you have listed on Cars.com for \$42,442 is still available.

☐ I have a trade in

Check availability

By clicking here, you authorize Cars.com and its sellers/partners to contact you by text/calls which may include marketing and be by autodialer. Calls may be pre-recorded. You also agree to our [Privacy Statement](#). Consent is not required to purchase goods/services.

We value your privacy. [Cars.com Privacy Statement](#)

Рисунок 1.2 – Вебсайт Cars.com

Коли покупець збирається придбати транспортний засіб, то перше, що він зробить, це виконує пошук в Інтернеті, так дослідження британської

компанії DMA показало, що 68 відсотків людей покладаються на огляди автомобілів для прийняття рішень щодо покупки, при цьому Cars.com є одним з провідних постачальників інструментів для ціноутворення та дослідження для автомобільної промисловості, щомісячно приблизно 15,5 мільйонів людей відвідує сайт, платформа допомагає підібрати для потенційних покупців автомобілів їх ідеальні транспортні засоби на основі рейтингів безпеки та відгуків споживачів і галузей, після того, як Cars.com знаходить найкращу модель автомобіля для покупця, він допоможе покупцю знайти цей автомобіль серед оголошень [6].

Порівняючи представлені застосунки та застосунок, що розробляється, варто вказати на декілька особливостей, які не були виявлені в проаналізованих програмних аналогах і які планується розробити в роботі.

Базовою характеристикою автомобіля є його vehicle identification number (VIN)-номер. VIN-номер дозволяє шукати попередні оголошення про продаж автомобілю. Таким чином інформація з попередніх оголошень може уточнити те, що не вказано в поточному оголошенні. Це дозволяє знайти відгуки на оголошення в момент початку продажу автомобіля і дізнатися про виявлені раніше проблеми з автомобілем. Проте ні Auto Ria, ні Cars.com не надають такої можливості, хоча і зазначають VIN-номер автомобіля.

Зокрема якщо користувач діє в умовах обмеженої пропозиції, то він може під час пошуку концентруватися надто довго над одним і тим же самим набором моделей. У такому випадку пошук необхідно розширити. Розширити потрібно пошук за рахунок розширення бази моделей. У такому випадку корисним буде інструмент пошуку подібних автомобілів. Тобто за виділеним екземпляром знайти подібні моделі і далі аналізувати оголошення про продаж даної моделі. Окрім того корисним може бути пошук, коли користувач не зазначає значення стандартного набору параметрів, а відповідає на запитання про власні переваги. Тоді програма буде діяти для нього в ролі менеджера.

Приведені результати аналізу програмних аналогів зведено до табл. 1.1.

Таблиця 1.1 – Результати аналізу програмних аналогів

Характеристика	Auto Ria	Cars.com	Програма TarasovMarket
Безкоштовний доступ	+	+	+
Пошук за великою кількістю параметрів	+	+	+
Система відгуків	+	+	+
Визначення вартості пропозицій за моделлю	+	+	+
Комунікація з продавцем	+	+	+
Пошук за оголошеннями за VIN-номером	—	—	+
Альтернативні сценарії пошуку	—	—	+

1.3 Висновки за розділом 1

Проведено аналіз предметної області, за результатами чого розглянуто сучасний стан проблеми продажу вживаних автомобілів, аналіз існуючих програмних засобів пошуку вживаних автомобілів з виконанням порівняння цих засобів з програмою, що розробляється. В якості особливостей останньої виділено пошук за оголошеннями за VIN-номером, що дозволяє знайти відгуки на оголошення в момент початку продажу автомобіля і дізнатися про виявлені раніше проблеми з автомобілем, та альтернативні сценарії пошуку, які за виділеним екземпляром дозволяють знайти подібні моделі і далі аналізувати оголошення про продаж даної моделі або знайти автомобілі за виділеними перевагами покупця.

Задачі, що мають бути досягнуті в роботі, передбачають виконання наступних завдань:

- виконати аналіз предметної області;
- виконати дослідження методів пошуку вживаних автомобілів;
- виконати проєктування програми;
- виконати реалізацію ПЗ.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача пошуку вживаних автомобілів

Задача пошуку вживаних автомобілів полягає у визначенні масиву автомобілів, які відповідають параметрам пошуку. Цей масив є підмножиною множини всіх наявних вживаних автомобілів у програмі.

Для того щоб визначити відповідну підмножину автомобілів, потрібно розділити всі наявні автомобілі на групи і визначити, яка група найкраще відповідає заданому образу, що формують параметри пошуку.

Параметри пошуку можуть бути визначені двома способами:

- конкретним автомобілем: тоді відбувається пошук подібних автомобілів;
- опитувальником: тоді визначаються значення параметрів за відповідями на запитання.

2.2 Метод кластеризації k-середніх

Найважливішою проблемою навчання без учителя можна вважати кластеризацію:

- як і будь-яка інша проблема такого роду, вона стосується пошуку структури в колекції нерозмічених даних;
- вільне визначення кластеризації може бути процесом організації об'єктів у групи, члени яких у чомусь схожі;
- кластер – це сукупність об'єктів, які подібні між собою і несхожі на об'єкти, що належать до інших кластерів;
- кластеризація на основі відстані: дано набір точок з визначенням відстані між точками, групуючи точки в деяку кількість кластерів, так що внутрішні (всередині кластера) відстані повинні бути малими, тобто члени кластерів подібні один до одного, зовнішні відстані повинні бути великими, тобто члени різних кластерів не схожі;

– метою кластеризації є визначення внутрішнього групування в наборі нерозмічених даних [7]-[11].

Метод кластеризації k -середніх – це ітеративний і заснований на прототипах метод кластеризації без учителя, де всі точки даних групуються в кластери, кожен з яких представлений своїми центроїдами (прототипом), відносно даного методу діють наступні основні принципи:

– центроїд кластера часто є середнім значенням усіх точок даних у цьому кластері;

– точки даних у кластері найближчі до центроїдів цього кластера: існує висока подібність між точками даних у кластері (висока подібність всередині класу або всередині кластера), а точки даних з одного кластера відрізняються від точок даних іншого кластера (низька міжкласова або міжкластерна подібність);

– подібність і відмінність обчислюються за допомогою відстані кластеризації між точками даних, його обчислюють за допомогою евклідової або кореляційної відстані (наприклад, кореляційні відстані Пірсона, Спірмена, Кендалла) [12]-[14].

Метод k -середніх є еволюційним алгоритмом, який об'єднує спостереження в k груп, де k надається як вхідний параметр, потім метод призначає кожне спостереження кластерам на основі близькості спостереження до середнього значення кластера, середнє значення кластера потім перераховується, і процес починається знову, формуючи наступну процедуру:

– довільно вибираються k точок як початкові центри кластера (середні);

– кожна точка в наборі даних призначається замкнутому кластеру на основі евклідової відстані між кожною точкою та центром кожного кластера;

– кожен центр кластера перераховується як середнє значення точок у цьому кластері;

– попередні два кроки повторюються, поки кластери не стануть стабільними: стабільність може бути визначена по-різному залежно від реалізації, але зазвичай це означає, що або жодні спостереження не змінюють кластери, коли кроки повторюються, або що зміни не впливають на визначення кластерів [11]-[15].

Одним з основних недоліків кластеризації k -середніх є той факт, що потрібно вказати кількість кластерів як вхідні дані для алгоритму, при цьому метод не здатний визначити відповідну кількість кластерів і залежить від користувача, щоб визначити це заздалегідь, оскільки визначити точно число k може бути складно, то з цієї причини часто доцільно експериментувати з різними значеннями k , щоб визначити значення, яке найкраще відповідає вашим даним [15].

2.3 Метод пошуку вживаних автомобілів

Метод пошуку вживаних автомобілів призначений для вирішення задачі, сформульованої в підрозділі 2.1.

На першому етапі роботи методу пошуку вживаних автомобілів формується опитувальник, який складається з запитань, кожне з яких направлено на визначення значення відповідного параметра автомобіля. Це підготовчий етап.

Для кожного запитання визначається правило, яке перетворює відповідь на запитання на значення або набір допустимих значень однієї з характеристик автомобіля. Набір правил формує масив A .

Далі для реалізації методу пошуку вживаних автомобілів мають бути сформовані кластери, до яких ці автомобілі належать.

Для цього спочатку має бути задано значення k , вказуючи на кількість кластерів, з якими буде відбуватися робота, а також визначено набір ознак, за якими кластеризація буде відбуватися.

Після цього потрібно створити вибірку всіх автомобілів, наявних в програмі. Вони витягаються з бази даних. Значення за всіма параметрами мають бути нормалізовані.

Далі всі ці автомобілі потрібно розділити на кластери, використовуючи метод k -середніх.

На наступному етапі визначаються значення параметрів пошуку вживаних автомобілів. Для цього існують два окремі способи, кожен з яких можна використовувати за наявності виділених кластерів.

Значення параметрів пошуку можуть визначатися опитувальником або значеннями параметрів одного з автомобілів. Тоді такий автомобіль представляє собою приклад для пошуку. На наступному етапі відбувається пошук автомобіля, подібного до даного.

Якщо заповнюється опитувальник, то отримані відповіді на запитання переводяться за допомогою масиву правил A на значення параметрів. Тоді всі значення параметрів формують крапку в просторі пошуку. Відповідно на наступному етапі реалізується пошук автомобілів, які відповідають цим параметрам пошуку.

На наступному етапі обчислюється відстань від визначеної крапки в просторі до кожного з k кластерів. Кластер, який має найменшу відстань, визнається переможцем. У результаті екземпляри, які визначають активні оголошення з даного виділеного кластеру, повертаються як масив результатів пошуку B .

Після цього елементи в масиві результатів пошуку B сортуються за збільшенням відстані від заданої крапки в просторі до кожного з екземплярів кластера.

Описаний метод пошуку вживаних автомобілів представлено на рис. 2.1.

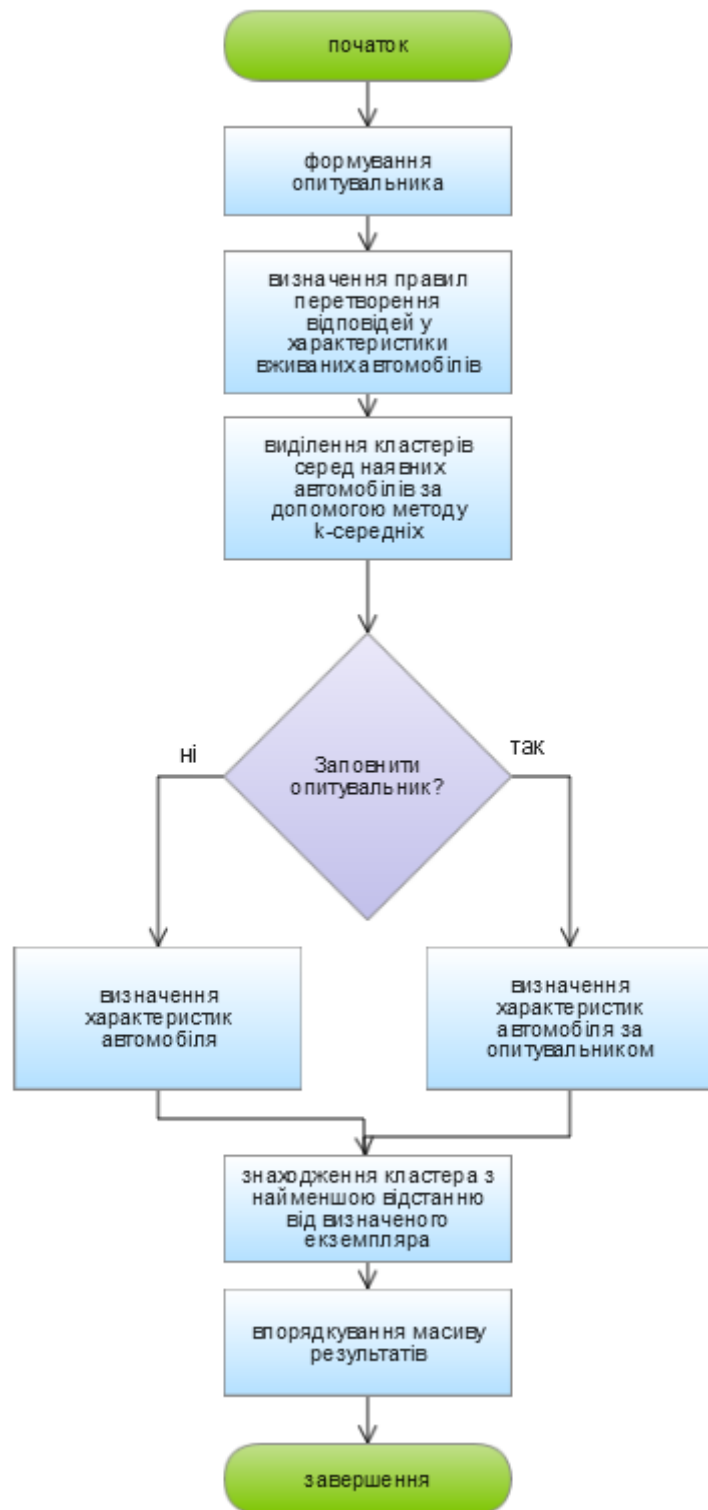


Рисунок 2.1 – Метод пошуку вживаних автомобілів

2.4 Висновки за розділом 2

Проведений аналіз методів кластеризації, які можуть використовуватися для вирішення сформульованої задачі пошуку вживаних

автомобілів, дозволив створити власний метод, який визначив один з основних результатів роботи. Наукова новизна роботи полягає в запропонованому методі пошуку вживаних автомобілів, який ґрунтується на застосуванні методу k-середніх для кластеризації пропозицій, що дозволяє на основі опитувальника або обраної моделі автомобіля обирати моделі автомобілів і оголошення, які відповідають вимогам покупця.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибір мови програмування

На даний момент, коли мова йде про веброзробку, то найчастіше зустрічаються дві мови програмування – Python і PHP, тому саме їх було обрано для порівняння (табл. 3.1) [16]-[18].

Таблиця 3.1 – Порівняння мов програмування

Критерій	Python	PHP
Основні парадигми програмування	Об'єктно-орієнтована, функціональна	Об'єктно-орієнтована, обмежено функціональна
Простота розробки вебзастосунків	Висока	Висока
Орієнтованість на довготривалі проєкти	Більша	Менша
Простота створення програм	Більша	Менша
Підтримка методів інтелектуального аналізу даних, у тому числі методу k-середніх	Краща	Гірша
Призначення	Загальне	Веброзробка

Кінцевий вибір мови програмування Python для розробки обумовлений тим, що вона більше орієнтована на довготривалі проєкти, краще їх підтримує (система, що розробляється, передбачає довготривалий цикл її використання), дозволяє простіше створювати код, а також має простіші і зручніші засоби інтелектуального аналізу даних.

3.2 Функціональні вимоги до програмного забезпечення

ПЗ має реалізовувати такі функціональні можливості для надання ефективних інструментів щодо взаємодії покупця і продавця:

- створення відгуку про автомобіль за оголошенням про продаж;
- створення відгуку про власника автомобіля;
- створення відгуку про модель автомобіля;
- пошук оголошень про продаж автомобілів;
- підбір автомобіля за вимогами покупця;
- публікація оголошення про продаж автомобіля;
- перегляд оголошень власника;
- перегляд оголошення про продаж автомобіля;
- висування пропозиції про продаж автомобіля або про огляд;
- приймання пропозиції про купівлю автомобіля або про огляд;
- скасування пропозиції;
- керування власними оголошеннями, у тому числі зняття автомобіля з продажу;
- перегляд візитної сторінки моделі автомобіля;
- створення закладки на оголошенні про продаж автомобіля;
- перегляд створених закладок;
- перегляд профілю користувача;
- перегляд історії вартості автомобіля;
- перегляд чату між власником і покупцем;
- відправлення повідомлення в чат;
- порівняння оголошень;
- перегляд власних пропозицій;
- пошук подібних автомобілів;
- пошук оголошень за VIN-номером автомобіля;
- перегляд рейтингу оголошень за середнім балом, за кількістю переглядів, створених закладок, за часом створення;

– перегляд рейтингу моделей автомобілів.

Виділені функціональні вимоги було об'єднано в сценарії використання програми користувачем (рис. 3.1).

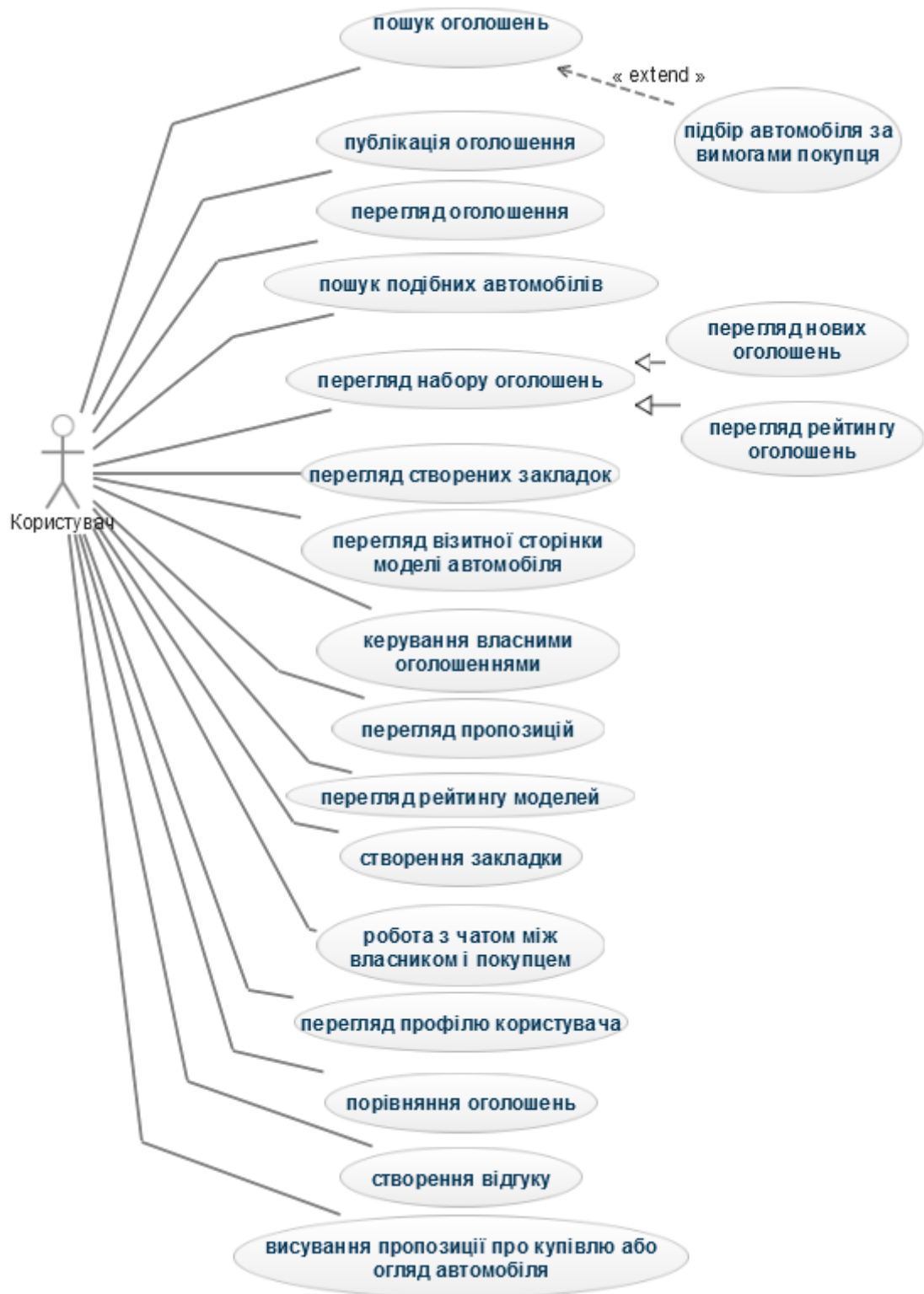


Рисунок 3.1 – Діаграма варіантів використання ПЗ користувачем

3.3 Проектування бази даних

За результатами виконаного аналізу предметної області, визначених функціональних вимог до програми, було виконано проектування бази даних. Системою керування базами даних було обрано MySQL.

Структуру бази даних описано нижче.

Таблиця `Producer` визначає виробника автомобілів і містить поля:

- `id` – ідентифікатор виробника;
- `title` – назва виробника.

Таблиця `CarModel` визначає модель автомобіля і містить поля:

- `id` – ідентифікатор моделі автомобіля;
- `title` – назва моделі автомобіля;
- `score` – середня оцінка, виставлена моделі автомобіля;
- `min_price` – мінімальна вартість автомобіля даної моделі;
- `max_price` – максимальна вартість автомобіля даної моделі;
- `average_price` – середня вартість автомобіля даної моделі;
- `producer_id` – виробник автомобіля, що представлений зовнішнім ключем з таблиці `Producer`;
- `image` – фотографія моделі автомобіля.

Кожен виробник представляє декілька моделей автомобілів, тому таблиця `Producer` зв'язана з таблицею `Model` зв'язком «один до багатьох».

Таблиця `Engine` визначає двигун автомобіля і містить поля:

- `id` – ідентифікатор двигуна;
- `enginetype` – тип двигуна автомобіля, що представлений зовнішнім ключем з таблиці `EngineType`;
- `horsepower` – кількість кінських сил двигуна;
- `title` – назва двигуна;
- `literes` – обсяг двигуна;
- `fuel` – тип пального, який використовується двигуном, що представлений зовнішнім ключем з таблиці `Fuel`;

- cylinders – кількість циліндрів двигуна;
- cityfuel – витрата пального в місті на 100 кілометрів шляху;
- routefuel – витрата пального на трасі на 100 кілометрів шляху;
- mixfuel – витрата пального в змішаному циклі на 100 кілометрів шляху.

Кожен тип двигуна представлений декількома конкретними двигунами, тому таблиця EngineType зв'язана з таблицею Engine зв'язком «один до багатьох».

Таблиця Package визначає комплектацію автомобіля і містить поля:

- id – ідентифікатор комплектації автомобіля;
- title – назва комплектації;
- description – опис комплектації.

Таблиця Fuel визначає тип пального і містить поля:

- id – ідентифікатор типу пального;
- title – назва типу пального.

Таблиця EngineType визначає тип двигуна і містить поля:

- id – ідентифікатор типу двигуна;
- title – назва типу двигуна.

Таблиця Proposition визначає пропозицію про купівлю автомобіля (або огляд автомобіля) і містить поля:

- id – ідентифікатор пропозиції про купівлю автомобіля;
- car – автомобіль, щодо купівлі якого визначено пропозицію, представлений зовнішнім ключем з таблиці Car;
- user – користувач, який висуває пропозицію про купівлю автомобіля, представлений зовнішнім ключем з таблиці User;
- number – сума, за яку покупець готовий придбати автомобіль;
- state – стан пропозиції;
- created – дата висунення пропозиції;
- description – повідомлення для власника про пропозицію;
- meeting – час зустрічі для купівлі або огляду;

– place – місце зустрічі для купівлі або огляду.

Кожне оголошення про продаж автомобіля може мати декілька пропозицій про купівлю автомобіля, тому таблиця Car зв'язана з таблицею Proposition зв'язком «один до багатьох». Кожен користувач може висунути декілька своїх пропозицій про купівлю автомобіля за оголошенням, тому таблиця User зв'язана з таблицею Proposition зв'язком «один до багатьох».

Таблиця Car визначає автомобіль (оголошення про продаж автомобіля) і містить поля:

- id – ідентифікатор автомобіля;
- engine – двигун автомобіля, що представлений зовнішнім ключем з таблиці Engine;
- transmission – тип трансмісії;
- doors – кількість дверей у автомобілі;
- places – кількість сидячих місць автомобіля;
- path – пройдений шлях;
- color – колір автомобіля, що представлений зовнішнім ключем з таблиці Color;
- score – середня оцінка, виставлена автомобілю;
- viewed – кількість переглядів оголошення;
- liked – кількість уподобань оголошення;
- propositions – кількість пропозицій, висунутих про придбання автомобіля;
- user – власник автомобіля, що представлений зовнішнім ключем з таблиці User;
- cartype – тип кузова автомобіля, що представлений зовнішнім ключем з таблиці CarType;
- produced – дата виробництва автомобіля;
- used – дата, з якої використовується автомобіль;
- location – місце розташування автомобіля, що представлений зовнішнім ключем з таблиці Location;

- regnumber – реєстраційний номер автомобіля;
- vin – VIN-номер автомобіля;
- package – комплектація автомобіля, що представлена зовнішнім ключем з таблиці Package;
- created – дата створення оголошення;
- problem – опис проблем, виявлених у автомобіля;
- activity – активність оголошення;
- drive – тип приводу автомобіля;
- cityfuel – витрата пального в місті на 100 кілометрів шляху;
- routefuel – витрата пального на трасі на 100 кілометрів шляху;
- mixfuel – витрата пального в змішаному циклі на 100 кілометрів шляху;
- carmodel – модель автомобіля.

Таблиця Car має зв'язки з наступними таблицями:

- таблицею Engine через поле EngineId: один двигун (модель двигуна) може бути встановлений на різних автомобілях, тому зв'язок між таблицями «один до багатьох»;
- таблицею User через поле UserId: один користувач може мати декілька оголошень про продаж автомобілів, тому зв'язок між таблицями «один до багатьох»;
- таблицею CarType через поле CarTypeId: один тип кузова представлений різними автомобілями з відповідними оголошеннями про їх продаж, тому зв'язок між таблицями «один до багатьох»;
- таблицею Package через поле PackageId: одна комплектація може бути представлені різними автомобілями з відповідними оголошеннями про їх продаж, тому зв'язок між таблицями «один до багатьох»;
- таблицею Location через поле LocationId: в одному місті може бути велика кількість оголошень про продаж автомобілів, тому зв'язок між таблицями «один до багатьох».

Таблиця Location визначає локацію і містить поля:

- id – ідентифікатор міста;
- title – назва міста.

Таблиця State визначає стан і містить поля:

- id – ідентифікатор стану;
- title – назва стану;
- description – опис стану;
- type – тип стану (технічний, стан лакофарбового покриття).

Оскільки одним і тим же самим станом можуть характеризуватися різні автомобілі, а кожен автомобіль може мати різні стани, адже кожен окремий стан автомобіля може відноситися до різних груп (загальний стан автомобіля, стан лакофарбового покриття), то між таблицями Car і State виникає зв'язок «багато до багатьох», який має бути реалізовано створенням окремої таблиці CarState.

Таблиця CarState визначає стан автомобіля і містить поля:

- id – ідентифікатор запису про стан автомобіля;
- car – автомобіль, до якого належить цей запис, що представлений зовнішнім ключем з таблиці Car;
- state – стан, яким характеризується автомобіль, що представлений зовнішнім ключем з таблиці State.

Таблиця CarType визначає тип кузова автомобіля і містить поля:

- id – ідентифікатор типу кузова автомобіля;
- title – назва типу кузова автомобіля.

Таблиця Marker визначає характеристику (маркер), що описує автомобіль, і містить поля:

- id – ідентифікатор характеристики автомобіля;
- title – назва характеристики;
- type – тип характеристики.

Оскільки однією і тією ж самою характеристикою (маркером) можуть характеризуватися різні автомобілі, а кожен автомобіль може мати різні

характеристики, то між таблицями Car і State виникає зв'язок «багато до багатьох», який має бути реалізовано створенням окремої таблиці CarMarker.

Таблиця CarMarker визначає характеристику конкретного автомобіля і містить поля:

- id – ідентифікатор запису про характеристику автомобіля;
- marker – характеристика автомобіля, що представлений зовнішнім ключем з таблиці Marker;
- car – автомобіль, до якого належить цей запис, що представлений зовнішнім ключем з таблиці Car.

Таблиця Color визначає колір і містить поля:

- id – ідентифікатор кольору;
- title – назва кольору;
- code – код кольору (HTML-код).

Таблиця User визначає користувача і містить поля:

- id – ідентифікатор користувача;
- name – повне ім'я користувача;
- username – логін;
- password – пароль;
- phone – номер телефону користувача;
- score – середній бал, виставлений цьому користувачу (власнику автомобілів);
- registered – дата реєстрації користувача;
- visited – дата останнього візиту користувача.

Таблиця Photo визначає фотографію автомобіля і містить поля:

- id – номер фотографії;
- file – шлях до файлу автомобіля;
- car – автомобіль, який характеризує ця фотографія, представлений зовнішнім ключем з таблиці Car.

Кожне оголошення про продаж автомобіля може мати одразу декілька фотографій, тому таблиця Car зв'язана з таблицею Photo зв'язком «один до багатьох».

Таблиця Like визначає уподобання автомобіля і містить поля:

- id – ідентифікатор уподобання;
- user – користувач, який уподобав оголошення, представлений зовнішнім ключем з таблиці User;
- car – оголошення, уподобане користувачем, представлене зовнішнім ключем з таблиці Car;
- liked – дата уподобання;
- message – нотатка про автомобіль від покупця.

Кожне оголошення може мати декілька уподобань, але кожне уподобання належить тільки одному оголошенню, тому таблиця Car зв'язана з таблицею Like зв'язком «один до багатьох». Кожен користувач може мати декілька уподобаних оголошень, тому таблиця User зв'язана з таблицею Like зв'язком «один до багатьох».

Таблиця View визначає перегляд оголошення і містить поля:

- id – ідентифікатор перегляду оголошення;
- user – користувач, який переглянув оголошення, представлений зовнішнім ключем з таблиці User;
- car – переглянуте оголошення, представлене зовнішнім ключем з таблиці Car;
- viewed – дата перегляду оголошення.

Кожне оголошення може переглядатися різними користувачами, тому таблиця Car зв'язана з таблицею View зв'язком «один до багатьох». Кожен користувач може переглядати декілька оголошень, тому таблиця User зв'язана з таблицею View зв'язком «один до багатьох».

Таблиця Price визначає запис про вартість автомобіля і містить поля:

- id – ідентифікатор запису про вартість автомобіля;

- car – автомобіль, щодо якого встановлено вартість, представлений зовнішнім ключем з таблиці Car;

- number – вартість автомобіля;

- created – дата внесення запису про вартість автомобіля.

Кожне оголошення може мати і різні моменти різну встановлену вартість, що формує історію вартості, тому таблиця Car зв'язана з таблицею Price зв'язком «один до багатьох».

Таблиця Chat визначає чат між власником автомобіля і потенційним покупцем і містить поля:

- id – номер повідомлення;

- user1 – власник автомобіля, якому адресовано повідомлення, представлене зовнішнім ключем з таблиці User;

- user2 – користувач, який написав повідомлення, представлений зовнішнім ключем з таблиці User;

- message – текст повідомлення;

- created – дата публікації повідомлення.

Кожен користувач може відправляти (як покупець) і отримувати (як продавець) повідомлення від різних інших користувачів, тому таблиця User зв'язана з таблицею Chat зв'язком «один до багатьох» двічі.

Таблиця ModelReview визначає відгук на модель автомобіля і містить поля:

- id – ідентифікатор відгуку;

- user – користувач, який створив відгук, представлений зовнішнім ключем з таблиці User;

- carmodel – модель автомобіля, про яку написано відгук, представлений зовнішнім ключем з таблиці CarModel;

- created – дата публікації відгуку;

- score – оцінка у відгуку;

- message – текст відгуку.

Кожна модель автомобіля може мати декілька відгуків, тому таблиця Model зв'язана з таблицею ModelReview зв'язком «один до багатьох». Кожен користувач може мати декілька відгуків про моделі автомобілів, тому таблиця User зв'язана з таблицею ModelReview зв'язком «один до багатьох».

Таблиця CarReview визначає відгук на конкретний автомобіль (оголошення) і містить поля:

- id – ідентифікатор відгуку;
- user – користувач, який створив відгук, представлений зовнішнім ключем з таблиці User;
- car – автомобіль, на оголошення про продаж якого написано відгук, представлений зовнішнім ключем з таблиці Car;
- created – дата публікації відгуку;
- score – оцінка у відгуку;
- message – текст відгуку.

Кожне оголошення може мати декілька відгуків, тому таблиця Car зв'язана з таблицею CarReview зв'язком «один до багатьох». Кожен користувач може мати декілька відгуків про оголошення про продаж автомобілів, тому таблиця User зв'язана з таблицею CarReview зв'язком «один до багатьох».

Таблиця БД (рис. 3.2) визначає відгук на власника автомобіля і містить поля:

- id – ідентифікатор відгуку;
- user1 – користувач, власник автомобіля, про якого написано відгук, представлений зовнішнім ключем з таблиці User;
- user2 – користувач, який написав відгук, представлений зовнішнім ключем з таблиці User;
- created – дата публікації відгуку;
- score – оцінка у відгуку;
- message – текст відгуку.

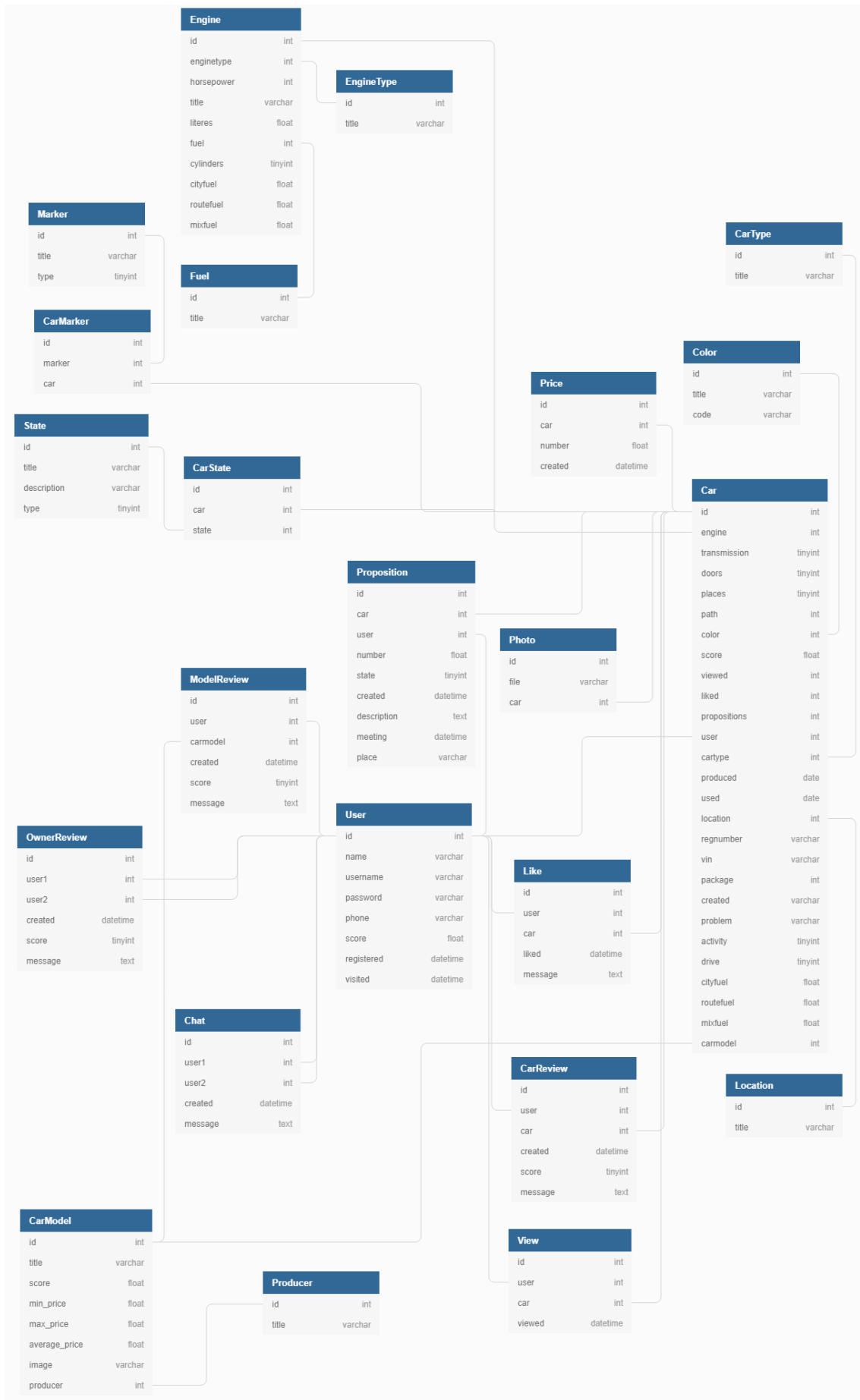


Рисунок 3.2 – Таблица БД

Кожен власник може мати декілька відгуків про себе як власника і відповідно кожен користувач як покупець може мати декілька відгуків про продавців, тому таблиця User зв'язана з таблицею OwnerReview зв'язком «один до багатьох» двічі.

3.4 Структура програмного забезпечення

На структуру ПЗ впливають обрані засоби розробки. На даному етапі обрану мову програмування Python було підсилено використанням вебфреймворку Django.

Фреймворк Django має такі переваги:

- вебфреймворк простий у використанні та впровадженні, що призводить до економії часу на розробку.
- фреймворк Django дозволяє створювати великі вебзастосунки та REST API з найменшими зусиллями, надає об'єктно-реляційне відображення, щоб маніпулювати значеннями в таблицях бази даних за допомогою екземплярів класів, визначених у файлі models.py, що спрощує процес роботи з базою даних, не потрібно писати складні інструкції SQL для безпосередньої обробки даних, однак Django також дозволяє писати реальні інструкції SQL, якщо це потрібно зробити;
- фреймворк Django надає класи для роботи з представленнями, при цьому вони на стільки добре розроблені, що можна маніпулювати кожним аспектом об'єктів запиту та відповіді;
- зміна одного компонента не впливає на інші компоненти: можна гнучко додати деякі поля до класу models.py, але це не вплине на функції представлення, представлення не зможе отримати значення нових полів у класі models.py, але це не пошкодить існуючий файл з представленнями [19]-[21].

Створену структуру програми за допомогою вебфреймворку Django схематично представлено на рис. 3.3.

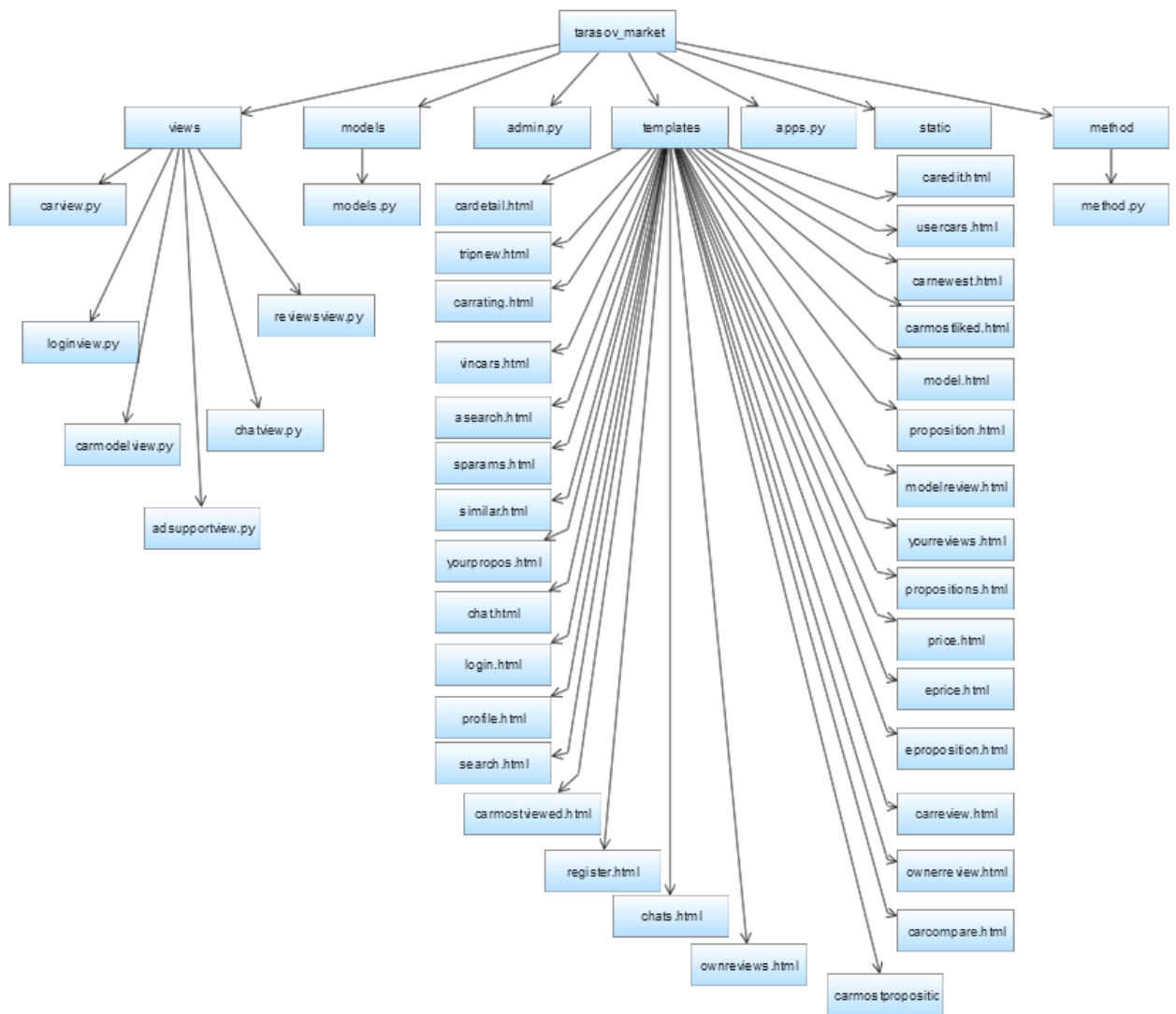


Рисунок 3.3 – Схема структури програми

Вебзастосунок складається з основних одиниць, що включають:

- models – моделі застосунку для роботи над даними;
- views – представлення застосунку;
- method містить скрипт, в якому реалізовано метод;
- templates – шаблони застосунку;
- admin.py – скрипт, в якому реалізується реєстрація моделей для роботи адміністративної панелі застосунку;
- static – статичні файли проєкту;
- apps.py – скрипт, в якому реалізується налаштування застосунку.

3.5 Висновки за розділом 3

Виконано проєктування програми за допомогою мови програмування Python, системи керування базою даних MySQL, вебфреймворку Django. Користуючись цими рішеннями, визначено структуру програми та описано структуру бази даних.

4 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування програми

Алгоритм функціонування програми (рис. 4.1) створено на основі режимів роботи, які послідовно обирає користувач. Алгоритм визначає послідовність таких переходів.

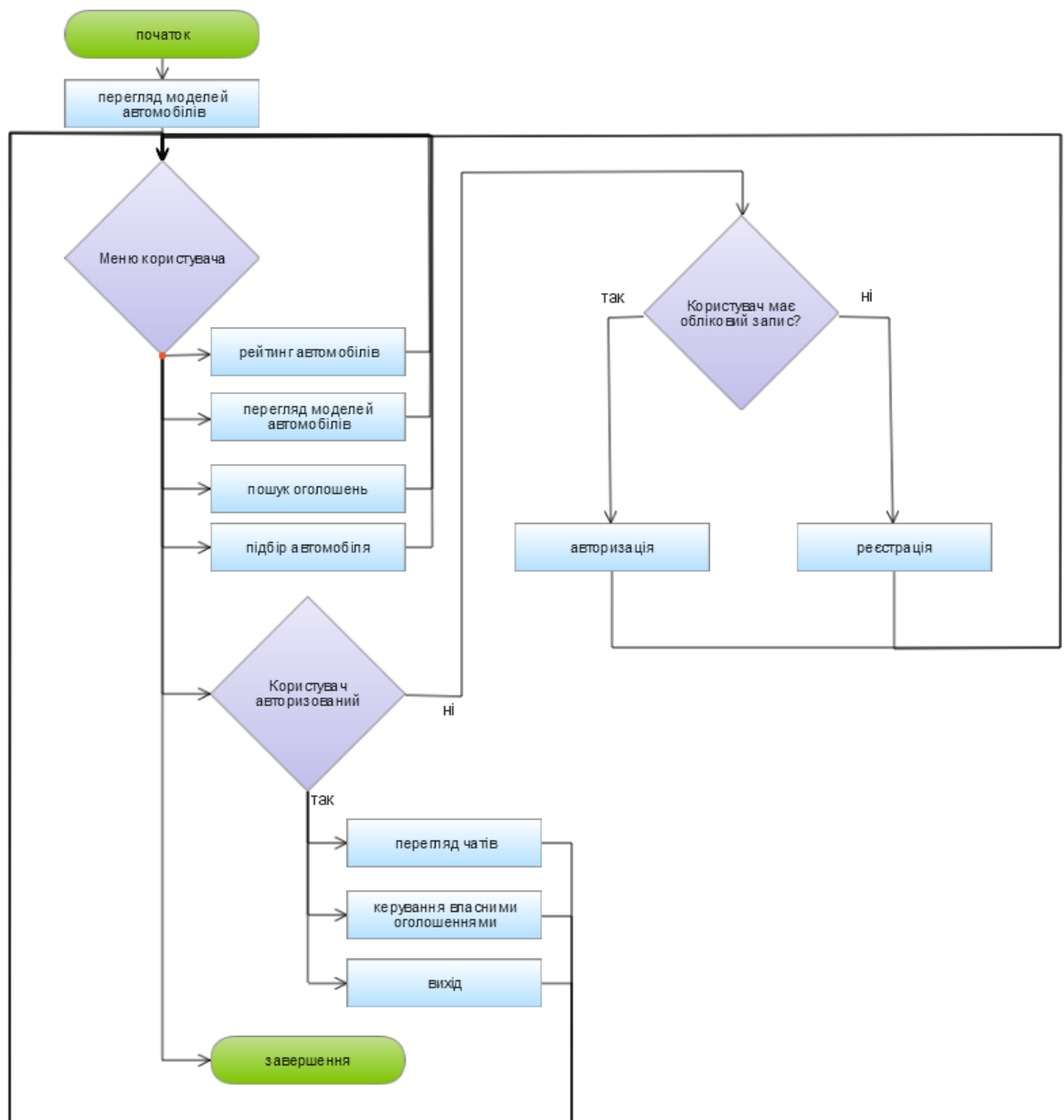


Рисунок 4.1 – Схема функціонування програми

Спочатку користувач отримує доступ до перегляду моделей автомобілів. Даний етап включає цілий набір варіантів роботи, адже звідси можна обрати виробника, перейшовши до обмеженого набору моделей, переглянути візитну сторінку обраної моделі, а також перейти до оголошень про продаж вживаних автомобілів цієї моделі.

Користувач може або продовжити роботу в цьому режимі, або обрати інший режим з меню. Меню включає частину пунктів, доступну всім користувач, і частину пунктів, для роботи з якими потрібно спочатку авторизуватися.

Для користувача без авторизації доступний перегляд моделей автомобілів, пошук оголошень про продаж вживаних автомобілів, підбір автомобілів на основі опитувальника.

Якщо користувач виконає авторизацію у власному обліковому запису, то йому буде відкрито доступ до роботи з чатами, звідки він може переглянути своє спілкування з власниками автомобілів, а також спілкування з покупцями, якщо він розмістив оголошення про продаж автомобіля на вебсайті.

Користувач після авторизації може керувати власними оголошеннями, в тому числі змінювати стан вже створених раніше оголошень, а також за потреби переходити до публікації нових оголошень.

4.2 Опис основних розроблених класів і шаблонів

Представлення вебзастосунку розроблені для визначення основної функціональної логіки під час заповнення даними шаблонів. Вони створені у вигляді класів.

Клас CarView забезпечує заповнення даними шаблону cardetail.html відносно обраного для перегляду оголошення на основі використання стандартного класу представлення DetailView деталізації об'єкта. Наслідування готових класів значно спрощує процес розробки. Для

заповнення даними в методі `get_context_data` реалізується витягання об'єкта оголошення про продаж автомобіля з номером, що відповідає заданому в параметрах значенню первинного ключа:

```
carcont = super().get_context_data(**kwargs)
carinfo = Car.objects.get(id=self.kwargs['pk'])
carcont['car'] = carinfo
```

Потім реалізується додаткове внесення відгуків, фотографій, пропозицій про купівлю автомобіля для заповнення ними контексту даних шаблону вебсторінки:

```
carreviews = CarReview.objects.filter(pk = self.kwargs['pk'])
carcont['reviews'] = carreviews
photos = Photo.objects.filter(car__id = self.kwargs['pk'])
carcont['photos'] = photos
propositions = Proposition.objects.filter(car__id = self.kwargs['pk'])
carcont['propositions'] = propositions
```

Також до контексту даних додається історія зміни вартості, а також виділяється окремо поточна вартість, для цього реалізується звернення до пов'язаної моделі:

```
prices = Price.objects.filter(car__id = self.kwargs['pk'])
carcont['prices'] = prices
currprice = prices[0].number
carcont['currprice'] = currprice
return carcont
```

Клас `ModelView` також побудований наслідуванням класу `DetailView` і забезпечує заповнення даних про модель автомобіля, включаючи відгуки про модель автомобіля.

Аналогічно побудований клас `UIView`, який створює контекст даних для перегляду інформації про користувача.

Клас `CarNewView` забезпечує публікацію оголошення про продаж автомобіля, наслідуючи клас `LoginRequiredMixin`, що вимагає авторизації для виконання публікації, і клас `CreateView`, що є основою для представлень створення об'єктів.

Аналогічно побудовано наступні класи наслідуванням класу `CreateView`.

Клас `AddPropositionView` забезпечує створення пропозиції про придбання або огляд автомобіля.

Клас `AddCarViewView` забезпечує додавання інформації про перегляд оголошення про продаж вживаного автомобіля користувачем.

Клас `AddPriceView` забезпечує внесення запису про вартість автомобіля.

Клас `AddChatView` забезпечує додавання повідомлення в чат між власником автомобіля і покупцем.

Клас `AddModelReviewView` забезпечує створення відгуку про модель автомобіля.

Клас `AddCarReviewView` забезпечує додавання відгуку про автомобіль.

Усі ці класи потребують попередньої авторизації, тому наслідують клас `LoginRequiredMixin`.

Клас `CarEditView` забезпечує редагування оголошення про продаж автомобіля, наслідуючи клас `LoginRequiredMixin`, що вимагає авторизації для виконання редагування, і клас `UpdateView`, що є основою для представлень редагування об'єктів.

Клас `PropositionEditView` забезпечує редагування пропозиції про купівлю або огляд автомобіля і реалізується шляхом наслідування класів `UpdateView` і `LoginRequiredMixin`.

Клас `UserCarsView` забезпечує створення представлення перегляду власних оголошень про продаж автомобілів продавцем. Оскільки дана функція потребує авторизації для визначення продавця, то клас наслідується від `LoginRequiredMixin`. Також клас наслідується від стандартного класу представлення перегляду переліку оголошень `ListView`. Об'єкти оголошень відсортовуються за датою створення оголошення, починаючи з найновіших.

Клас `CarsRatingView` не передбачає необхідності авторизації, тому наслідується тільки від класу `ListView`, створюючи перелік відсортованих за рейтингом оголошень про продаж автомобілів. Перелік складається зі 100 оголошень з найвищим рейтингом. До рейтингу входять тільки активні на даний момент оголошення, тобто оголошення, за якими автомобілі було продано, та скасовані оголошення не входять до рейтингу.

Клас `MostViewedCarsView` створює перелік оголошень з найбільшим числом переглядів покупцями, наслідуючи клас `ListView`. Усі неактивні оголошення з переліку виключаються. Оголошення сортуються за кількістю переглядів, повертається перелік зі 100 оголошень з найбільшою сумарною кількістю переглядів.

Клас `MostLikedCarsView` створює перелік оголошень, які найчастіше вносились користувачами до закладок. Усі неактивні оголошення з переліку виключаються. Оголошення сортуються за кількістю випадків внесення у закладки, повертається перелік зі 100 оголошень з найбільшою сумарною кількістю закладок.

Клас `MostPropositionsCarsView` створює перелік оголошень, які викликають найбільшу активність покупців. Це оголошення, за якими було винесено найбільшу кількість пропозицій про купівлю або огляд. Усі неактивні оголошення з переліку виключаються. Оголошення сортуються за

кількістю висунутих пропозицій, повертається перелік зі 100 оголошень з найбільшою сумарною кількістю пропозицій.

Клас `NewestCarsView` створює перелік найновіших оголошень про продаж автомобілів. Усі неактивні оголошення з переліку виключаються. Оголошення сортуються за датою публікації, повертається перелік зі 100 оголошеннями з найпізнішою датою публікації.

Клас `CompareCarsView` забезпечує підтримку порівняння оголошень. Клас реалізовано наслідуванням від класу `ListView`. Клас заповнює даними перелік, вносячи інформацію про автомобілі з переліку порівняння.

Клас `SearchCarView` виконує підтримку пошуку вживаних автомобілів для придбання. Клас створено наслідуванням класу `ListView`. Параметри пошуку передаються через параметри, значення яких отримуються через GET-запит:

- `producer` визначає виробника автомобіля (у такому випадку конкретна модель не вказується);
- `carmodel` визначає модель автомобіля;
- `transmission` визначає тип трансмісії автомобіля;
- `doorsmin` визначає мінімальну кількість дверей, яку повинен мати автомобіль;
- `doorsmax` визначає максимальну кількість дверей, яку повинен мати автомобіль;
- `placesmin` визначає мінімальну кількість місць у автомобілі;
- `placesmax` визначає максимальну кількість місць у автомобілі;
- `pathmin` визначає мінімальний пробіг автомобіля;
- `pathmax` визначає максимальний пробіг автомобіля;
- `color` визначає колір автомобіля;
- `scoremin` визначає мінімальну оцінку, яку повинно мати оголошення;
- `scoremax` визначає максимальну оцінку, яку повинно мати оголошення;
- `cartype` визначає тип кузова автомобіля;

- producedmin визначає найпізнішу дату виробництва автомобіля;
- producedmax визначає найранішу дату виробництва автомобіля;
- location визначає місце, де знаходиться вживаний автомобіль з оголошення;
- package визначає комплектацію автомобіля;
- createdmin визначає найпізнішу дату публікації оголошення;
- createdmax визначає найранішу дату публікації оголошення;
- drive визначає привід автомобіля.

Якщо будь-яке з цих значень не задано, то воно не враховується в параметрах пошуку. Отримані результати пошуку, що відповідають заданим параметрам, сортуються за середньою оцінкою за зменшенням.

Також на основі класу ListView організовано наступні класи.

Клас AdvancedSearchView виконує підтримку пошуку вживаних автомобілів за уподобаннями користувача на основі використання розробленого в роботі методу.

Клас LikesView створює перелік закладок оголошень про продаж вживаних автомобілів, створених користувачем.

Клас UserReviewsView створює перелік відгуків користувача на моделі автомобілів, їх оголошення та власників автомобілів.

Клас OwnerReviewsView створює перелік відгуків на користувача, який виступає власником автомобілів.

Клас UserPropositionsView створює перелік пропозицій, висунутих заданим користувачем щодо придбання і огляду автомобілів.

Клас PropositionsView створює перелік пропозицій про придбання заданого автомобіля і огляд даного автомобіля.

Клас ChatView створює перелік повідомлень для формування чату між власником автомобіля та покупцем.

Клас LikeView побудований на основі класу View для підтримки створення закладки на оголошення про продаж автомобіля.

Клас UnlikeView побудований на основі класу View для підтримки вилучення закладки на оголошення про продаж автомобіля.

Створені шаблони реалізовані на основі відкритого шаблону [22].

4.3 Висновки за розділом 4

Виконано програмну реалізацію. Розроблена програма має чітко визначену схему свого функціонування, яку було представлено і описано. Розроблені класи і шаблони вебзастосунку описано з представленням використаних під час їх розробки рішень.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМИ

5.1 Призначення програми

Програма призначена для організації процесу продажу автомобілів і комунікації в даному процесі між продавцями та покупцями. Задля цього програма призначена для підтримки роботи з оголошеннями, пошуку вживаних автомобілів та спілкування між продавцями та покупцями.

5.2 Умови виконання програми

Оскільки подібні системи передбачають роботи з великою кількістю користувачів і обробку великої кількості запитів, то технічні засоби для підтримки роботи вебсервера повинні включати наступний мінімальний набір:

- чотириядерний процесор з частотою від 2 ГГц;
- обсяг оперативної пам'яті від 8 Гб;
- 2 Гб вільного простору на жорсткому диску.

Для експлуатації ПЗ окрім безпосередньо наявності самої програмної системи потрібно також встановити на сервер систему керування базами даних MySQL.

5.3 Робота з програмою

Робота з програмою реалізується в двох режимах: адміністратора і користувача.

Робота адміністратора виконується через адміністративну панель і допомагає вносити дані для роботи програми. Це дані, що стосуються моделей автомобілів, типа палива тощо.

Робота звичайного користувача реалізується через вебінтерфейс (рис. 5.1), логіка роботи якого була описана в розділі 4.

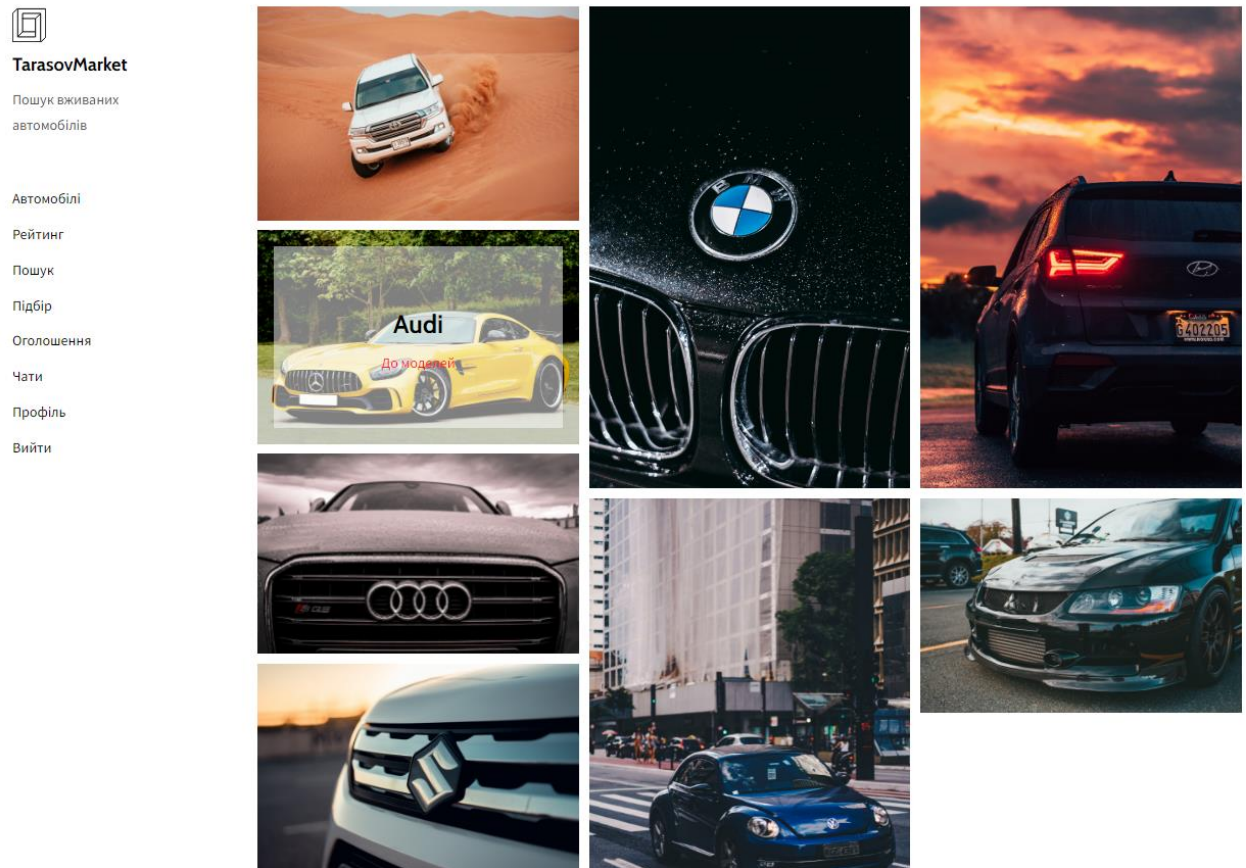


Рисунок 5.1 – Відображення виробників автомобілів з можливістю переходу до моделей автомобілів

Кожен авторизований користувач працює зі стандартним меню програми, що забезпечує переміщення між основними групами функцій, які надаються програмою.

Меню «Автомобілі» призначено для перегляду даних про моделі автомобілів загалом. Початково відкривається сторінка з переліком виробників. Кожен виробник представлений фотографією першої своєї моделі автомобіля. Якщо навести курсор на зображення, то відображається назва виробника і пропонується перейти до сторінки цієї моделі, де узагальнюється інформація про цю модель, зокрема середня вартість

автомобілів цієї моделі, що обчислюється за окремими оголошеннями про продаж автомобілів.

Меню «Рейтинг» надає можливість переглянути 100 найперших автомобілів у рейтингу за одним з критеріїв, що включають середню оцінку, кількість закладок за цією моделлю, кількість переглядів, час створення оголошення тощо.

За кожним оголошенням можна відкрити сторінку деталізації (рис. 5.2).



TarasovMarket

Пошук вживаних автомобілів

- Автомобілі
- Рейтинг
- Пошук
- Підбір
- Оголошення
- Чати
- Профіль
- Вийти

Mitsubishi Lancer 2018



В закладки

Шукати подібні

Залишити відгук

Написати власнику

Висунути пропозицію

+

Порівняти

Стан оголошення: активне.

Розташування: Запоріжжя

Рік виготовлення: 2018

Перше використання: 05.02.2019

Вартість: 560000 грн.

[Історія зміни вартості](#)

Оцінка: 4.8

Реєстраційний номер: AA1111AA

VIN: GFJQKZ2U4B6392741

Технічні характеристики

Двигун: Бензиновий, Express 1.8 л 240 кін.сил

Пройдений шлях: 41000 км

Тип кузова: седан

Привід: передній

Трансмісія: механічна

Витрата пального в місті: 10.3 л / 100 км

Витрата пального на трасі: 6.2 л / 100 км

Витрата пального в змішаному циклі: 8.1 л / 100 км

Загальні характеристики

Дверей: 4

Місць: 5

Колір: чорний

Комплектація: Comfort

Особливості

Без ДТП

Проблеми

Має декілька невеликих поляпин на лакофарбовому покритті.

Власник

Ігор Надольський

Оцінка: 4.5

Відгуки

Андрій Молочний

08.12.2021 20:11

Машина в досить гарному стані. Подряпин більше ніж вказано. Вартість можливо трохи завищена.

Євген Силов

08.12.2021 18:49

Рисунок 5.2 – Оголошення про продаж вживаного автомобіля

На сторінці представляється вся наявна інформація за оголошенням. Звідси можна перейти до перегляду інших оголошень за цією моделлю

автомобіля. Для цього потрібно вгорі натиснути в назві оголошення на назву моделі.

За оголошенням можна переглянути зокрема перелік відгуків, середню обчислену оцінку оголошення, інформацію про власника, в тому числі його оцінку (натискаючи на ім'я власника, можна перейти до перегляду даних про власника, включаючи самі відгуки).

За допомогою кнопки «В закладки» можна внести оголошення до закладок користувача.

За допомогою кнопки «Залишити відгук» можна створити відгук про оголошення.

За допомогою кнопки «Висунути пропозицію» можна створити власну пропозицію про купівлю або перегляд автомобіля. За всіх цих варіантів роботи через кнопки відкривається окрема сторінка. Також на цій сторінці доступний перелік останніх пропозицій про купівлю автомобіля.

За допомогою кнопки «Шукати подібні» можна перейти до пошук автомобілів, подібних до відкритого. Пошук реалізується за допомогою запропонованого методу.

За допомогою кнопки «Написати власнику» можна перейти до чату з власником автомобіля, де можна відправити йому повідомлення.

За допомогою кнопки «+» можна додати автомобіль до переліку порівняння. Відкрити сторінку порівняння можна за допомогою кнопки «Порівняти».

Меню «Пошук» дозволяє виконати пошук за розширеною кількістю параметрів. Спочатку всі параметри мають довільні значення.

Можна задати наступні параметри (рис. 5.3):

- виробника автомобіля, а після цього з наступного переліку – модель автомобіля;
- тип трансмісії автомобіля;
- мінімальну і максимальну кількість дверей, яку повинен мати автомобіль;

- мінімальну і максимальну кількість місць у автомобілі;
- мінімальний і максимальний пробіг автомобіля;
- колір автомобіля;
- мінімальну і максимальну оцінку, яку повинно мати оголошення;
- тип кузова автомобіля;
- найранішу і найпізнішу дату виробництва автомобіля (рік);
- місце, де знаходиться вживаний автомобіль з оголошення;
- комплектацію автомобіля;
- найпізнішу і найранішу дату публікації оголошення;
- привід автомобіля.



TarasovMarket

Пошук вживаних автомобілів

Автомобілі

Рейтинг

Пошук

Підбір

Оголошення

Чати

Профіль

Вийти

Виробник

Toyota

Модель

Camry

Двигун

Будь-який

Привід

Передній

Трансмісія

Механічна

Колір

Будь-який

Дверей від

2

Дверей до

5

Місць від

2

Місць до

5

Пробіг від

0

Пробіг до

20000

Тип кузова

Седан

Місто

Запоріжжя

Паливо

Бензин

Комплектація

Будь-яка

Оцінка від

4

Оцінка до

5

Вироблено від

2016

Вироблено до

2021

Публікація оголошення

дд.мм.rrrr

До

дд.мм.rrrr

Шукати



Toyota Camry 2020

Детальніше



Toyota Camry 2017

Детальніше



Toyota Camry 2015

Детальніше

Рисунок 5.3 – Пошук вживаних автомобілів

Результати пошуку відображаються на сторінці нижче за натискання на кнопку «Шукати».

Меню «Підбір» (рис. 5.4) дозволяє виконати підбір автомобіля на основі представленого в роботі метода, отримуючи спочатку відповіді користувача на запитання. В кінці потрібно натиснути на кнопку «Знайти». Тоді відкривається сторінка з результатами пошуку, де відображаються знайдені автомобілі, що відповідають вимогам користувача.



TarasovMarket

Пошук вживаних
автомобілів

Автомобілі

Рейтинг

Пошук

Підбір

Оголошення

Чати

Профіль

Вийти

Підбір автомобіля

Будь ласка, заповніть опитувальник повністю для покращення отриманих результатів.

1. На скільки ви любите впливати на керування автомобілем?

- ☒ суцільно і весь час
- ☐ люблю повний контроль, але іноді треба і відпочити
- ☐ це не має значення
- ☐ для мене головне - дістатися заданого місця

2. На скільки критичний для вас зовнішній стан автомобіля?

- ☐ все має бути ідеально
- ☒ важливий
- ☐ хотілось бі покраще
- ☐ готовий до ремонту
- ☐ це для мене не має значення

3. Як ви збираєтесь використовувати автомобіль?

- ☐ тільки в місті
- ☐ місто та поїздки по країні
- ☒ місто та глибинка
- ☐ місто, глибинка та тривалі подорожі

Рисунок 5.4 – Підбір автомобіля

5.4 Експериментальне дослідження

Експериментальне дослідження методу пошуку вживаних автомобілів було реалізовано щодо організації процесу кластеризації автомобілів. Під час експериментального дослідження потрібно було дослідити те, на скільки якісно було виконано кластеризацію даних, тобто на скільки згрупованими є дані.

Під час експериментального дослідження виконувався пошук автомобілів, подібних до заданого.

Дослідження проводилось на основі вибірки даних [23]. Ґрунтуючись на цій вибірці, було обрано різні значення кількості кластерів k і досліджено якість кластеризації для кожного значення k (табл. 5.1).

Якість кластеризації було обчислено шляхом використання індекса Данна, який дозволяє оцінити відносне співвідношення між відстанню екземплярів з різних кластерів і відстанню екземплярів з одного кластера.

Таблиця 5.1 – Результати дослідження якості кластеризації за методом пошуку вживаних автомобілів

Значення параметра k (кількість кластерів)	Якість кластеризації
3	0,5276
5	0,6119
7	0,6594
9	0,6803
12	0,7152
15	0,7496
17	0,7815
20	0,8091
25	0,7653

Отримані результати дослідження продемонстрували достатньо якісні результати, при цьому значення k , що дорівнює 20, призвело до найвищого рівня якості кластеризації.

5.5 Висновки за розділом 5

Розроблена програма призначена для організації процесу продажу автомобілів і комунікації в даному процесі між продавцями та покупцями. Розглянуті засоби програми підтримують роботу з оголошеннями, пошук вживаних автомобілів та спілкування між продавцями і покупцями. Проведено дослідження впливу кількості кластерів на якість кластеризації даних.

ВИСНОВКИ

Метою кваліфікаційної роботи було розробити програмне забезпечення пошуку вживаних автомобілів для забезпечення створення відповідного вимогам покупця портфоліо пропозицій про придбання вживаних автомобілів для подальшого вибору найкращого, що було досягнуто за результатами роботи.

Виконано аналіз предметної області. Розглянуто сучасний стан проблеми продажу вживаних автомобілів, виконано аналіз існуючих програмних засобів пошуку вживаних автомобілів з виконанням порівняння цих засобів з програмою, що розроблялась. Особливостями виконуваної розробки є пошук за оголошень за VIN-номерами та надання альтернативних сценаріїв пошуку.

Виконано дослідження методів пошуку вживаних автомобілів. Для цього було проаналізовано принципи роботи методу кластеризації k-середніх і створено власний метод вирішення даної проблеми. Наукова новизна роботи полягає в запропонованому методі пошуку вживаних автомобілів, який ґрунтується на застосуванні методу k-середніх для кластеризації пропозицій, що дозволяє на основі опитувальника або обраної моделі автомобіля обирати моделі автомобілів і оголошення, які відповідають вимогам покупця.

Виконано проєктування програми за допомогою мови програмування Python, системи керування базою даних MySQL, вебфреймворку Django. Користуючись цими рішеннями, визначено структуру програми та описано структуру бази даних.

Виконано програмну реалізацію. Розроблена програма призначена для організації процесу продажу автомобілів і комунікації в даному процесі між продавцями та покупцями. Задля цього програма надає засоби для підтримки роботи з оголошеннями, пошуку вживаних автомобілів як на основі пошуку

за параметрами, так і на основі заповнення опитувальника або вибору подібної моделі автомобіля та спілкування між продавцями і покупцями.

Усі поставлені завдання на роботу виконано.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Used Vehicle Market Size, Trends. Forecast Report, 2021-2021 [Electronic resource]. – Access mode : <https://www.fortunebusinessinsights.com/used-vehicle-market-104665>.
2. Global Used Vehicle Market Size [2021-2028] to Reach USD 1,969.72 Billion with a CAGR of 1.79% [Electronic resource]. – Access mode : <https://finance.yahoo.com/news/global-used-vehicle-market-size-084200676.html>.
3. Stenquist, P. Tips for Buying a Used Car [Electronic resource] / P. Stenquist. – Access mode : <https://www.nytimes.com/2021/12/09/business/used-car-buying-tips.html>.
4. AUTO.RIA – Автобазар №1. Купити і продати авто легко як ніколи [Електрон. ресурс]. – Режим доступу : <https://auto.ria.com/uk/>.
5. New Cars, Used Cars, Car Dealers, Prices & Reviews. Cars.com [Electronic resource]. – Access mode : <https://www.cars.com/>.
6. The Ultimate Guide to Cars.com for Dealerships – Podium [Electronic resource]. – Access mode : <https://www.podium.com/article/cars-com-for-dealers/>.
7. Mishra, S. Unsupervised Learning and Data Clustering [Electronic resource] / S. Mishra. – Access mode : <https://towardsdatascience.com/unsupervised-learning-and-data-clustering-eeecb78b422a>.
8. Kaushik, S. An Introduction to Clustering and different methods of clustering [Electronic resource] / S. Kaushik. – Access mode : <https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/>.
9. Exploring Clustering Algorithms [Electronic resource] : Explanation and Use Cases. – Access mode : <https://neptune.ai/blog/clustering-algorithms>.
10. Gan, G. Data Clustering [Text] : Theory, Algorithms, and Applications / Guojun Gan, Chaoqun Ma, Jianhong Wu. – Philadelphia : SAIM, 2020. – 406 p.
11. Nasraoui, O. Clustering Methods for Big Data Analytics [Text] :

Techniques, Toolboxes and Applications / Olfa Nasraoui, Chiheb-Eddine Ben N'Cir. – Berlin : Springer, 2019. – 196 p.

12. Bedre, R. k-means clustering in Python with example [Electronic resource] / R. Bedre. – Access mode : <https://www.reneshbedre.com/blog/kmeans-clustering-python.html>.

13. Кластеризация [Электрон. ресурс] : алгоритмы k-means и c-means. – Режим доступа : <https://habr.com/ru/post/67078/>.

14. Garbade, M. J. Understanding K-means Clustering in Machine Learning [Electronic resource] / M. J. Garbade. – Access mode : <https://towardsdatascience.com/understanding-k-means-clustering-in-machine-learning-6a6e67336aa1>.

15. Chapple, M. What Is K-Means Clustering? [Electronic resource] / M. Chapple. – Access mode : <https://www.lifewire.com/k-means-clustering-1019648>.

16. Barry, P. Head First Python [Text] : A Brain-Friendly Guide / P. Barry. – Sebastopol, CA : O'Reilly Media, 2016. – 624 p.

17. Бизли, Д. М. Python. Подробный справочник, 4-е издание [Текст] / Д. М. Бизли. – СПб. : Символ-Плюс, 2010. – 864 с.

18. Саммерфилд, М. Программирование на Python 3. Подробное руководство [Текст] / М. Саммерфилд. – СПб. : Символ-Плюс, 2009. – 608 с.

19. Advantages and Disadvantages of Django Architecture [Electronic resource]. – Access mode : <https://www.nexsoftsys.com/articles/pros-and-cons-of-django-architecture.html>.

20. The web framework for perfectionists with deadlines. Django [Electronic resource]. – Access mode : <https://www.djangoproject.com/>.

21. Руководство по веб-фреймворку Django [Электрон. ресурс]. – Режим доступа : <https://metanit.com/python/django/>.

22. Savory Free Website Template. Free CSS Templates. Free CSS [Electronic resource]. – Access mode : <https://www.free-css.com/free-css-templates/page273/savory>.

23. Used Cars Dataset. Vehicles listings from Craigslist.org [Electronic resource]. – Access mode : <https://www.kaggle.com/austinreese/craigslist-carstrucks-data>.

ДОДАТОК А
Технічне завдання

Вступ

Програмне забезпечення пошуку вживаних автомобілів не може виступати тільки сховищем оголошень, адже в такому випадку дуже важко віднайти в масиві створених різними користувачами оголошень те, що може по-сравжньому зацікавити покупця.

Задля вирішення цієї проблеми необхідно розробити програмне забезпечення, що окрім організації сховища буде реалізовувати інтелектуальні засоби пошуку вживаних автомобілів.

A.1 Підстави для розробки

Підставою для розробки є завдання на дипломну кваліфікаційну роботу магістра, визначене за темою «Дослідження та програмна реалізація методів пошуку вживаних автомобілів», яку затверджено наказом № 435 від 5 листопада 2021 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма призначена для організації процесу продажу автомобілів і комунікації в даному процесі між продавцями та покупцями. Задля цього програма надає засоби для підтримки роботи з оголошеннями, пошуку вживаних автомобілів різними способами та спілкування між продавцями та покупцями.

Програма може експлуатуватися продавцями вживаних автомобілів і їх покупцями.

A.3 Основні вимоги до програми

A.3.1 Вимоги до функціональних характеристик

Програма має забезпечувати для користувача виконання таких функцій:

- створення відгуку про автомобіль за оголошенням про продаж;
- створення відгуку про власника автомобіля;
- створення відгуку про модель автомобіля;
- пошук оголошень про продаж автомобілів;
- підбір автомобіля за вимогами покупця;
- публікація оголошення про продаж автомобіля;
- перегляд оголошень власника;
- перегляд оголошення про продаж автомобіля;
- висування пропозиції про продаж автомобіля або про огляд;
- приймання пропозиції про купівлю автомобіля або про огляд;
- скасування пропозиції;
- керування власними оголошеннями, у тому числі зняття автомобіля з продажу;
- перегляд візитної сторінки моделі автомобіля;
- створення закладки на оголошенні про продаж автомобіля;
- перегляд створених закладок;
- перегляд профілю користувача;
- перегляд історії вартості автомобіля;
- перегляд чату між власником і покупцем;
- відправлення повідомлення в чат;
- порівняння оголошень;
- перегляд рейтингу оголошень;
- перегляд власних пропозицій;
- перегляд рейтингу оголошень за середнім балом;
- перегляд рейтингу оголошень за кількістю переглядів;

- перегляд рейтингу оголошень за кількістю створених закладок;
- перегляд рейтингу оголошень за часом створення;
- перегляд рейтингу моделей автомобілів.

A.3.2 Вимоги до складу та параметрів технічних засобів

Технічні засоби для підтримки роботи вебсервера повинні включати наступний мінімальний набір:

- чотириядерний процесор з частотою від 2 ГГц;
- обсяг оперативної пам'яті від 8 Гб;
- 2 Гб вільного простору на жорсткому диску.

A.3.3 Умови експлуатації

Для експлуатації програмного забезпечення окрім безпосередньо наявності самої програмної системи потрібно також встановити на сервер систему керування базами даних MySQL.

A.3.4 Вимоги до транспортування та збереження

Збереження програми має реалізовуватися шляхом розташування її на виділеному для цього сервері.

A.3.5 Вимоги до програмної документації

До складу програмної документації входять технічне завдання, керівництво оператора, що вноситься до складу розділу 5 пояснювальної записки, текст програми.

А.4 Порядок контролю та приймання

Календарний план є основою для контролю виконання роботи, приймання роботи здійснюється під час її захисту перед екзаменаційною комісією.

ДОДАТОК Б
Текст програми

Б.1 Текст файла models.py

```

from django.contrib.auth.models import AbstractUser
from django.db import models
from django.urls import reverse
from .managers import CustomUserManager


class CarModel (models.Model):
    title = models.CharField(max_length=100)
    score = models.FloatField()
    min_price = models.IntegerField()
    max_price = models.IntegerField()
    average_price = models.IntegerField()
    producer = models.ForeignKey (Producer,
on_delete=models.CASCADE)

    def __str__(self):
        return self.producer.title + ' ' + self.title

    def get_absolute_url(self):
        return reverse('carmodel', args=[self.id])


class Car (models.Model):
    engine = models.ForeignKey (Engine,
on_delete=models.CASCADE)
    carmodel = models.ForeignKey (CarModel,
on_delete=models.CASCADE)
    transmission = models.SmallIntegerField()
    doors = models.SmallIntegerField()
    places = models.SmallIntegerField()
    path = models.IntegerField()
    color = models.ForeignKey (Color, on_delete=models.CASCADE)
    score = models.FloatField()
    viewed = models.IntegerField()
    liked = models.IntegerField()
    propositions = models.IntegerField()
    user = models.ForeignKey (User, on_delete=models.CASCADE)

```

```

        cartype = models.ForeignKey (CarType,
on_delete=models.CASCADE)
        produced = models.DateField()
        used = models.DateField()
        location = models.ForeignKey (Location,
on_delete=models.CASCADE)
        regnumber = models.CharField(max_length=8)
        vin = models.CharField(max_length=17)
        package = models.ForeignKey (Package,
on_delete=models.CASCADE)
        created = models.DateTimeField()
        problem = models.CharField(max_length=255)
        activity = models.SmallIntegerField()
        drive = models.SmallIntegerField()
        cityfuel = models.DecimalField(max_digits=3,
decimal_places=1)
        routefuel = models.DecimalField(max_digits=3,
decimal_places=1)
        mixfuel = models.DecimalField(max_digits=3,
decimal_places=1)
        states = models.ManyToManyField(State)
        markers = models.ManyToManyField(Marker)

    def __str__(self):
        return self.model + ' ' + self.produced

    def get_absolute_url(self):
        return reverse('car', args=[self.id])

    def get_transmission_type(self):
        if self.transmission == 0:
            return 'механічна'
        elif self.transmission == 1:
            return 'роботизована'
        elif self.transmission == 2:
            return 'варіаторна'

    def get_drive_type(self):
        if self.drive == 0:
            return 'передній'
        elif self.drive == 1:

```

```

        return 'задній'
    elif self.drive == 2:
        return 'повний'

    def get_activity_type(self):
        if self.activity == 0:
            return 'активне'
        elif self.activity == 1:
            return 'зняте'
        elif self.activity == 2:
            return 'продано'

class Color (models.Model):
    title = models.CharField(max_length=50)
    code = models.CharField(max_length=7)

    def __str__(self):
        return self.title

class State (models.Model):
    title = models.CharField(max_length=100)
    description = models.CharField(max_length=255)
    type = models.SmallIntegerField()

    def __str__(self):
        return self.title

    def get_state_type(self):
        if self.type == 0:
            return 'повністю непошкоджений'
        elif self.type == 1:
            return 'професійно відремонтований'
        elif self.type == 2:
            return 'потребує ремонту'

class User(AbstractUser):
    name = models.CharField(max_length=100)
    phone = models.CharField(max_length=13)

```

```

score = models.FloatField()
registered = models.DateTimeField()
visited = models.DateTimeField()

objects = CustomUserManager()

def __str__(self):
    return self.name

class Producer (models.Model):
    title = models.CharField(max_length=100)

    def __str__(self):
        return self.title

class EngineType (models.Model):
    title = models.CharField(max_length=100)

    def __str__(self):
        return self.title

class Engine (models.Model):
    enginetype = models.ForeignKey (EngineType,
on_delete=models.CASCADE)
    horsepower = models.IntegerField()
    title = models.CharField(max_length=100)
    literes = models.FloatField()
    cylinders = models.FloatField()
    cityfuel = models.DecimalField(max_digits=3,
decimal_places=1)
    routefuel = models.DecimalField(max_digits=3,
decimal_places=1)
    mixfuel = models.DecimalField(max_digits=3,
decimal_places=1)

    def __str__(self):
        return self.title

```

```

class Package (models.Model):
    title = models.CharField(max_length=100)
    description = models.CharField(max_length=255)

    def __str__(self):
        return self.title

class Fuel (models.Model):
    title = models.CharField(max_length=50)

    def __str__(self):
        return self.title

class Location (models.Model):
    title = models.CharField(max_length=50)

    def __str__(self):
        return self.title

class Marker (models.Model):
    title = models.CharField(max_length=50)
    type = models.IntegerField()

    def __str__(self):
        return self.title

    def get_marker_type(self):
        if self.type == 0:
            return 'безпека'
        elif self.type == 1:
            return 'стан'
        elif self.type == 2:
            return 'комфорт'

class CarType (models.Model):
    title = models.CharField(max_length=50)

    def __str__(self):

```

```
return self.title
```

```
class Proposition (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    number = models.IntegerField()
    state = models.BooleanField()
    created = models.DateTimeField()
    description = models.TextField()
    meeting = models.DateTimeField()
    place = models.CharField(max_length=100)

    def __str__(self):
        return self.user.name + ' ' + self.number

    def get_state(self):
        if self.state:
            return 'активна'
        else:
            return 'неактивна'

    class Meta:
        ordering = ["-created"]

class Photo (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    file = models.CharField(max_length=255)

class Like (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    liked = models.DateTimeField()
    message = models.TextField()

    class Meta:
        ordering = ["-liked"]
```

```

class View (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    viewed = models.DateTimeField()

class Price (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    number = models.IntegerField()
    created = models.DateTimeField()

    def __str__(self):
        return self.created + ' ' + self.user.name

    class Meta:
        ordering = ["-created"]

class Chat (models.Model):
    user1 = models.ForeignKey (User, on_delete=models.CASCADE)
    user2 = models.ForeignKey (User, on_delete=models.CASCADE)
    message = models.TextField()
    created = models.DateTimeField()

    class Meta:
        ordering = ["-created"]

class ModelReview (models.Model):
    carmodel = models.ForeignKey (CarModel,
on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    message = models.TextField()
    created = models.DateTimeField()
    score = models.FloatField()

    class Meta:
        ordering = ["-created"]

```

```

class CarReview (models.Model):
    car = models.ForeignKey (Car, on_delete=models.CASCADE)
    user = models.ForeignKey (User, on_delete=models.CASCADE)
    message = models.TextField()
    created = models.DateTimeField()
    score = models.FloatField()

    class Meta:
        ordering = ["-created"]

class OwnerReview (models.Model):
    user1 = models.ForeignKey (User, on_delete=models.CASCADE)
    user2 = models.ForeignKey (User, on_delete=models.CASCADE)
    message = models.TextField()
    created = models.DateTimeField()
    score = models.FloatField()

    class Meta:
        ordering = ["-created"]

```

Б.2 Текст файла /views/carview.py

```

from django.views.generic.detail import DetailView
from django.contrib.auth.mixins import LoginRequiredMixin
from django.views.generic.edit import CreateView, UpdateView
import tarasov_market.models
import datetime

```

```

class CarView (DetailView):

    model = Car
    template_name = 'cardetail.html'

    def get_context_data(self, **kwargs):
        carcont = super().get_context_data(**kwargs)
        carinfo = Car.objects.get(id=self.kwargs['pk'])
        carcont['car'] = carinfo

```

```

        carreviews = CarReview.objects.filter(pk =
self.kwargs['pk'])
        carcont['reviews'] = carreviews
        photos = Photo.objects.filter(car__id =
self.kwargs['pk'])
        carcont['photos'] = photos
        propositions = Proposition.objects.filter(car__id =
self.kwargs['pk'])
        carcont['propositions'] = propositions
        prices = Price.objects.filter(car__id =
self.kwargs['pk'])
        carcont['prices'] = prices
        currprice = prices[0].number
        carcont['currprice'] = currprice
        return carcont

```

```

class CarNewView(LoginRequiredMixin, CreateView):

```

```

    template_name = 'carnew.html'
    model = Car

```

```

    fields = [
        'engine',
        'carmodel',
        'transmission',
        'doors',
        'places',
        'path',
        'color',
        'cartype',
        'produced',
        'location',
        'regnumber',
        'vin',
        'package',
        'problem',
        'drive',
        'cityfuel',
        'routefuel',
        'mixfuel',

```

```

        'states',
        'markers'
    ]

    login_url = 'authview'
    success_url = 'carview'

    def form_valid(self, form):
        form.instance.user = self.request.user
        form.score = 0
        form.viewed = 0
        form.liked = 0
        form.propositions = 0
        form.created = datetime.datetime.now()
        form.activity = 0
        return super().form_valid(form)

```

```
class CarEditView (LoginRequiredMixin, UpdateView):
```

```

    template_name = 'caredit.html'
    model = Car

```

```

    fields = [
        'engine',
        'carmodel',
        'transmission',
        'doors',
        'places',
        'path',
        'color',
        'cartype',
        'produced',
        'location',
        'regnumber',
        'vin',
        'package',
        'problem',
        'drive',
        'cityfuel',
        'routefuel',
    ]

```

```

        'mixfuel',
        'states',
        'markers',
        'activity'
    ]

    login_url = 'authview'
    success_url = 'carview'

class UserCarsView (LoginRequiredMixin, ListView):

    model = Car
    template_name = 'usercars.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(UserCarsView, self).get_queryset(*args,
**kwargs)

        owner = self.request.user
        cars = Car.objects.filter(user = owner)
        cars = cars.order_by('-created')
        carcont['cars'] = cars
        return carcont

class VINCarsView (LoginRequiredMixin, ListView):

    model = Car
    template_name = 'vincars.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(VINCarsView, self).get_queryset(*args,
**kwargs)

        vinnumber = self.request.GET.get('vin')
        cars = Car.objects.filter(vin = vinnumber)
        cars = cars.order_by('-created')
        carcont['cars'] = cars
        return carcont

```

```

class CarsRatingView (ListView):

    model = Car
    template_name = 'carrating.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(CarsRatingView,
self).get_queryset(*args, **kwargs)
        cars = Car.objects.filter(activity = 0)
        cars = cars.order_by('-score')
        carcont['cars'] = cars[0:100]
        return carcont


class MostViewedCarsView (ListView):

    model = Car
    template_name = 'carmostviewed.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(MostViewedCarsView,
self).get_queryset(*args, **kwargs)
        cars = Car.objects.filter(activity = 0)
        cars = cars.order_by('-viewed')
        carcont['cars'] = cars[0:100]
        return carcont


class MostLikedCarsView (ListView):

    model = Car
    template_name = 'carmostliked.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(MostLikedCarsView,
self).get_queryset(*args, **kwargs)
        cars = Car.objects.filter(activity = 0)
        cars = cars.order_by('-liked')
        carcont['cars'] = cars[0:100]
        return carcont

```

```

class MostPropositionsCarsView (ListView):

    model = Car
    template_name = 'carmostpositioned.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(MostPropositionsCarsView,
self).get_queryset(*args, **kwargs)
        cars = Car.objects.filter(activity = 0)
        cars = cars.order_by('-propositions')
        carcont['cars'] = cars[0:100]
        return carcont


class NewestCarsView (ListView):

    model = Car
    template_name = 'carnewest.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(NewestCarsView,
self).get_queryset(*args, **kwargs)
        cars = Car.objects.filter(activity = 0)
        cars = cars.order_by('-created')
        carcont['cars'] = cars[0:100]
        return carcont


class CompareCarsView (ListView):

    model = Car
    template_name = 'carcompare.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(CompareCarsView,
self).get_queryset(*args, **kwargs)
        carcomparison = self.request.session.get('carcomp', [])
        cars = []
        for car in carcomparison:
            car = Car.objects.get(id = car)
            cars.append({'car': car})

```

```

carcont['cars'] = cars
return carcont

```

```

class SearchCarView (ListView):

    model = Car
    template_name = 'search.html'

    def get_queryset(self, *args, **kwargs):
        carcont = super(SearchCarView, self).get_queryset(*args,
**kwargs)

        cars = Car.objects.filter(activity = 0)
        pproducer = self.request.GET.get('producer')
        if pproducer:
            cars.filter(carmodel_producer = pproducer)
        pcarmodel = self.request.GET.get('carmodel')
        if pcarmodel:
            cars.filter(carmodel = pcarmodel)
        ptransmission = self.request.GET.get('transmission')
        if ptransmission:
            cars.filter(transmission = ptransmission)
        pdoorsmin = self.request.GET.get('doorsmin')
        if pdoorsmin:
            cars.filter(doors__gte = pdoorsmin)
        pdoorsmax = self.request.GET.get('doorsmax')
        if pdoorsmax:
            cars.filter(doors__lte = pdoorsmax)
        pplacesmin = self.request.GET.get('placesmin')
        if pplacesmin:
            cars.filter(places__gte = pplacesmin)
        pplacesmax = self.request.GET.get('placesmax')
        if pplacesmax:
            cars.filter(places__lte = placesmax)
        ppathmin = self.request.GET.get('pathmin')
        if pppathmin:
            cars.filter(path__gte = ppathmin)
        ppathmax = self.request.GET.get('pathmax')
        if ppathmax:
            cars.filter(path__lte = pathmax)
        pcolor = self.request.GET.get('color')

```

```

if pcolor:
    cars.filter(color = pcolor)
pscoremin = self.request.GET.get('scoremin')
if pscoremin:
    cars.filter(score__gte = pscoremin)
pscoremax = self.request.GET.get('scoremax')
if pscoremax:
    cars.filter(score__lte = pscoremax)
pcartype = self.request.GET.get('cartype')
if pcartype:
    cars.filter(cartype = pcartype)
pproducedmin = self.request.GET.get('producedmin')
if pproducedmin:
    pproduceddt =
datetime.datetime.strptime(pproducedmin, "%d.%m.%Y").date()
    cars.filter(produced__gte = pproduceddt)
pproducedmax = self.request.GET.get('producedmax')
if pproducedmax:
    pproduceddt =
datetime.datetime.strptime(pproducedmax, "%d.%m.%Y").date()
    cars.filter(produced__lte = pproduceddt)
plocation = self.request.GET.get('location')
if plocation:
    cars.filter(location = plocation)
ppackage = self.request.GET.get('package')
if ppackage:
    cars.filter(package = ppackage)
pcreatedmin = self.request.GET.get('createdmin')
if pcreatedmin:
    cars.filter(created__gte = pcreatedmin)
pcreatedmax = self.request.GET.get('createdmax')
if pcreatedmax:
    cars.filter(created__lte = pcreatedmax)
pdrive = self.request.GET.get('drive')
if pdrive:
    cars.filter(drive = pdrive)
penginetype = self.request.GET.get('enginetype')
if penginetype:
    cars.filter(engine_enginetype = penginetype)
pfuel = self.request.GET.get('fuel')
if pfuel:

```

```
        cars.filter(engine_fuel = pfuel)
cars = cars.order_by('-score')
carcont['cars'] = cars
return carcont
```

ДОДАТОК В
Слайди презентації

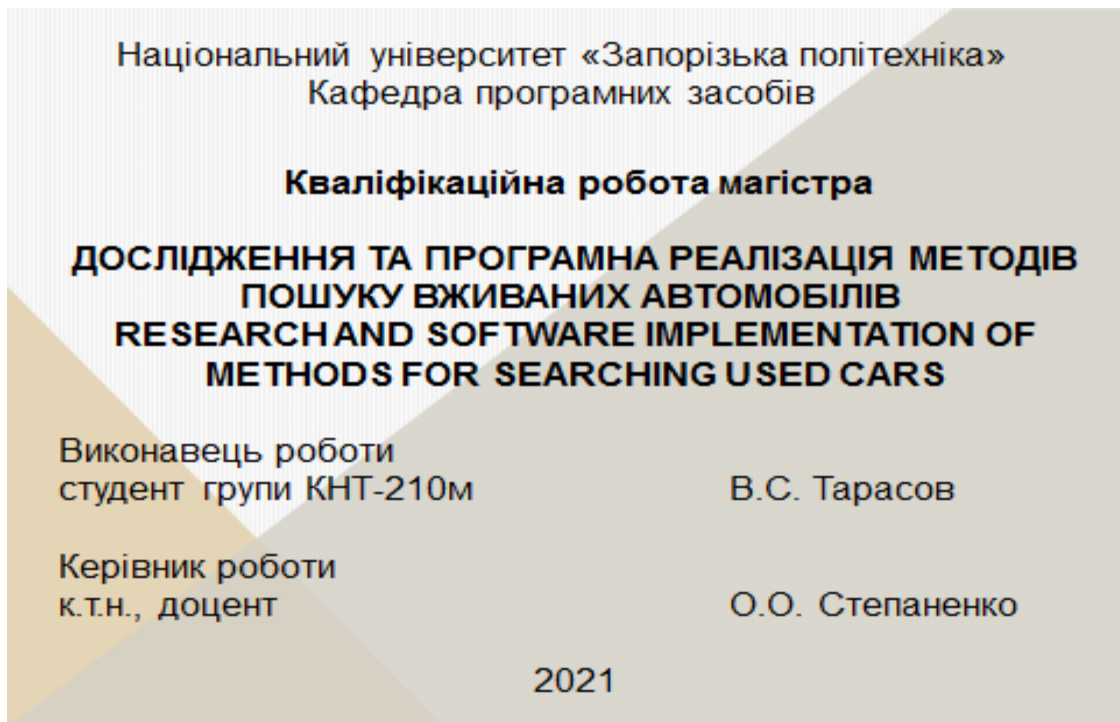


Рисунок В.1 – Слайд 1

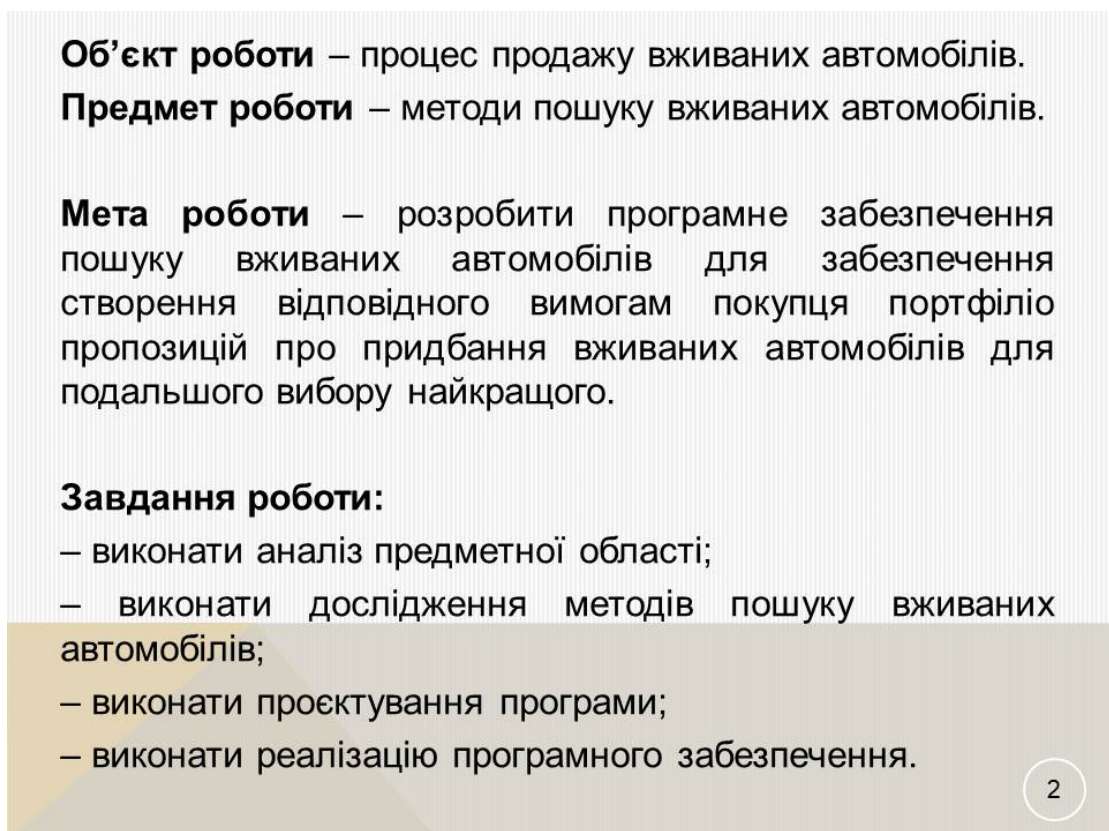


Рисунок В.2 – Слайд 2

Результати аналізу предметної області

Характеристика	Auto Ria	Cars.com	TarasovMarket
Безкоштовний доступ	+	+	+
Пошук за великою кількістю параметрів	+	+	+
Система відгуків	+	+	+
Визначення вартості пропозицій за моделлю	+	+	+
Комунікація з продавцем	+	+	+
Пошук за оголошеннями за VIN-номером	-	-	+
Альтернативні сценарії пошуку	-	-	+

3

Рисунок В.3 – Слайд 3

Метод пошуку вживаних автомобілів



4

Рисунок В.4 – Слайд 4

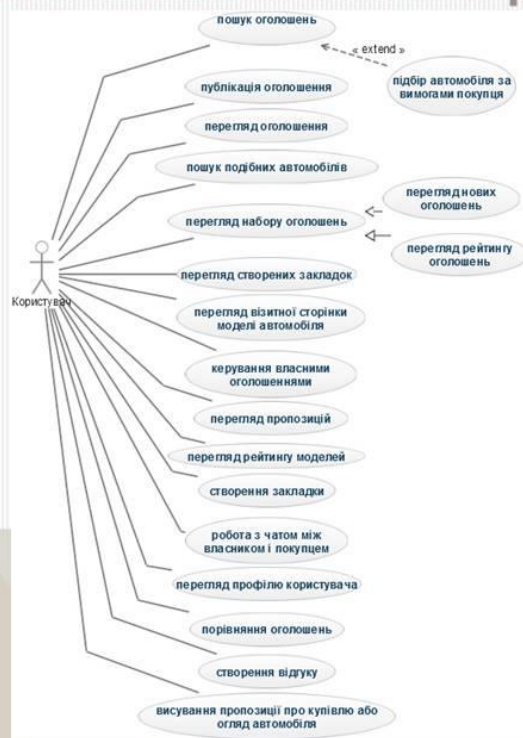
Порівняння мов програмування

Критерій	Python	PHP
Основні парадигми програмування	Об'єктно-орієнтована, функціональна	Об'єктно-орієнтована, обмежено функціональна
Простота розробки вебзастосунків	Висока	Висока
Орієнтованість на довготривалі проєкти	Більша	Менша
Простота створення програм	Більша	Менша
Підтримка методів інтелектуального аналізу даних, у тому числі методу k-середніх	Краща	Гірша
Призначення	Загальне	Веброзробка

5

Рисунок В.5 – Слайд 5

Діаграма варіантів використання програмного забезпечення користувачем



6

Рисунок В.6 – Слайд 6

```

graph TD
    tarasov_market --> views
    tarasov_market --> models
    tarasov_market --> adminpy[admin.py]
    tarasov_market --> templates
    tarasov_market --> apps_py[apps.py]
    tarasov_market --> static
    tarasov_market --> method

    views --> carviewpy[carView.py]
    views --> loginviewpy[loginView.py]
    views --> camodelviewpy[camModelView.py]
    views --> adsupportviewpy[adSupportView.py]
    views --> chatviewpy[chatView.py]
    views --> reviewsviewpy[reviewView.py]

    models --> modelspy[models.py]

    templates --> cardetailhtml[cardetail.html]
    templates --> tripnewhtml[tripnew.html]
    templates --> carratinghtml[carrating.html]
    templates --> vncarshtml[vnCars.html]
    templates --> asearchhtml[aSearch.html]
    templates --> sparamshhtml[sparams.h.html]
    templates --> similarchhtml[similar.html]
    templates --> youngroposhhtml[youngropos.html]
    templates --> chathhtml[chat.html]
    templates --> loginhtml[login.html]
    templates --> profilehtml[profile.html]
    templates --> searchhtml[search.html]
    templates --> carmostviewedhtml[carMostViewed.html]
    templates --> registerhtml[register.html]
    templates --> chatshtml[chats.html]
    templates --> ownreviewshhtml[ownreviews.html]
    templates --> carsmospropostic[carmospropostic]

    static --> caredithtml[cardeth.html]
    static --> userscarshtml[usercars.html]
    static --> camnewesthtml[camNewest.html]
    static --> carmoslikedhtml[carmosLiked.html]
    static --> modelhtml[model.html]
    static --> propositionhtml[proposition.html]
    static --> modelreviewhtml[modelreview.html]
    static --> yourreviewshtml[yourreviews.html]
    static --> propositionshtml[propositions.html]
    static --> pricehtml[price.html]
    static --> epricehtml[eprice.html]
    static --> eproposishhtml[epropositio.html]
    static --> carreviewhtml[carreview.html]
    static --> ownerreviewhtml[ownerreview.html]
    static --> carcomparehtml[carcompare.html]

```

Рисунок В.8 – Слайд 8



Рисунок В.9 – Слайд 9

Робота програми. Оголошення

TarasovMarket

Пошук виживаних автомобілів

Автомобілі

Рейтинг

Пошук

Підбір

Оголошення

Чати

Профіль

Вийти

Mitsubishi Lancer 2018

В закладки

Залишити відгук

Висунути пропозицію

Шукати подоби

Написати власнику

Порівняти

Власник

Ігор Надольський

Оцінка: 4.5

Відгуки

Андрій Молочний
08.12.2021 20:11
Машина в досить гарному стані. Подякував більше ніж вказано. Вартість можливо трохи завищена.

Євген Сілов
08.12.2021 18:49

Стан оголошення: активне.

Розташування: Запоріжжя

Рік виготовлення: 2018

Перше використання: 05.02.2019

Вартість: 560000 грн.

Історія зміни вартості

Оцінка: 4.8

Регістраційний номер: AA1111AA

VIN: GFJQKZ2U486392741

Технічні характеристики

Двигун: Бензиновий, Express 1.8 л 240 кін.сил

Пройдений шлях: 41000 км

Тип кузова: седан

Привід: передній

Трансмісія: механічна

Витрата пального в місті: 10.3 л / 100 км

Витрата пального на трасі: 6.2 л / 100 км

Витрата пального в змішаному циклі: 8.1 л / 100 км

Загальні характеристики

Дверей: 4

Місць: 5

Колір: чорний

Комплектація: Comfort

Особливості

Без ДТП

Проблеми

Має невеличкі пошкодження на лакофарбовому покритті.

Рисунок В.10 – Слайд 10

Робота програми. Пошук вживаних автомобілів



TarasovMarket
Пошук вживаних автомобілів

- Автомобілі
- Рейтинг
- Пошук
- Підбір
- Оголошення
- Чати
- Профіль
- Вийти

Пошук вживаних автомобілів

Виробник	Toyota	Модель	Camry	Двигун	Будь-який	Привід	Передній
Трансмісія	Механічна	Колір	Будь-який	Дверей від	2	Дверей до	5
Місць від	2	Місць до	5	Пробіг від	0	Пробіг до	20000
Тип кузова	Седан	Місто	Залоріжжя	Паливо	Бензин	Комплектація	Будь-яка
Оцінка від	4	Оцінка до	5	Вироблено від	2016	Вироблено до	2021
Публікація оголошення	дд.мм.рррр дд.мм.рррр						

Шукати



Toyota Camry 2020
[Детальніше](#)



Toyota Camry 2017
[Детальніше](#)



Toyota Camry 2015
[Детальніше](#)

11

Рисунок В.11 – Слайд 11

Робота програми. Підбір вживаного автомобіля



TarasovMarket
Пошук вживаних автомобілів

- Автомобілі
- Рейтинг
- Пошук
- Підбір
- Оголошення
- Чати
- Профіль
- Вийти

Підбір автомобіля

Будь ласка, заповніть опитувальник повністю для покращення отриманих результатів.

- На скільки ви любите впливати на керування автомобілем?
 - ☒ суцільно і весь час
 - ☐ люблю повний контроль, але іноді треба і відпочити
 - ☐ це не має значення
 - ☐ для мене головне - дістатися заданого місця
- На скільки критичний для вас зовнішній стан автомобіля?
 - ☐ все має бути ідеально
 - ☒ важливий
 - ☐ хотілось би покраще
 - ☐ готовий до ремонту
 - ☐ це для мене не має значення
- Як ви збираєтесь використовувати автомобіль?
 - ☐ тільки в місті
 - ☐ місто та поїздки по країні
 - ☒ місто та глибинка
 - ☐ місто, глибинка та тривалі подорожі

12

Рисунок В.12 – Слайд 12

Дослідження якості кластеризації

Кількість кластерів	Якість кластеризації
3	0,5276
5	0,6119
7	0,6594
9	0,6803
12	0,7152
15	0,7496
17	0,7815
20	0,8091
25	0,7653

13

Рисунок В.13 – Слайд 13

Висновки

Виконано аналіз предметної області. Розглянуто сучасний стан проблеми продажу вживаних автомобілів, виконано аналіз існуючих програмних засобів пошуку вживаних автомобілів з виконанням порівняння цих засобів з програмою, що розроблялась.

Виконано дослідження методів пошуку вживаних автомобілів.

Наукова новизна роботи полягає в запропонованому методі пошуку вживаних автомобілів, який ґрунтується на застосуванні методу k-середніх для кластеризації пропозицій, що дозволяє на основі опитувальника або обраної моделі автомобіля обирати моделі автомобілів і оголошення, які відповідають вимогам покупця.

Виконано проєктування і реалізацію програми.

Усі поставлені завдання на роботу виконано.

14

Рисунок В.14 – Слайд 14