МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д.т.н., проф.,
_____Ю.П.Кондратенко
«____»_____2022 року

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

РЕКОМЕНДАЦІЙНА ІНФОРМАЦІЙНА СИСТЕМА АВТОСАЛОНУ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧІВ

Спеціальність 124 «Системний аналіз»

124 – МКР – 607.21830801

Студент _____І. І. Бурлака
«____» лютого 2022 р.
Консультант _____ Н. М. Болюбаш
канд. пед. наук, доцент
«____» лютого 2022 р.

Миколаїв – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Освітньо-кваліфікаційний рівень **магістр**
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)
Спеціальність **124 «Системний аналіз»**
(шифр і назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«___» _____ 20__ р.

З А В Д А Н Н Я
на магістерську кваліфікаційну роботу

Бурлаці Ігорю Івановичу

1. Тема магістерської кваліфікаційної роботи «Рекомендаційна інформаційна система автосалону на основі уподобань користувачів».

Керівник роботи Болюбаш Надія Миколаївна, к. пед. наук, доцент.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «__» ____ 20__ р. № _____

2. Строк подання студентом роботи ___ _____ 20__ р.

3. Вхідні (початкові) дані до роботи: загальні відомості про мережеві сервіси у сфері продажу автомобілів в Україні та характеристики автомобілів.

Очікуваний результат роботи: інформаційна система автосалону, де передбачено збір та аналіз інформації у процесі її функціонування й надання рекомендацій користувачам для прийняття рішення про вибір автомобіля .

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

— здійснення аналізу мережевих сервісів у сфері продажу автомобілів та дослідження теоретичних засад створення рекомендаційних систем автосалону на основі уподобань користувачів;

— обґрунтуваня вибору інструментальних засобів розробки інформаційної системи автосалону;
— розробка та здійснення програмної реалізації інформаційної системи автосалону із вбудованою системою рекомендацій.

5. Перелік графічного матеріалу: презентація, рисунки, таблиці.

6. Завдання до спеціальної частини: Охорона праці та безпека у надзвичайних ситуаціях.

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	д.б.н.,професор Л. І. Григор'єва	
Методична частина	к.пед.н., доцент Н.М. Болюбаш	

Керівник роботи __к. пед. наук, доц. Болюбаш Н. М._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Бурлака І. І._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_____» _____ 20__ р.

КАЛЕНДАРНИЙ ПЛАН

виконання магістерської кваліфікаційної роботи

Тема: «Рекомендаційна інформаційна система автосалону на основі уподобань користувачів»

№	Найменування роботи	Початок	Закінчення	Примітки
1.	Отримання завдання на магістерську наукову роботу	18.10.2021	01.11.2021	Виконано
2.	Огляд літератури за темою роботи	01.11.2021	22.11.2021	Виконано
3.	Аналіз існуючих мережевих сервісів у сфері продажу автомобілів	22.11.2021	26.11.2021	Виконано
4.	Аналіз предметної області та методів фільтрації рекомендаційних систем	01.12.2021	13.12.2021	Виконано
5.	Робота над першим та другим розділом пояснювальної записки	14.12.2021	27.12.2021	Виконано
6.	Створення дизайну, проектування та програмна реалізація інформаційної системи	28.12.2021	04.01.2022	Виконано
7.	Робота над третім розділом пояснювальної записки	04.012022	14.01.2022	Виконано
8.	Тестування системи	18.01.2022	24.01.2022	Виконано
9.	Розробка методичної частини	24.01.2022	25.01.2022	Виконано
10.	Розробка спеціальної частини з безпеки в надзвичайних ситуаціях	25.01.2022	01.02.2022	Виконано
11.	Створення слайдів для захисту та написання доповіді	01.02.2022	15.02.2022	Виконано
12.	Обговорення отриманих результатів з керівником та попередній захист магістерської роботи	28.01.2022	31.01.2022	Виконано
13.	Корегування роботи за результатами попереднього захисту	01.02.2022	15.02.2022	Виконано
14.	Остаточне оформлення пояснювальної записки та слайдів	16.02.2022		Виконано
15.	Захист магістерської роботи	24.02.2022		Виконано

Розробив студент _____Бурлака І.І._____ _____

(прізвище та ініціали) (*підпис*)

Керівник роботи __к.пед.н., доц. Болюбаш Н.М. _____ __ _____

(наук. ступінь, вчене звання, прізвище та ініціали) *(підпис)*

«____» _____ 202__р.

АНОТАЦІЯ
до магістерської кваліфікаційної роботи роботи

Тема: «Рекомендаційна інформаційна система автосалону на основі уподобань користувачів»

Студент: Бурлака Ігор Іванович

Керівник: к.пед.н доцент Болюбаш Надія Миколаївна

Магістерська кваліфікаційна робота присвячена розробці та здійсненню програмної реалізації рекомендаційної інформаційної системи автосалону на основі уподобань користувачів з використанням колаборативної фільтрації.

Об'єкт дослідження – діяльність автосалонів з продажу автомобілів.

Предмет дослідження – програмні засоби для підтримки продажу автомобілів у автосалонах з використанням алгоритмів побудови рекомендацій.

Мета дослідження – підвищення ефективності роботи автосалону у сфері подажу автомобілів шляхом створення рекомендаційної інформаційної системи з використанням колаборативної фільтрації, яка зменшує навантаження на менеджерів з продажу та полегшує вибір товару для покупців.

Дипломна робота складається з фахового розділу, методичної і спеціальної частини з охорони праці. Пояснювальна записка дипломної роботи складається зі вступу, трьох розділів, висновків та додатку.

У першому розділі розкрито теоретичні засади використання рекомендаційних систем у сфері продажу автомобілів, проаналізувано сучасний стан мережевих сервісів у сфері продажу автомобілів, розглянуто основні підхлди до створення рекомендаційних систем.

У другому розділі обґрунтовано вибір технологій і засобів розробки рекомендаційної системи. У третьому розділі описано проектування та програмну реалізацію рекомендаційної інформаційної системи автосалону на основі уподобань користувачів.

У спеціальній частині з охорони праці розглядаються питання охорони праці та безпеки у надзвичайних ситуаціях.

Дипломна робота містить ___ сторінку (без додатків), ___ рисунків, ___ таблиці, ___ джерел, ___ додаток.

ABSTRACT

for master's scientific work

Subject: “Recommended car dealership information system based on user preferences”

Student Burlaka Igor Ivanovich
Leader: Ph.D., associate professor Bolyubash Nadiya Mikolaivna

Master's thesis is devoted to the development and implementation of software implementation of the recommended information system of the car showroom based on user preferences using collaborative filtering.

Object of research – activity of car dealerships.

Subject of research – software to support the sale of cars in car dealerships using algorithms for building recommendations.

The purpose of the study is to increase the efficiency of the car showroom in the field of car delivery by creating a recommendation information system using collaborative filtering, which reduces the burden on sales managers and facilitates the choice of goods for buyers.

Thesis consists of a professional section, methodical and special part on labor protection. The explanatory note of the thesis consists of an introduction, three sections, conclusions and an appendix.

The first section reveals the theoretical foundations of the use of recommendation systems in the field of car sales, analyzes the current state of network services in the field of car sales, considers the main approaches to creating recommendation systems. The second section substantiates the choice of technologies and means of developing a recommendation system. The third section describes the design and software implementation of the recommended information system of the showroom based on user preferences.

The special part on labor protection deals with issues of labor protection and safety in emergency situations.

Thesis contains ___ page (without appendices), ___ figures, ___ tables, ___ sources, ___

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему:

«РЕКОМЕНДАЦІЙНА ІНФОРМАЦІЙНА СИСТЕМА АВТОСАЛОНУ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧІВ»

Спеціальність 124 «Системний аналіз»

124 – МКР – 607. 21830801

Студент _____ Бурлака І.І.
«__» _____ 20___ р.

Консультант _____ Болюбаш Н.М.
к.пед.наук, доцент
«__» _____ 20___ р.

м. Миколаїв – 2022

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

БД – база даних

ІС – інформаційна система

РС – рекомендаційна система

ГРС – гібридна рекомендаційна система

CBF – алгоритм фільтрування за контентом (content based filtering)

CF – алгоритм колаборативного фільтрування (collaborative filtering)

ПЗ – програмне забезпечення

ORM – Object Relational Mapping

ВСТУП

Актуальність. Бурхливі темпи інформатизації обумовлюють накопичення великої кількості інформації стосовно різних сфер життєдіяльності сучасного суспільства та сфери продажу автомобілів зокрема. Це супроводжується впровадженням у діяльність автосалонів автоматизованих систем з реалізованими алгоритмами аналізу контенту та цифрових слідів користувачів. Створення рекомендаційних інформаційних систем, які здатні прогнозувати поведінку покупців та давати їм відповідні поради стосовно вибору автомобілів на основі їх уподобань, дозволяє суттєво підвищити ефективність діяльності автосалонів.

Продаж автомобілів останнім часом зростає та стає все більш Інтернет-орієнтованим. Діяльність авторинків та автосалонів розширюється за рахунок впровадження веб-ресурсів, якими широко користуються як покупці, так і ті, хто продає автомобілі та запчастини до них. До найбільш популярних в Україні ресурсів, які надають інформацію про автомобілі та їх характеристики і містять відгуки про рівень обслуговування й сервісу відносять: AvtoBazar, Auto.Ria, RST, AutoSite, сайти-агрегатори Auto.Meta, AutoMoto, UAvto, AvtoPoisk, AutoSale. Кращі веб-сайти автосалонів – Towne Ford, Santa Margarita Toyota, “Фалькон-Авто”, “Ю.Р.К”, орієнтовані на надання допомоги покупцям у знаходженні автомобілів, які задовольняють їх потреби, використовуючи при цьому перш за все інформаційну та маркетингову функції.

Однак аналіз накопиченої інформації шляхом її фільтрації з метою вивчення поведінки споживачів та їх уподобань і потреб, представлений недостатньо. Тому є потреба у створенні рекомендаційної інформаційної системи автосалону, де передбачено автоматизований збір та аналіз інформації у процесі її функціонування й надання рекомендацій користувачам для прийняття рішення про вибір автомобіля.

Мета дослідження – підвищення ефективності роботи автосалону у сфері подажу автомобілів шляхом створення рекомендаційної інформаційної системи з

використанням колаборативної фільтрації, яка зменшують навантаження на менеджерів з продажу та полегшує вибір товару для покупців.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1. Здійснити аналіз мережевих сервісів у сфері продажу автомобілів та дослідити теоретичні засади створення рекомендаційних систем автосалону на основі уподобань користувачів.
2. Обґрунтувати вибір інструментальних засобів розробки інформаційної системи автосалону.
3. Розробити та здійснити програмну реалізацію інформаційної системи автосалону із вбудованою системою рекомендацій.

Об’єктом дослідження є діяльність автосалонів з продажу автомобілів.

Предметом дослідження є програмні засоби для підтримки продажу автомобілів у автосалонах з використанням алгоритмів побудови рекомендацій.

Методологічною основою дослідження є загальнонаукові аналітичні методи та методи колаборативної фільтрації, які дозволили вивчити предмет та об’єкт дослідження, дослідити розвиток науково-методичних засад, напрямів та шляхів підвищення ефективності роботи автосалону шляхом створення рекомендаційної інформаційної системи, яка зменшує навантаження на менеджерів з продажу та полегшує вибір товару покупцями.

Наукова новизна одержаних результатів дослідження полягає у тому, що автором: запропоновано та обґрунтовано напрями вдосконалення надання рекомендацій з вибору автомобілів покупцям; одержали подальший розвиток підходи врахування уподобань користувачів при наданні рекомендацій; узагальнено теоретичні засади створення рекомендаційних систем.

Результати дослідження обговорювалися на XXIV Всеукраїнської науково-практичної конференції «Могилянські читання – 2021: Досвід та тенденції розвитку суспільства в Україні: глобальний, національний та регіональний аспекти» та отримали схвалення.

Практичне значення отриманих результатів полягає в тому, що сформульовані теоретичні положення та практичні рекомендації щодо підвищення ефективності продажу автомобілів можна застосувати у діяльності автосалонів шляхом впровадження розробленої рекомендаційної системи.

Структура магістерської роботи. Відповідно до мети, завдань і предмета дослідження, магістерська робота містить основну, методичну та спеціальну частини. Основна частина магістерської роботи складається із вступу, трьох розділів, висновку, списку використаних джерел та __ додатків. Загальний обсяг магістерської роботи – __ сторінок, із них основного тексту основної частини – __ сторінок, методичної частини – __ сторінок, спеціальної – __ сторінок. Кількість використаних джерел – __.

РОЗДІЛ 1. ТЕОРЕТИЧНІ ЗАСАДИ ВИКОРИСТАННЯ РЕКОМЕНДАЦІЙНИХ СИСТЕМ У СФЕРІ ПРОДАЖУ АВТОМОБІЛІВ

1.1. Розвиток мережевих сервісів з продажу автомобілів

За останні 10 років попит на автомобілі виріс дуже стрімко. Якщо у минулому сторіччі та в 2000-х роках мала група суспільства могла собі дозволити придбати автомобіль, то з часом ситуація змінилася. На даний момент кожна третя родина в Україні має хоча б 1 авто. Попит виріс, тому кожен з нас може простежити яка велика кількість автосалонів наявна у тому чи іншому місті як нашої країни, так і будь-якої іншої країни за кордоном. Оскільки ІТ також розвивалися у цей період, то кожен власник автосалону, хоча б, середнього рівня забезпечив свій бізнес веб-сайтом. Людина, як споживач у сьогоденні має високий запит будь-який продукт покупки, тому кількість автосалонів значущого рівня виявляється досить малою.

Зробимо аналіз наявних сайтів для підтримки діяльності з продажу автомобілів. До найбільш популярних в Україні ресурсів, які надають інформацію про автомобілі та їх характеристики і містять відгуки про рівень обслуговування й сервісу відносять: AvtoBazar, Auto.Ria, RST, AutoSite, сайти-агрегатори Auto.Meta, AutoMoto, UAvto, AvtoPoisk, AutoSale. Кращі веб-сайти автосалонів – Towne Ford, Santa Margarita Toyota, “Фалькон-Авто”, “Ю.Р.К”, орієнтовані на надання допомоги покупцям у знаходженні автомобілів, які задовольняють їх потреби, використовуючи при цьому перш за все інформаційну та маркетингову функції.

Розглянемо сайт (рис. 1.1) з продажу нових автомобілів - http://pa.od.ua/car. Розробка доволі цікава та належить до високого рівня, оскільки рік розробки сайту – 2015, використано новий дизайн, новітні технології кодування. При переході за посиланням відкривається головна сторінка, яка несе в собі загальну інформацію. На даному етапі треба мислити як користувач, що бажає підібрати

авто для купівлі. Отже, переходимо до пункту «Машины», У випадаючому списку одразу зрозуміло, що весь товар розділено за критерієм країни-виробника. На даному етапі виникає перший недолік – мала кількість користувачів обізнана в питанні країни виробника автомобіля тієї марки, яку хотів би розглянути як варіант купівлі.

Рис. 1.1. Скріншот сайту PLATINUM AUTO

Після вибору країни та марки авто, надається відфільтрований набір авто, які влаштовують запит користувача. Дана функція є оптимальною та універсальною для більшості додатків данного типу. В ході аналізу данного сайту виявлено, що на сайті представлена незначна кількість марок автомобілів.

Другий аналог - https://rikauto.com.ua/ru. Це сайт (див. рис. 1.2) для продажу як нових авто, так і використаних авто. Сайт розроблено ще у 2011 році, тому одразу можно сказати, що його рівень низький. Переходячи за посиланням одразу відкривається головна сторінка з основною формацією, як і в попередньому прикладі. На відміну від попереднього прикладу, даний сайт пропонує обирати авто одразу за назвою марки (не наводячи малюнок логотипу марки) або за

категорією «По типу кузова». Дана ідея про вибір кузова має сенс і розглянута для додавання її до власного застосунку.

Не зважаючи на велику кількість недоліків є одна вагома перевага – великий вибір марок автомобілів. Це і є один з основних напрямків, які включено до розробки власного web-застосунку.

Рис. 1.2. Скріншот сайту РІКавто

Останнім аналогом є сайт http://autoboutique.cars.ua/. Це сайт (див. рис. 1.3) для продажу нових авто, рік розробки не вказано, але за виглядом та функціональність можна його віднести до сайту високого рівня розробки. Відкривши головну сторінку видно, що одразу представлено автомобілі дуже високої вартості. Тому такий сайт можна віднести до користувачів, економічний рівень життя яких належить до «високого» або, принаймні, до «вище середнього». Вибір автомобіля здійснюється у випадаючому меню, яке також складається виключно з назви, без малюнку логотипу марки автомобіля.

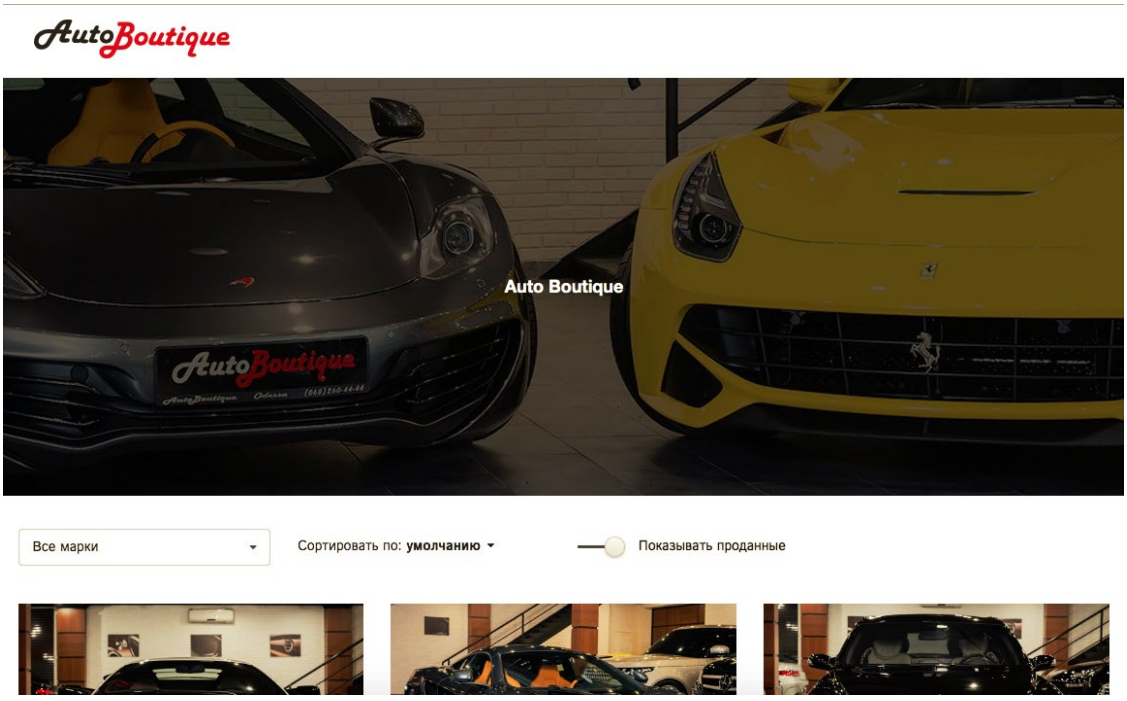

Рис. 1.3. AutoBoutique

Більшість автосалонів мають сайт-візитку, але покупцям все ж таки простіше прийти в салон, щоб отримати достатньо інформації для визначення бажаного автотранспорту або його комплектуючих. У веденні бізнесу з продажу автомобілів спостерігаються такі проблеми:

- відсутність повноцінного консультування онлайн;
- працівники витрачають багато часу на процедуру консультування безпосередньо в автосалоні;
- низький рівень інформативності про авто різних марок в одному інтернет-джерелі.

Тким чином, діяльність авторинків та автосалонів розширюється за рахунок впровадження веб-ресурсів, якими широко користуються як покупці, так ті, хто продає автомобілі та запчастини до них. Однак аналіз накопиченої інформації шляхом її фільтрації з метою вивчення поведінки споживачів та їх уподобань і потреб, представлений недостатньо. Тому є потреба у створенні рекомендаційної

інформаційної системи автосалону, де передбачено автоматизований збір та аналіз інформації у процесі її функціонування й надання рекомендацій користувачам для прийняття рішення про вибір автомобіля.

1.2. Основні підходи до створення рекомендаційних систем

Рекомендаційна система – це система, що будує рейтинг товарів користувача серед колосального набору даних, на підґрунті вже існуючої інформації про користувача. Товари – елементи, що надаються сервісом, в яких користувач може бути зацікавлений, такі як: автомобілі, фільми, музика, побутові товари, книги та ін. Є декілька способів, щоб описати зацікавленість користувачів: можна скористуватись оцінками та відгуками, що користувач залишає до послуг, або (та) використавши ключові слова, характеристики, що має товар.

Інтереси користувачів можуть бути представлені декількома способами: із застосуванням оцінок, які користувачі надають товарам або за допомогою ключових слів кожного товару. Щоб зберігати вподобання користувачів стосовно товарів, РС використовують профілі користувачів.

У більшості РС профіль користувача містить набори оцінок та/або ключових слів (тегів). Оцінки, надані користувачами товарам, можуть належати різним проміжкам (0-1, 1-5, 1-10): чим вищий рейтинг, тим більше конкретний товар сподобався користувачу. Після кожного оцінювання всі рейтинги користувача агрегуються через ряд обчислень, вимірюються схожість користувачів, а потім прогнозуються рекомендації для даного користувача. Ключові слова автоматично підвантажуються з текстів або товарів, які користувачі проглядали або оцінювали в минулому. Вони також можуть мати ваги в залежності від того, на скільки користувач оцінив конкретне слово, або більш значущі слова матимуть більшу вагу, ніж менш значущі (алгоритм TF-IDF).

Після цього тексти (товари) зіставляються з профілем користувача та найбільш відповідні йому – рекомендуються. Рейтинги можуть бути явними та неявними.

Явна оцінка – це оцінка, якою користувач показав зацікавленість даним товаром в межах своєї системи оцінювання. Неявні рейтинги вираховуються з історії покупок або поведінки користувачів. До їх переваг можна віднести зниження навантаження на користувача оцінюванням товарів. Джерелом неявних рейтингів можуть бути час, витрачений на читання статті, посилання на товар в інших джерелах 14 (наприклад алгоритм ранжування сторінок Google). Інші індикатори поведінки перегляду, як рух курсору, ввід клавіатури та швидкість прокрутки сторінки, також були досліджені в якості неявних показників інтересу та показали непогані результати.

Базову архітектуру рекомендаційної системи (рис. 1.4) можна зобразити наступним чином:

1) *Довідкова інформація* – існуюча інформація про послуги або товари, що надаються сервісом.

2) *Вхідна інформація* – інформація, що користувач явно або неявно надає системі для отримання рекомендацій. Може бути оцінками, поведінкою або прямими запитами, наприклад пошук авто за маркою або певними характеристиками.

3) *Рекомендаційний алгоритм* – алгоритм, що комбінує зібрані довідкові дані та вхідні дані, отримані від користувача, для побудови рекомендацій.

Типові системи в якості довідкових даних найчастіше використовують портрети користувачів, а в якості вхідних – дії користувача (оцінки товарів, час, проведений за переглядом товару, тощо).

У більшості рекомендаційних систем використовується один з двох базових підходів: контентна фільтрація (content-based filtering) або колаборативна фільтрація (collaborative filtering). Також існує клас підходів, що базуються на поєднанні двох основних – гібридна фільтрація (hybrid filtering). Розглянемо більш детально кожен з цих підходів

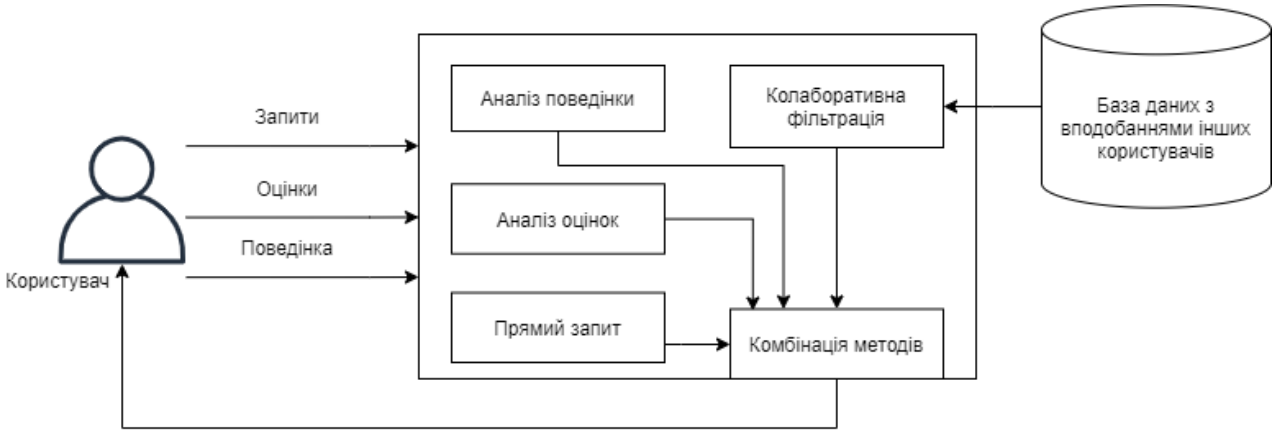

Рисунок 1.4. Базова архітектура рекомендаційної системи

1. *Колаборативна фільтрація* є методом прогнозу в рекомендаційних системах, який використовує відомі переваги (оцінки) групи користувачів для прогнозування невідомих переваг (оцінок) іншого користувача. За допомогою цього алгоритму будується певна таблиця користувачів, які групуються за схожістю, та прогнозуються результати для інших користувачів.

Колаборативна фільтрація прогнозує рекомендації, засновані на моделі попередньої поведінки користувача. Ця модель може бути побудована виключно на основі поведінки цього користувача або - що більш ефективно - з урахуванням поведінки інших користувачів з подібними характеристиками.

У тих випадках, коли колаборативна фільтрація бере до уваги реакцію інших користувачів, вона використовує знання про групу (group knowledge) для вироблення рекомендацій на основі схожості користувачів. По суті рекомендації базуються на автоматичній взаємодії множини користувачів і на виділені (методом фільтрації) тих користувачів, які демонструють схожі уподобання або шаблони поведінки.

Наприклад, при створенні веб-блогу, на якому необхідно запровадити рекомендаційну систему на основі інформації від багатьох користувачів, які переглядають автомобілі, можна згрупувати цих користувачів за їх інтересами. Можна об'єднати в одну групу користувачів, які цікавляться однаковими марками

авто і за цією інформацією ідентифікувати найпопулярніші моделі серед тих, які дивляться учасники цієї групи. Потім конкретному користувачу з цієї групи будуть рекомендуватися найпопулярніші марки автомобілів, які він ще не переглядав.

Переваги колаборативної фільтрації: швидка робота алгоритмів, мала кількість ітерацій, прості в реалізації. Недоліки: не вирішені проблеми холодного старту, шахрайства, нема що рекомендувати новим або нетиповим користувачам, є розрідженими матриці оцінок (іноді неможливо зробити прогноз).

2. *Контентна фільтрація* будує рекомендації, базуючись на поведінці користувачів. Наприклад, цей підхід може використовувати ретроспективну інформацію про перегляди марок автомобілів на сайті автосалону. Якщо який-небудь користувач зазвичай переглядає інформацію про автомобілі одного призначення (спортивні, легкові тощо) і регулярно залишає коментарі під матеріалами, то контентна фільтрація може використовувати цю ретроспективну інформацію для виявлення подібного контенту і пропозиції такого контенту як рекомендованого для цього користувача. Цей контент може бути визначений в ручному режимі або завантажений автоматично на базі інших методів подібності.

Переваги контентної фільтрації: більш точний результат, немає проблеми холодного старту, оскільки рекомендації базуються на моделі об’єкта, а не на попередніх оцінках користувачів. Недоліки: “затратне” створення моделі (її побудова досить складна), невисока швидкодія алгоритмів (багато обчислень) та втрата точності при скороченні параметрів моделі.

Гібридні алгоритми фільтрації, які поєднують колаборативну і контентну фільтрацію, також підвищують ефективність (і складність) рекомендаційних систем (табл. 1.1). Об'єднання результатів колаборативної і контентної фільтрації потенційно дозволяє підвищити точність рекомендації. Гібридний підхід може бути корисний, якщо застосування колаборативної фільтрації відбувається на сильно розріджених даних (приклад холодного старту). Гібридний підхід дозволяє спочатку зважувати результати згідно контентної фільтрації, а потім зміщувати ці

ваги у напрямку до колаборативної фільтрації (в міру "визрівання" доступного набору даних для конкретного користувача).

Таблиця 1.1

Моделі рекомендаційних систем на базі гібридних алгоритмів

Тип	**Характеристика**
Зважена	Комбінуються оцінки, отримані іншими методами фільтрації, агрегуються для отримання єдиної рекомендації
Комутуюча	Використовуються критерії, для перемикання між методами контентної та колаборативної фільтрації
Каскадна	Рекомендаціям надаються чіткі пріоритети, нижчі пріоритети розривають зв’язки при оцінюванні вищих
Комбінаційна	Критерії отримані з різноманітних джерел комбінуються в єдиний рекомендаційний алгоритм
Нарощувальна	Критерії отримані з одного рекомендаційного методу використовуються як вхідні дані для іншого
Змішана	Одночасно показуються рекомендації, отримані різними рекомендаційними методами
Мета-рівень	Один рекомендаційний метод використовується для надання довідкових даних для іншої системи

Переваги алгоритмів гібридної фільтрації: велика швидкодія; кращі результати. Недоліки: дуже дорога розробка рекомендаційної системи, оскільки реалізація цього типу алгоритмів дуже складна, важко підтримувати, оскільки навіть незначні зміни в роботі призводять до змін роботи алгоритму.

Вибір моделі впровадження рекомендаційної системи залежить від типу та кількості даних, які необхідно проаналізувати з метою надання рекомендацій. Використання методу контентної фільтрації має проблеми у разі пасивної

поведінки користувачів та при наданні рекомендацій новому користувачеві, який ще не виконував ніяких дій у системі. Метод колаборативної фільтрації не вимагає конкретизованих запитів, тому для реалізації рекомендаційної інформаційної системи автосалону доцільно взяти за основу метод колаборативної фільтрації, який базується на даних, що містять характеристики автомобілів. Найбільш достовірним підходом до отримання початкових даних є явний зворотній зв'язок, коли інформація отримується від користувача через системний інтерфейс. Але користувач не завжди готовий надати достатню кількість інформації, тому у рекомендаційних системах застосовують також неявний зворотній зв'язок. При цьому уподобання визначаються автоматично шляхом відслідковування дій користувача: історії покупок, переглядів товарів, аналізу переходів за посиланнями, пошукових запитів, натиснення кнопок тощо.

Основними задачами, які має вирішувати рекомендаційна система автосалону, є пошук користувачем автомобілів за вказаними характеристиками та надання йому рекомендацій з вибору. Від користувача система отримує вектор бажаних характеристик: $U = \{u_1, u_2, ..., u_n\}$, де n – кількість характеристик. Ці дані необхідно порівняти з векторами характеристик автомобілів $I_k = \{i_1, i_2, ..., i_n\}$ (I_k – вектор характеристик k-го автомобіля), які містить система та найти ті з них, які є найбільш близькими до запиту користувача. Для якісної роботи система повинна мати базу даних з набором характеристик наявних на авторинку моделей авто.

Для вибору найбільш підходящих автомобілів доцільно у цьому випадку застосувати *метод k-ближніх сусідів*, який для розв'язання заданої задачі реалізується насутпним чином.

1. На першому кроці алгоритму задається число k – кількість найближчих об'єктів (автомобілів), які будуть вважатися схожими (сусідами).

2. На другому кроці знаходять міру близькості від нового об'єкта – автомобіля з характеристиками, сформованими відповідно до уподобань користувача до кожного з об'єктів – автомобілів, інформація про які міститься у

базі даних інформаційної системи. Розрахунок мір близькості може здійснюватися різними способами: шляхом обчислення відстані Евкліда, коефіцієнту кореляції Пірсона, косинусу подібності, коефіцієнту подібності Джакарта. Найбільш поширеним є використання відстані Евкліда d_E, яка розраховується за формулою:

$$d_E(x_i, x_j) = \sqrt{\sum_{t=1}^{m} (x_{it} - x_{jt})^2}$$, де

x_i – модель автомобіля характеристиками, які вказав користувач,

x_j – характеристики j-го автомобіля у базі даних інформаційної системи.

Перед розрахунком відстані необхідно у разі потреби здійснити перетворення типів даних та їх нормалізацію.

3. На наступному кроці алгоритму після розрахунку мір близькості бажаних для користувача характеристик автомобіля до характеристик наявних моделей автомобілів, система рекомендує k моделей, які є найбільш близькими до уподобань користувача.

Програмна реалізація такого алгоритму відносно проста. Результат роботи легко піддається інтерпретації. Можливість модифікації алгоритму, шляхом використання найбільш підходящих функцій сполучення й метрик дозволяє підбудувати алгоритм під конкретне завдання.

1.3. Постановка задачі

Провівши аналіз мережевих сервісів з продажу автомобілів та основних підходів до розробки рекомендаційних систем було зроблено висновок про розробку рекомендаційної системи автосалону на основі методу колаборативної фільтрації, який базується на даних, що містять характеристики автомобілів.

Об’єктом дослідження є діяльність автосалонів з продажу автомобілів.

Предметом дослідження є програмні засоби для підтримки продажу автомобілів у автосалонах з використанням алгоритмів побудови рекомендацій.

Метою дослідження є підвищення ефективності роботи автосалону у сфері подажу автомобілів шляхом створення рекомендаційної інформаційної системи з використанням колаборативної фільтрації, яка зменшують навантаження на менеджерів з продажу та полегшує вибір товару для покупців.

Досягнення поставленої мети обумовлює необхідність вирішення наступних **завдань**:

1. Здійснити аналіз мережевих сервісів у сфері продажу автомобілів та дослідити теоретичні засади створення рекомендаційних систем автосалону на основі уподобань користувачів.
2. Обґрунтувати вибір інструментальних засобів розробки інформаційної системи автосалону.
3. Розробити та здійснити програмну реалізацію інформаційної системи автосалону із вбудованою системою рекомендацій.

Висновок до розділу 1

Установлено, що сьогодні цільова споживацька аудиторія все більше використовує Інтернет для отримання інформації про автомобілі, їх характеристики та ціни, відгуки про рівень обслуговування та сервісу. Це обумовило різке зростання кількості представницьких, корпоративних веб-сайтів та сайтів-візиток, які спрямовані на надання такої інформації користувачам. Однак вказані ресурси виконують в основному інформаційну та маркетингову функції, а наявінсть функцій, які відповідають за збір інформації з метою вивчення поведінки споживачів, їх смаків і потреб та надання їм рекомендацій, представлена не достатньо. Існує потреба у інформаційних системах автосалону, у яких є вбудовані рекомендаційні системи на основі уподобань користувача**.**

У результаті проведеного дослідження установлено, що рекомендаційна система – це система, що будує рейтинг відеоконтенту серед великого набору даних, на підґрунті вже існуючої інформації про уподобання користувача при перегляді відео. Виявлено, що у більшості рекомендаційних систем використовуються рекомендаційні алгоритми колаборативної та контентної фільтрації та підхід, який базується на поєднанні цих алгоритмів – гібридна фільтрація. Колаборативна фільтрація передбачає здійснення кластеризації для виявлення користувачів з подібними характеристиками в оцінці відео й на основі цього прогнозує персональні рекомендації для відео перегляду. Контентна фільтрація передбачає надання рекомендацій для перегляду на основі ретроспективних даних про автомобілі. Гібридна фільтрація поєднує у собі переваги алгоритмів колаборативної та контентної фільтрації й є найбільш популярною.

РОЗДІЛ 2. ІНСТРУМЕНТАЛЬНІ ЗАСОБИ РОЗРОБКИ РЕКОМЕНДАЦІЙНОЇ СИСТЕМИ

2.1. Мова програмування Python

Python — це інтерпретована мова програмування високого рівня. Його мовна структура та об’єктно-орієнтований підхід розроблені, щоб допомогти програмістам писати чіткі логічні коди для малих і великих проектів.

Python підтримує декілька парадигм програмування, включаючи структурне (процедурне), об’єктно-орієнтоване та функціональне програмування.

Python 2.0 був випущений 16 жовтня 2000 року з багатьма важливими новими функціями, включаючи збірник сміття для виявлення циклу та підтримкою Unicode. Python 3.0 був випущений 3 грудня 2008 року. Багато з його основних функцій було перенесено до Python версій 2.6.x і 2.7.x. Реліз Python 3 містить утиліту 2to3, яка автоматично (принаймні частково) перетворює код Python 2 на Python 3. Термін дії Python 2.7 спочатку був встановлений у 2015 році, а потім перенесений на 2020 рік через побоювання, що велика кількість існуючого коду може бути нелегкою в перенесені на Python 3. Він не випускатиме більше виправлень безпеки та інших покращень. Наприкінці життєвого циклу підтримується лише Python 3.6.x і новіших версій. Python 3.9.2 і 3.8.8 прискорені, оскільки всі версії Python мають проблеми з безпекою, що призводить до можливого віддаленого виконання коду.

Python — це мова програмування з кількома парадигмами. Він повністю підтримує об’єктно-орієнтоване програмування та структурне програмування, а багато його функцій підтримують функціональне програмування та аспектно-орієнтоване програмування (включаючи метапрограмування та метаоб’єктний (магічний метод)). Розширення підтримує багато інших парадигм, включаючи дизайн контрактів і логічне програмування.

Python використовує комбінацію динамічного введення тексту та підрахунку посилань і збирач сміття, який визначає цикли для керування пам'яттю. Він також має динамічне розділення імен (пізнє прив'язування) для зв'язування імен методів і змінних під час виконання програми.

Python розроблено для забезпечення певної підтримки функціонального програмування в традиції Lisp. Він має функції фільтрації, відображення та скорочення; перераховує розуміння, словник, набір і вираз генератора. Стандартна бібліотека має два модулі (itertools і functools), які реалізують функціональні інструменти, запозичені з Haskell і Standard ML.

Python не має всіх своїх функцій, вбудованих у його ядро, що зроблено, щоб бути дуже розширюваним (з модулями). Ця компактна модульність робить його особливо популярним як спосіб додавання програмованих інтерфейсів до існуючих програм. Python прагне до простішого та стислого синтаксису та граматики, надаючи розробникам можливість вибору методів кодування. Python висвітлює філософію дизайну: «Повинен бути один – бажано лише один – очевидний спосіб зробити це». Алекс Мартеллі, член Python Software Foundation і автор книги про Python, написав: «У культурі Python опис чогось як «розумного» не вважається компліментом».

Розробники Python прагнуть уникнути передчасної оптимізації та відмовляються виправляти некритичні частини референсної реалізації CPython, які забезпечують невелике збільшення швидкості за ціною наочності. Коли швидкість важлива, програмісти Python можуть перемістити важливі за часом функції в розширення, написані на таких мовах, як C, або використовувати PyPy, компілятор «точно вчасно». Також доступний Cython, який перетворює скрипти Python на C і здійснює прямі виклики API рівня C до інтерпретатора Python.

Поширеним новим словом у спільноті Python є python, яке може мати широкий спектр значень, пов'язаних зі стилем програмування. Сказати, що код є pythonic, означає, що він добре використовує ідіоми Python, що є природним, або що він демонструє вільну мову, яка відповідає мінімалістичній філософії Python і

підкреслює читабельність. На відміну від цього, код, який важко зрозуміти або прочитати як приблизну копію іншої мови програмування, називається не Python.

Python використовує пробіли замість фігурних дужок або ключових слів для розділення блоків. Збільшення відступу відбувається після певних операторів; зменшення відступу означає кінець поточного блоку. Тому візуальна структура програми точно відображає семантичну структуру програми. Цю функцію іноді називають «зовнішніми» правилами, і в деяких інших мовах вона є, але в більшості мов відступ не має семантичного значення. Рекомендований розмір відступу – чотири пробіли.

Оператори Python включають (серед іншого):

- ✓ Використовуйте знак рівності = оператор присвоєння.
- ✓ оператор if, умовно виконати блок коду, а також else та elif (скорочення від else-if).
- ✓ Оператор for для перегляду ітераційного об’єкта шляхом захоплення кожного елемента в локальну змінну для використання вкладеним блоком.
- ✓ Поки умова істинна, виконується оператор while блоку коду.
- ✓ Оператор try, який дозволяє витягам, що містяться у вкладених блоках коду, захоплювати й обробляти вміст, відмінний від речень; він також гарантує, що незалежно від того, як виходить блок, чистий код у блоці в кінцевому підсумку буде виконано.
- ✓ Оператор raise використовується для отримання вказаного винятку або повторного виклику перехопленого винятку.
- ✓ Оператор класу, який виконує блок коду та приєднує свій локальний простір імен до класу для використання в об’єктно-орієнтованому програмуванні.
- ✓ Оператор def, який визначає функцію або метод.
- ✓ Оператор Python 2.5, випущений у вересні 2006 р. [82], охоплює блок коду в диспетчері контексту (наприклад, отримання блокування перед запуском блоку коду і зняття блокування після цього або відкриття файлу, а потім

його закриття), що дозволяє Поведінка, подібна до пошуку ресурсів, — це ініціалізація (RAII), яка замінює поширену ідіому try / finally.

- ✓ Оператор break виходить з циклу.
- ✓ Оператор continue пропускає цю ітерацію і переходить до наступного елемента.
- ✓ Оператор del видаляє змінну, а це означає, що посилання з імені на значення видаляється, а спроби використати змінну призведуть до помилки. Видалену змінну можна повторно призначити.
- ✓ Оператор проходу, виконує функцію NOP. Це синтаксично необхідно для створення порожнього блоку коду.
- ✓ Оператори тверджень, які використовуються для перевірки умов, що застосовуються під час налагодження.
- ✓ Оператор yield, який повертає значення з функції генератора. У Python 2.5 yield також є оператором. Ця форма використовується для реалізації спільного плану.
- ✓ Оператор повернення, який використовується для повернення значення з функції.
- ✓ Оператор імпорту, що використовується для імпорту модуля, чия функція або змінна може використовуватися в поточній програмі. Існує три способи імпорту: імпорт <ім’я модуля> [як <псевдонім>] або за допомогою <ім’я модуля> імпорт * або за допомогою <ім’я модуля> імпорт <визначення 1> [як <псевдонім 1>], <визначення 2> [як <псевдонім 2>], ...
- ✓ Оператор присвоєння (=) діє шляхом прив'язки імені як посилання на окремий динамічно розподілений об'єкт. Потім змінні можуть бути відскочені в будь-який час до будь-якого об’єкта. У Python ім'я змінної є загальним власником посилання і не має фіксованого типу даних, пов’язаного з ним. Однак у даний момент часу змінна

- ✓ буде посилатися на якийсь об'єкт, який матиме тип. Це називається динамічним набором тексту і протиставляється статично набраним мовам програмування, де кожна змінна може містити лише значення певного типу.

Деякі вирази Python подібні до виразів у таких мовах, як C і Java, а деякі ні.

Метод для об’єкта — це функція, приєднана до класу об’єкта; синтаксис instance.method (аргумент) — це синтаксичний цукор Class.method (екземпляр, аргумент) для поширених методів і функцій. Методи Python мають явний параметр self для доступу до даних екземпляра, що є протилежністю неявного self (або цього) в деяких інших об’єктно-орієнтованих мовах програмування (наприклад, C++, Java, Objective-C або Ruby).

Python використовує набір качок і має типізовані об’єкти, але нетиповані імена змінних. Обмеження типу не перевіряються під час компіляції; навпаки, операції над об’єктом можуть бути невдалими, а це означає, що об’єкт не має відповідного типу. Хоча Python динамічно типізується, він забороняє невизначені операції (наприклад, додавання чисел до рядка) і не намагається зрозуміти їх тихо.

Python дозволяє програмістам визначати власні типи, використовуючи класи, які найчастіше використовуються в об’єктно-орієнтованому програмуванні. Новий екземпляр класу створюється шляхом виклику класу (наприклад, SpamClass() або EggsClass()).Клас — це екземпляр типу метакласу (сам екземпляр), який дозволяє виконувати метапрограмування та відображати.

До версії 3.0 Python мав два типи класів: старі та нові. Синтаксис двох стилів однаковий, різниця полягає в тому, чи успадковується об’єкт класу безпосередньо чи опосередковано (усі класи нового стилю успадковуються від об’єкта і є екземплярами типу). У Python 2, починаючи з Python 2.2, ви можете використовувати ці два типи класів. Класи старого стилю були ліквідовані в Python 3.0.

Довгостроковий план передбачає підтримку інкрементного введення, а в Python 3.5 синтаксис мови дозволяє вказувати статичні типи, але вони не

перевіряються в реалізації CPython за замовчуванням. Експериментальна додаткова статична перевірка типу муру підтримує перевірку типу під час компіляції.

CPython є еталонною реалізацією Python. Він написаний на C, відповідає стандарту C89 і має кілька вибраних функцій C99 (в більш пізніх версіях C він вважається застарілим; CPython включає власні розширення C, але сторонні розширення не обмежуються старими C версія, наприклад, може бути реалізована на C11 або C++). Він компілює програми Python у проміжні байт-коди, які потім виконуються його віртуальною машиною. CPython постачається з великою стандартною бібліотекою, яка написана сумішшю C і рідного Python. Він працює на багатьох платформах, включаючи Windows (починаючи з Python 3.9, інсталятор Python навмисно не зміг встановити в Windows 7 і 8; Windows XP підтримувалася до Python 3.5) і більшість сучасних Unix-подібних систем, включаючи macOS (і Apple M1). Mac), починаючи з Python 3.9.1, з експериментальним інсталятором) та неофіційною підтримкою, наприклад VMS. Переносність платформи є одним з її головних пріоритетів, і навіть OS/2 і Solaris підтримувалися під час Python 1 і 2; з тих пір рівень підтримки багатьох платформ знизився.

Версії, які не містять нових функцій, випускаються приблизно кожні 3 місяці і випускаються, коли було виправлено достатню кількість помилок з моменту останнього випуску. Уразливості безпеки також були усунені в цих версіях. Третя і остання частина номера версії збільшено. Багато альфа-, бета- та реліз-кандидатів також попередньо переглядаються та тестуються перед остаточним релізом. Хоча кожна проблема має приблизний графік, якщо код не готовий, вони зазвичай затримуються. Команда розробників Python відстежує стан коду, виконуючи велику кількість модульних тестів під час процесу розробки.

Головна академічна конференція для Python — PyCon. Також існують спеціальні навчальні програми Python, наприклад Pyladies.

Pythons 3.10 припиняє wstr (буде видалено в Python 3.12; це означає, що розширення Python потрібно буде змінити до того часу), а також планує додати відповідність шаблону до мови.

З 2003 року Python є однією з десяти найпопулярніших мов програмування в індексі програмного забезпечення TIOBE, а станом на лютий 2021 року це третя за популярністю мова (після Java та C). Вона була обрана мовою програмування року («Річний найвищий ріст рейтингу») у 2007, 2010, 2018 та 2020 роках (єдина мова, яка робила це чотири рази).

Емпіричні дослідження показали, що для проблем програмування, включаючи маніпулювання рядками та пошук у словнику, мови сценаріїв, такі як Python, є більш продуктивними, ніж звичайні мови, такі як C і Java, і виявилося, що споживання пам'яті часто краще, ніж у Java, і немає краще, ніж C або Наскільки поганий C++".

Великі організації, які використовують Python, включають Wikipedia, Google, Yahoo!, CERN, NASA, Facebook, Amazon, Instagram, Spotify, а також деякі менші організації, такі як ILM та ITA. Сайт соціальних новин Reddit в основному написаний на Python.

Python може бути мовою сценаріїв для веб-програм, наприклад mod_wsgi для веб-сервера Apache. Стандартні API були розроблені для полегшення роботи цих програм через інтерфейс шлюзу веб-сервера. Веб-фреймворки, такі як Django, Pylons, Pyramid, TurboGears, web2py, Tornado, Flask, Bottle і Zope, підтримують розробників для розробки та підтримки складних програм. Pyjs і IronPython можна використовувати для розробки клієнтської сторони програм на основі Ajax. SQLAlchemy можна використовувати як перетворювач даних у реляційних базах даних. Twisted — це фреймворк для програмного зв’язку між комп’ютерами, який використовується (наприклад) Dropbox.

Такі бібліотеки, як NumPy, SciPy і Matplotlib, дозволяють ефективно використовувати Python в наукових обчисленнях, тоді як спеціалізовані бібліотеки, такі як Biopython і Astropy, забезпечують функції, залежні від домену.

Програмне забезпечення SageMath-Math з інтерфейсом ноутбука, яке можна запрограмувати на Python: його бібліотека охоплює багато аспектів математики, включаючи алгебру, комбінаторику, числову математику, теорію чисел і числення. OpenCV має палітурні прив'язки з багатим набором функцій для комп'ютерного зору та обробки зображень.

Python часто використовується в проектах штучного інтелекту та машинного навчання, включаючи такі бібліотеки, як TensorFlow, Keras, Pytorch і Scikit-learn. Python, як мова сценаріїв з модульною архітектурою, простим синтаксисом і багатими інструментами обробки тексту, часто використовується для обробки природних мов. Python був успішно інтегрований у багато програмних продуктів як мова сценаріїв, включаючи програмне забезпечення кінцевих елементів, таке як Abaqus, засоби моделювання 3D-параметрів, такі як FreeCAD, пакети 3D-анімації, такі як 3ds Max, Blender, Cinema 4D, Lightwave, Houdini, Maya, modo, MotionBuilder, Softimage, композитор візуальних ефектів Nuke, програми для 2D-зображень (такі як GIMP, Inkscape, Scribus і Paint Shop Pro) і програми для нот (наприклад, автор і група). Налагоджувач GNU використовує Python як хороший принтер для відображення складних структур, таких як контейнери C++. Esri рекламує Python як найкращий вибір для написання сценаріїв у ArcGIS. Він також використовувався в багатьох відеоіграх і був прийнятий як перша з трьох доступних мов програмування в Google App Engine, двома іншими є Java і Go.

Багато операційних систем використовують Python як стандартний компонент. Він постачається з більшістю дистрибутивів Linux, AmigaOS 4 (з використанням Python 2.7), FreeBSD (як пакет), NetBSD, OpenBSD (як пакет) і macOS, і може використовуватися з командного рядка (термінал). У багатьох дистрибутивах Linux використовуються інсталятори, написані на Python: Ubuntu використовує інсталятор Ubiquity, а Red Hat Linux і Fedora — інсталятор Anaconda. Gentoo Linux використовує Python у своїй системі керування пакетами Portage.

Python широко використовується для інформаційної безпеки, включаючи розробку експлойтів. Більшість додатків Sugar Laptop XO, які зараз розробляє Sugar Labs, написані на Python. Проект одноплатної комп'ютерної програми Raspberry Pi прийняв Python як основну мову програмування користувачів. LibreOffice включає Python і має намір замінити Java на Python. Його постачальник сценаріїв Python є основною особливістю версії 4.0 від 7 лютого 2013 року.

Виходячи зі сфери застосування та величезних можливостей мови Python, описаної вище, можна зробити висновок, що вона універсальна. Ось чому ця мова програмування була обрана для виконання цієї роботи.

2.2. Середовище розробки PyCharm

PyCharm — це інтегроване середовище розробки (IDE) для професійної розробки програм на мові програмування Python. Його розробила чеська компанія JetBrains. За основу взято інше інтегроване середовище розробки Intelli IDEA. PyCharm забезпечує аналіз коду, графічний налагоджувач, інтегрований модуль тестування та інтеграцію системи контролю версій (VCS), а також підтримує використання Django для веб-розробки та Anaconda для аналізу даних (рис. 2.1).

PyCharm є кросплатформним і має версії для Windows, macOS та Linux. Спільнота доступна за ліцензією Apache, а також є професійна версія з додатковими функціями, випущена за власною ліцензією. PyCharm Community Edition — версія PyCharm з відкритим кодом, яка була запущена 22 жовтня 2013 року.

Існує PyCharm у двох версіях: безкоштовній (Community Edition) і платній (Professional Edition). Також існує версія Educational Edition - перероблене середовище розробки для навчання програмуванню. На відміну від звичайних IDE дана версія має убудовані приклади й завдання для навчання, і спрощений інтерфейс.

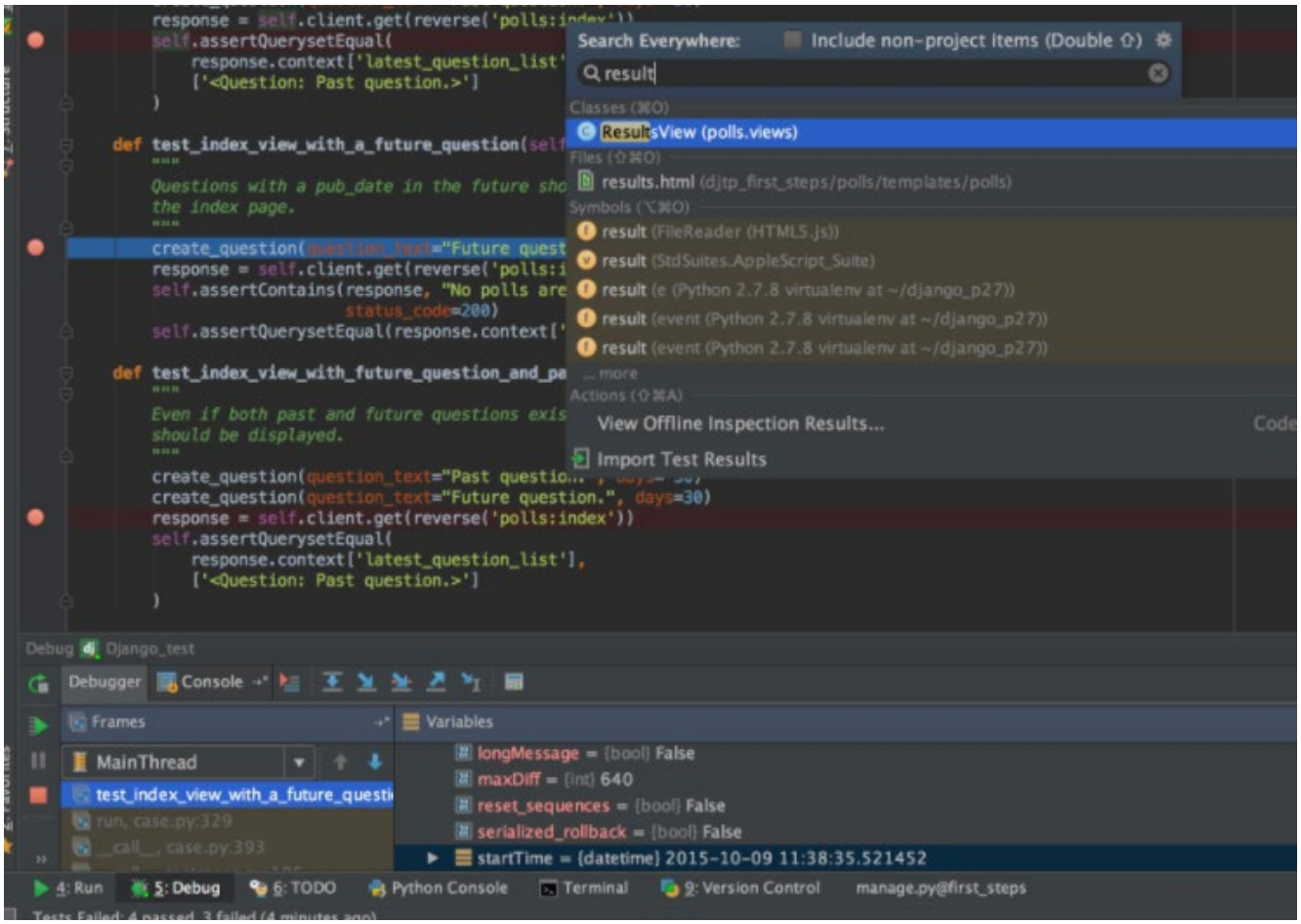

Рис. 2.1. Середовище професійної розробки PyCharm

Платний варіант є більш розширеною й функціональною версією з можливістю розробки в тому числі багатомовних веб-застосунків. Professional Edition підтримує фреймворки: Django, Flask, Google App Engine, Pyramid, web2py. Та дає можливість вилученої розробки, а також роботи з базами даних.

PyCharm має зручний редактор коду з усіма корисними функціями: підсвічуванням синтаксису, автоматичним форматуванням, доповненням і відступами. PyCharm дозволяє перевіряти версії інтерпретатора мови на сумісність, а також використовувати шаблони коду. Утиліта підтримує всі свіжі версії Django, а також IronPython, Jython, Cython, PyPy wxPython, PyQt, PyGTK і багато інших інструментів. В PyCharm можна проводити інтегроване Unit тестування, використовувати інтерактивні консолі для Python, Django, SSH, відладчика баз даних.

PyCharm має велику колекцію плагінів, і його можна використати у зв'язуванні з різними трекерами JIRA, Youtrack, Lighthouse, Redmine, Trac.

Розглянемо основні можливості PyCharm.

1. Допомога при написанні коду – PyCharm робить розробку максимально продуктивною завдяки функціям автодополнения й аналізу коду, миттєвому підсвічуванню помилок і швидких виправлень. Автоматичні рефакторинги допомагають ефективно редагувати код, а зручна навігація дозволяє миттєво переміщатися по проекту.

Допомога при редагуванні – редактор PyCharm призначений для максимально продуктивної розробки на Python, JavaScript, CoffeeScript, TypeScript, CSS і популярних мовах шаблонів. Функції автодополнения, виявлення помилок і швидкі виправлення враховують особливості кожної з підтримуваних мов.

Зручна навігація - розумний пошук дозволяє швидко перейти до будь-якого класу, файлу або символу, а також до потрібного вікна або дії IDE. Перехід до вищестоячого методу, тесту, оголошенню, входження або реалізації здійснюється в одне натискання.

Швидкі й безпечні рафакторинги – PyCharm надає широкі можливості реорганізації коду за допомогою рефакторингів Rename й Delete, Extract Method, Introduce Variable, Inline Variable, Inline Method і багатьох інших. Рефакторинги враховують особливості конкретної мови або фреймворка, допомагаючи вносити зміни по всьому проекту.

2. Убудовані інструменти для розроблювачів – PyCharm пропонує великий набір інструментів з коробки: убудований отладчик й інструмент запуску тестів, профіліровщик Python, полнофункціональний убудований термінал, інструменти для роботи з базами даних. IDE інтегрована з популярними системами контролю версій, містить убудований SSH-термінал, підтримує можливості вилученої розробки й вилучених інтерпретаторів, а також інтеграцію з Docker й Vagrant.

Налагодження, тестування й профілювання – візуальний відладчик Python й JavaScript дозволяє створювати й запускати тести, використовувати функції розумного редагування, і переглядати звіти про запуск у зручному графічному інтерфейсі. Повністю контролювати код завдяки інтеграції із профіліровщиком Python.

Контроль версій, розгортання й вилучена розробка – використання універсального інтерфейсу для роботи з Git, SVN, Mercurial й іншими системами контролю версій. Запуск та налагоджування коду на вилученій машині. Набудовування автоматичного розгортання на вилученому хості або віртуальній машині та управління інфраструктурою за допомогою Vagrant і Docker.

Інструменти для роботи з базами даних – доступ до Oracle, SQL Server, PostgreSQL, MySQL й інших баз даних здійснюється прямо з IDE. PyCharm допомагає редагувати SQL-код, виконувати запити, переглядати дані й змінювати схеми.

3. Веб-разработка – PyCharm надає повноцінну підтримку різних веб-фреймворков і платформ для розробки на Python, підтримує темплейтні мови цих фреймворків, а також JavaScript, CoffeeScript, TypeScript, HTML/CSS, AngularJS, Node.js і багато хто інших.

Веб-фреймворки Python – PyCharm забезпечує підтримку популярних вебфреймворков, таких як Django, Flask, Google App Engine, Pyramid й web2py. Ви можете створювати й налагоджувати Django-шаблони, працювати з утилітами manage.py й appcfg.py, а також використовувати специфічні для фреймворків автодополнення й навігацію.

JavaScript й HTML – IDE забезпечує першокласну підтримку мов JavaScript, CoffeeScript, TypeScript, HTML й CSS, а також їхніх сучасних спадкоємців. Відтладчик JavaScript включений в PyCharm й інтегрований з Run-конфигурацією запуску сервера Django.

Live Edit - функція Live Editing Preview дозволяє відкрити редактор і браузер одночасно й відслідковувати результати внесених у код змін на

вебсторінці. PyCharm зберігає зміни автоматично, і вони миттєво відображаються в браузері без перезавантаження сторінки.

4. Інструменти для наукових обчислень – PyCharm дає змогу працювати з ноутбуками Jupyter, запускати команди в інтерактивній консолі Python, підключати бібліотеки Anaconda, а також працювати з іншими бібліотеками для наукових обчислень й аналізу даних, включаючи Matplotlib й NumPy.

Інтерактивна консоль для Python – дозволяє запустити консоль REPL для Python, що має багато переваг над стандартною консоллю. Серед них перевірка синтаксису на лету за допомогою інспекцій, зіставлення дужок і лапок й, звичайно, автодополнения.

Підтримка наукових бібліотек – PyCharm підтримує Pandas, Numpy, Matplotlib й інші бібліотеки для наукових обчислень. IDE забезпечує розумне редагування, дозволяє переглядати набори даних у вигляді графіків й у табличній формі. Інтеграція з Conda – дозволє створювати окреме оточення Conda й инсталювати тільки потрібні бібліотеки для кожного проекту. PyCharm дозволяє легко створювати й вибирати правильне оточення.

2.3. СКБД SQLite

Для структурованого зберігання використовуються бази даних SQLite – популярний формат баз даних, який з’являється в багатьох мобільних системах і традиційних операційних системах. SQLite — це технологічна бібліотека, яка реалізує незалежну конфігурацію бази даних SQL без серверів із нульовою конфігурацією. Код SQLite є загальнодоступним, тому його можна використовувати безкоштовно для будь-яких цілей, комерційних чи приватних.

SQLite — це вбудований механізм баз даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. SQLite безпосередньо читає та записує звичайні дискові файли. Повна база даних SQL з кількома таблицями, індексами, тригерами та представленнями міститься в

одному дисковому файлі. Як правило, чим швидше працює SQLite, тим більше пам’яті ви надаєте йому. Однак навіть у середовищі з низьким рівнем пам’яті продуктивність зазвичай досить хороша. Відповідно до способу використання, SQLite може бути швидшим, ніж прямий ввід / вивід файлової системи.

Більшість вихідного коду SQLite призначено лише для тестування та перевірки. Автоматизований набір тестів запускає мільйони тестових випадків, включає сотні мільйонів окремих операторів SQL і забезпечує 100% тестування гілок. SQLite реагує на помилки виділення пам’яті та дискового вводу-виводу. Все це перевіряється автоматизованим тестуванням за допомогою спеціальних інструментів тестування, які імітують збої системи. Звичайно, навіть з усіма цими тестами все одно є помилки. Але на відміну від деяких подібних проектів (особливо від комерційних конкурентів), SQLite відкритий і чесний щодо всіх помилок, а також надає список помилок і похвилинну історію змін коду.

Тому, щоб чітко зрозуміти переваги, ми вибрали одного з конкурентів SQLite, а саме SQL Server, а потім коротко порівняли їх:

- Модель ціноутворення: SQLite безкоштовний;
- Серверні операційні системи:
  - SQLite не потребує сервера;
  - SQL Server працює на Linux та Windows;
- API та інші методи доступу:
  - SQLite підтримує такі драйвери: ADO.NET, JBDC, ODBC;
- SQL Server підтримує: OLE DB, ADO.NET, JBDC, ODBC, Табличний потік даних (TDS);
- Підтримувані мови програмування:
  - SQL Server підтримує такі мови програмування (C#, C++, Delphi, Go, Java, JavaScript (Node.js), PHP, Python, R, Ruby, Visual Basic);
  - SQLite підтримує майже будь-які мови програмування, про які ви можете подумати (Actionscript, Ada, Basic, C, C#, C++, D, Delphi,

Forth, Fortran, Haskell, Java, JavaScript, Lisp, Lua, MatLab, Objective-C, OCaml, Perl, PHP, PL/SQL, Python, R, Ruby, Scala, Scheme, Smalltalk, Tcl);

- Підтримка збереженої процедури:
  - SQLite не підтримує збережену процедуру;
  - SQL Server має його з мовами Transact SQL та .NET;
- Методи розділення:
  - SQLite не підтримує;
  - У SQL Server таблиці можуть розподілятися між кількома файлами (горизонтальне розділення);
- Методи реплікації:
  - SQLite не підтримує;
  - SQL Server підтримує;
- Контроль доступу користувачів: SQLite не має концепції контролю доступу користувачів.

SQL Server має чіткі права доступу відповідно до стандарту SQL.

Виходячи з усіх описаних вище переваг SQLite та факту первинної наявності бази даних SQLite в проектах Django, основною СКБД для розроблення даної інформаційної системи обрано саме SQLite.

2.4. Бібліотека машинного навчання Scikit-learn

Scikit-learn – це безкоштовна бібліотека машинного навчання для мови програмування Python. Вона містить різні алгоритми класифікації, регресії та кластеризації, включаючи машини опорних векторів, випадкові ліси, підвищення градієнта, k-середніх і DBSCAN. Scikit-learn розроблена для взаємодії з числовими та науковими бібліотеками Python NumPy і SciPy. Scikit-learn – це фінансовий проект NumFOCUS.

Проект scikit-learn розпочався як scikits.learn, проект Google Summer of Code французького вченого Девіда Курнапо. Його назва походить від уявлення про те, що це «SciKit» (SciPy Toolkit), окремо розроблене та поширене стороннє розширення для SciPy. Scikit-learn є однією з найпопулярніших бібліотек машинного навчання на GitHub. Бібліотека дозволяє генерувати нові значення та працювати з розбалансованими вибірками. Бібліотека містить готові датасети, що дозвлоляє одразуж приступати до створення моделей, не витрачаючи часу на отримання, очистку та перетворення даних.

Scikit-learn в основному написаний на Python і широко використовує NumPy для високопродуктивної лінійної алгебри та операцій з масивами. Крім того, деякі основні алгоритми написані на Cython для підвищення продуктивності. Машини опорних векторів реалізовані за допомогою обгортки Cython навколо LIBSVM; логістичної регресії та лінійної опорної векторної машини за аналогічною обгорткою навколо LIBLINEAR. У таких випадках розширення цих методів за допомогою Python може бути неможливим.

Scikit-learn добре інтегрується з багатьма іншими бібліотеками Python, такими як Matplotlib і plotly для побудови графіків, NumPy для векторизації масивів, Pandas dataframes, SciPy та багатьма іншими.

2.5. Фреймворк Django

Django – це безкоштовний популярний багатофункціональний себрверний веб-фреймворк, створений на основі Python, що відповідає архітектурному зразку model-template-views (MTV). Він підтримується Django Software Foundation (DSF), американською незалежною організацією, створеною як некомерційна організація.

Основна мета Django – полегшити створення складних веб-сайтів, керованих базами даних. Структура підкреслює багаторазовість та "підключеність" компонентів, менше коду, низьку зв’язок, швидкий розвиток та

принцип "не повторюватися". Python використовується всюди, навіть для налаштувань, файлів та моделей даних. Django також надає додатковий адміністративний інтерфейс створення, читання, оновлення та видалення, який генерується динамічно за допомогою самоаналізу та налаштовується за допомогою моделей адміністратора.

Веб-системи на Django будуються з одного або декількох додатків, які рекомендується робити відчуджуваними й такими, що підключаються. Це одне з помітних архітектурних відмінностей цього фреймворка від деяких інших. Також, на відміну від багатьох інших фреймворков, оброблювачі URL в Django конфігуруються явно (за допомогою регулярних виразів), а не автоматично задаються зі структури контролерів.

Django проектувався для роботи під керуванням Apache (з модулем mod_python) і з використанням PostgreSQL як база даних. У цей час, крім PostgreSQL, Django може працювати з іншими СУБД: MySQL (MariaDB), SQLite, Microsoft SQL Server, DB2, Firebird, SQL Anywhere й Oracle. Для роботи з базою даних Django використає власний ORM, у якому модель даних описується класами Python, і по ній генерується схема бази даних.

Архітектура Django схожа на "Модель-Подання-Контролер" (MVC). Контролер класичної моделі MVC приблизно відповідає рівню, що в Django називається Подання (View), а презентаційна логіка Подання реалізується в Django рівнем Шаблонів (Templates). Через цього уровневую архітектуру Django часто називають "Модель-Шаблон-Подання" (MTV).

Спочатку розробка Django велася для забезпечення більше зручної роботи з ресурсами новин, що досить сильно відбилося на архітектурі: фреймворк надає ряд засобів, які допомагають у швидкій розробці веб-сайтов інформаційного характеру. Наприклад, розроблювачеві не потрібно створювати контролери й сторінки для адміністративної частини сайту, в Django є убудований додаток для керування вмістом, яке можна включити в будь-який сайт, зроблений на Django, і яке може управляти відразу декількома сайтами на одному сервері.

Адміністративний додаток дозволяє створювати, змінювати й видаляти будь-які об'єкти наповнення сайту, протоколюючи всі зроблені дії, і надає інтерфейс для керування користувачами й групами (з пообъектным призначенням прав).

Деякі можливості Django:

- ✓ ORM, API доступу до БД із підтримкою транзакцій
- ✓ убудований інтерфейс адміністратора, із уже наявними перекладами на багато мов
- ✓ диспетчер URL на основі регулярних виражень
- ✓ розширювана система шаблонів з тегами й спадкуванням
- ✓ система кеширования
- ✓ інтернаціоналізація
- ✓ архитектура, що підключається до додатків, які можна встановлювати на будь-які Django-сайти
- ✓ "generic views" - шаблони функцій контролерів
- ✓ авторизація й аутентификация, підключення зовнішніх модулів аутентификации: LDAP, Openi та інші.
- ✓ система фільтрів ("middleware") для побудови додаткових оброблювачів запитів, як наприклад включені в дистрибутив фільтри для кеширования, стиску, нормалізації URL і підтримки анонімних сесій
- ✓ бібліотека для роботи з формами (спадкування, побудова форм по існуючій моделі БД)
- ✓ убудована автоматична документація по тегах шаблонів і моделям даних, доступна через адміністративний додаток

Деякі компоненти фреймворка між собою зв'язані слабко, тому їх можна досить просто заміняти на аналогічні. Але з деякими (наприклад, з ORM) це зробити не дуже просто. Крім можливостей, убудованих у ядро фреймворка, існують пакети, що розширюють його можливості.

На базі Django розроблено досить багато готових рішень, розповсюджуваних під вільною ліцензією, серед яких системи для керування інтернет-магазинами, універсальні системи керування змістом, а також більш вузьконаправлені проєкти.

Висновок до розділу 2

Для розробки рекомендаційної інформаційної системи автосалону було використано мову програмування Python, інтегроване середовище розробки PyCharm, яке забезпечує аналіз коду, має графічний налагоджувач, інтегрований модуль тестування та інтеграцію з системами контролю версій, інструменти для роботи з базами даних, підтримує фреймворк Django, який тиакож було використано, і платформи для розробки на Python та бібліотеки для наукових обчислень Pandas, Numpy, Matplotlib. Для реалізації алгоритмів фільтрації рекомендаційної системи було використано бібліотеку машинного навчання Scikit-learn.

Зберігання даних було реалізоване з використанням реляційної бази даних SQLite, яка не використовує парадигму клієнт-сервер, а є бібліотекою, з якою програма компонується й зберігає всю базу даних (включаючи таблиці, індекси й дані) у єдиному файлі на там, де виконується програма.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ АВТОСАЛОНУ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧІВ

3.1. Діаграма прецедентів та діаграма діяльності

Рекомендаційна інформаційна система включає в себе авторизацію під різними статусами які відповідають правам людини, яка авторизується. Створено такі статуси, відповідно до прав:

- адміністратор;
- клієнт.

Вище перераховані статуси відповідають порядку зменшення кількості прав.

Адміністратор має такі права та можливості:

- ведення клієнтської бази;
- ведення бази товару;
- консультування клієнтів.

Клієнт має такі права та можливості:

- перегляд товару;
- заповнення форми з характеристиками товару;
- отримання інформації про наявні автомобілі, які є в автосалоні за уподобаннями користувача.

Узагальнити інформацію про права та можливості користувачів із різними статусами можна за допомогою рис. 3.1.

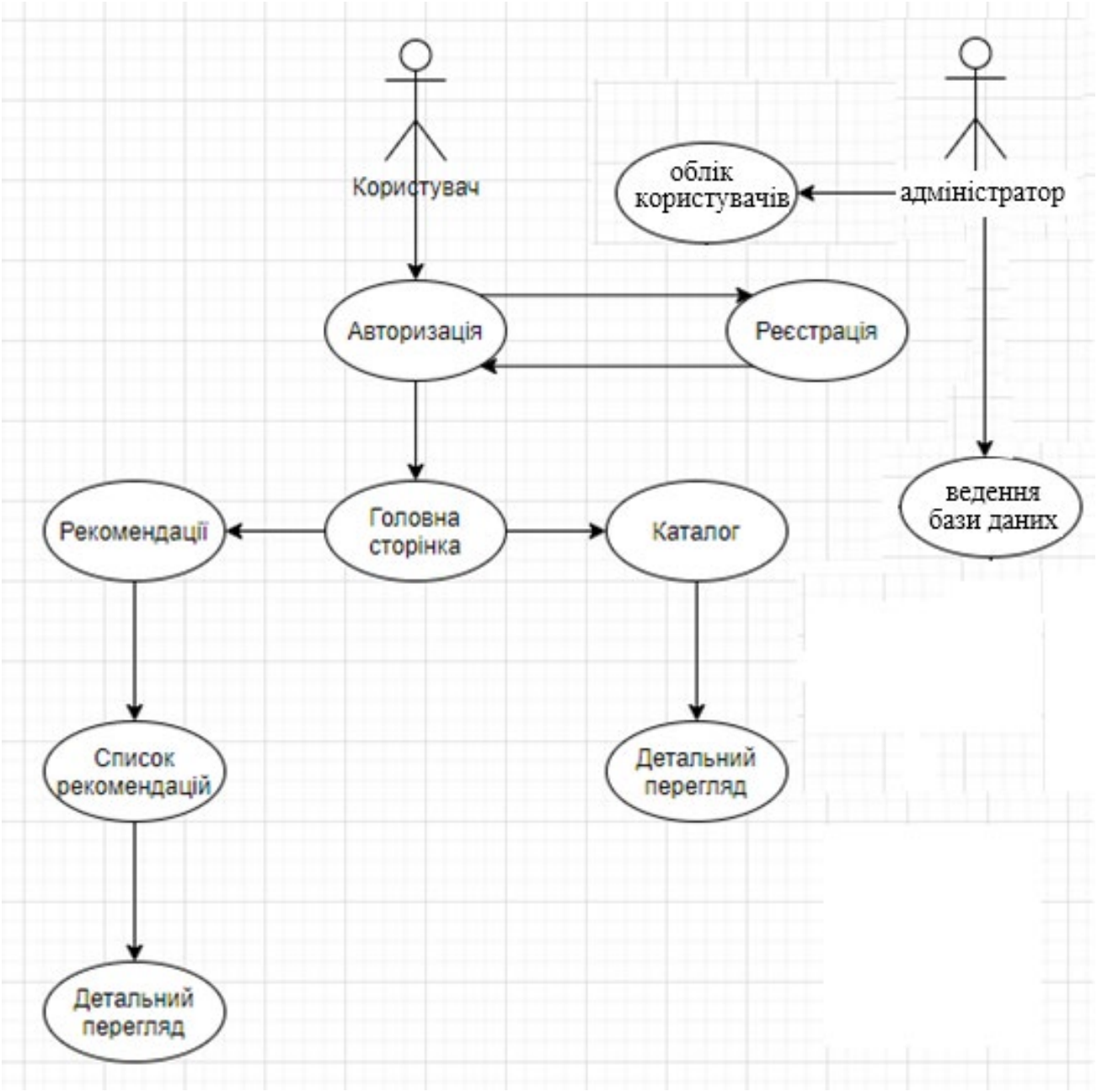

Рис. 3.1. Діаграма прецедентів автосалону

Наведена діаграма прецедентів міститть можливі сценарії використання для адміністратора та користувача. Користувач може перегладати каталог товарів та надавати інформацію про бажані характеристики авто. Адміністратор відповідає за наповнення каталогу та має можливості для його редагування.

В ході підготовки програмного проекту було створено діаграму, яка зображає можливі дії користувачів, варіанти подальшої поведінки системи. Один з таких прикладів зображено на рис. 3.2.

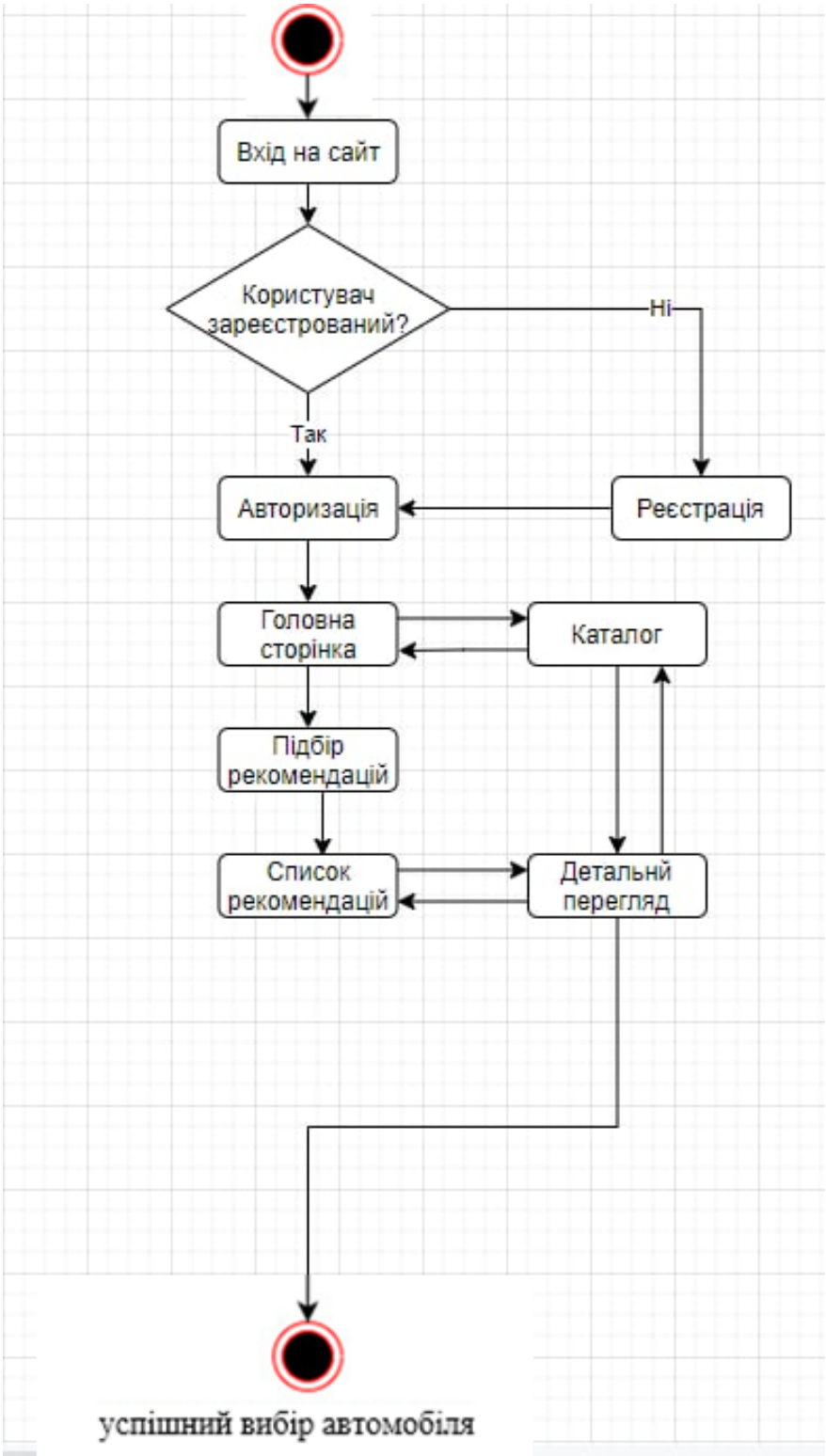

Рисунок 3.3. Діаграма діяльності

3.2. Створення проекту Django в PyCharm

Середовище розробки PyCharm Professional Edition підтримує фреймворк Django й працює з різними версіями Python. Для створення проекту в PyCharm необхідно обрати Create New Project (рис. 3.3).

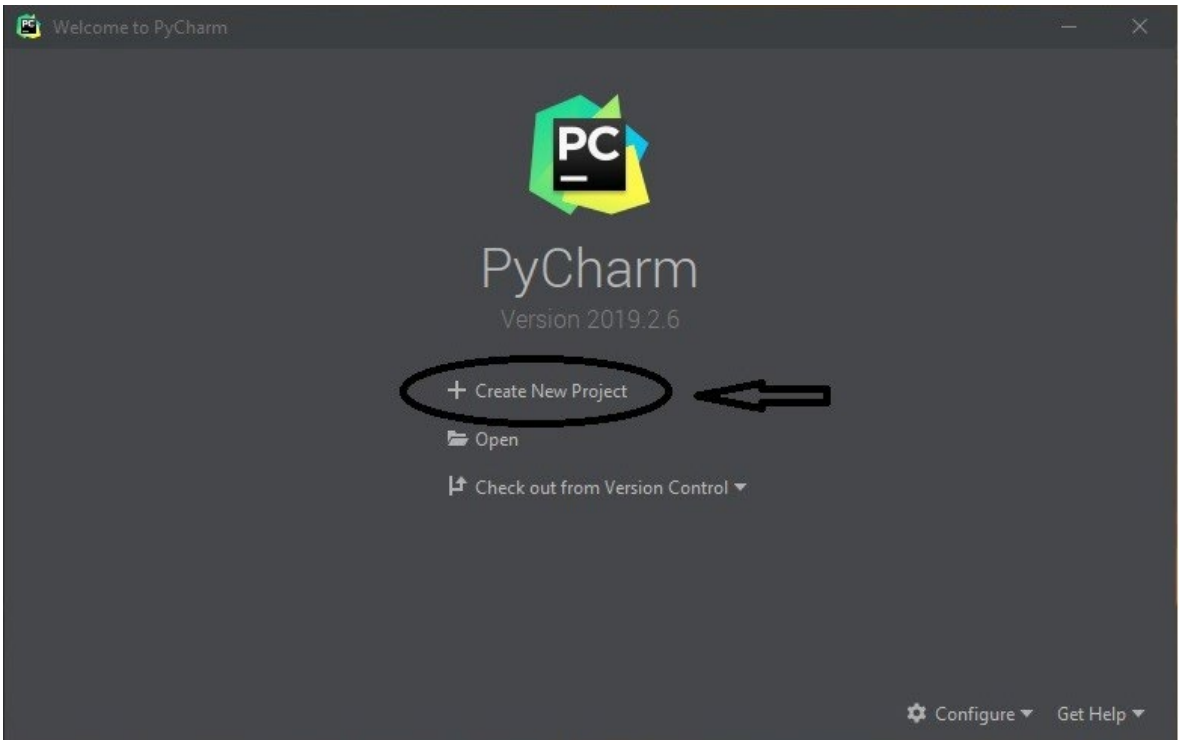

Рис. 3.3. Створення проекту в PyCharm

Далі у вікні створення нового проекту необхідно обрати Django, здійснити необхідні налаштування проекту і натиснути Create (рис. 3.4). Після цього чекаємо скачування фрейворку та створення проекту. Проектом називається сукупність усього програмного коду, що становить розроблювальний сайт. Фізично він є папкою, у якій перебувають різноманітні файли з вихідним кодом (рис. 3.5). У папці із проектом буде створена структура файлів. Розглянемо їх докладніше.

manage.py – програмний файл із кодом однойменної утиліти, з використанням якої виробляються різні дії над самим проектом.

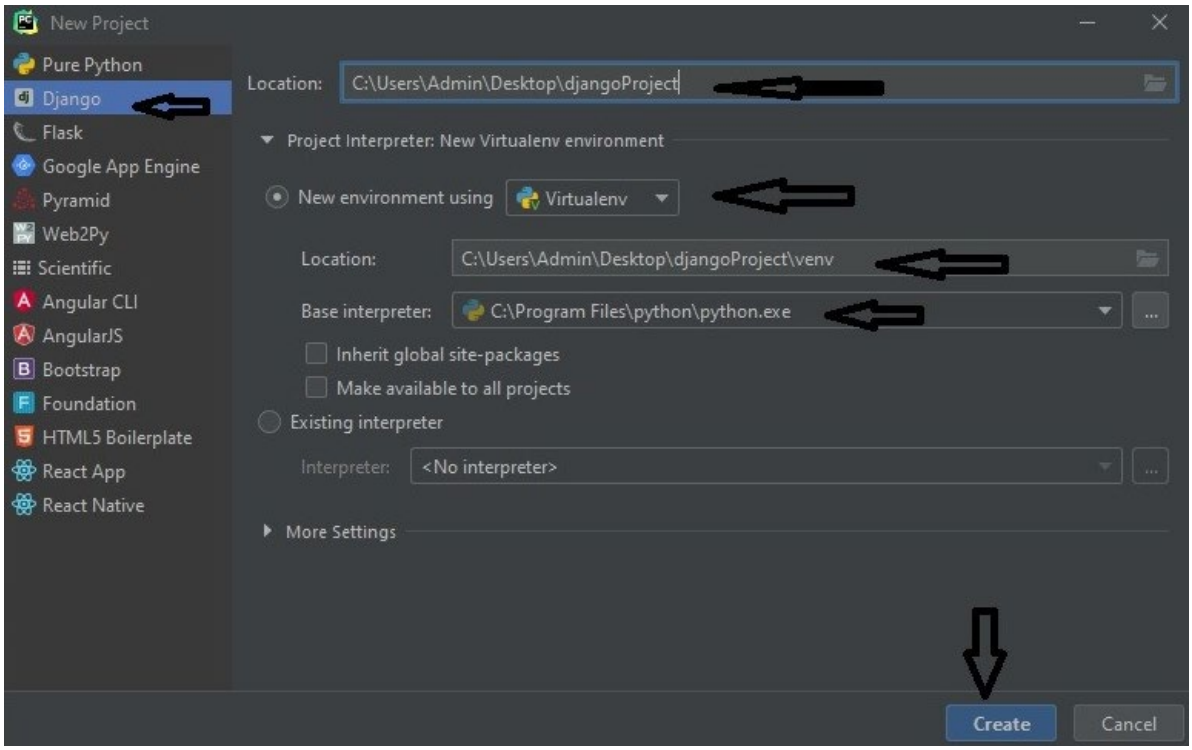

Рис. 3.4. Створення нового проекту Django

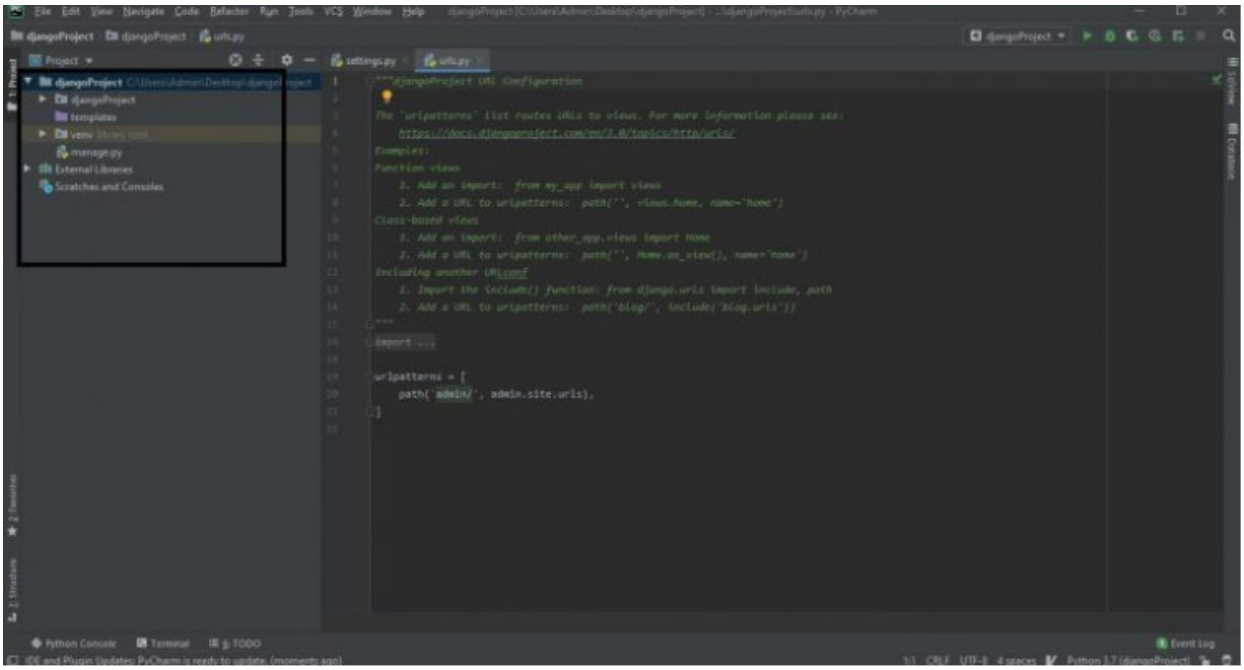

Рис. 3.5. Структура проекту Django

Внутрішня папка *projectDjango* – формує пакет мови Python, що містить модулі, які відносяться до проекту і задають його конфігурацію (ключові настроювання). Назва цього пакету збігається з назвою проекту. У даному пакеті лежать файли:

__init.py__ - порожній файл, що повідомляє Python, що папка, у якій він перебуває, є повноцінним пакетом.

settings.py - модуль із настроюваннями самого проекту. Включає опис конфігурації бази даних проекту, шляхи ключових папок, важливі параметри, пов'язані з безпекою.

urls.py - модуль із маршрутами рівня проекту.

wsgi.py - модуль, що зв'язує проект із веб-сервером. Використається при публікації готового сайту в Інтернеті.

asgi.py - модуль призначений для забезпечення стандартного інтерфейсу між асинхронними веб-серверами Python, фреймворками й додатками.

Пісял цього створюємо необхідні додатки, які і будуть складовими нашого веб-застосунку (додаток А). Після створення застосунку його слід зараєструвати у проекті, що різні утиліти його задіяли, наприклад, для додавання моделей у базу даних.

У Django по замовчуванню установлена база даних SQLite, яка легко налаштовується у проекті, оскільки не потребує окремих бібліотек для її підтримки. Необхідні для цього команди:

python manage.py makemigrations

python manage.py migrate

Потрібно також створити користувача для взаємодії з таблицями бази даних з допомогою команди:

python manage.py createsuperuser

Для роботи з базою даних Django використовує власний ORM (Object – Relational Mapping), у якому модель даних описується класами Python, і по ній генерується схема бази даних. Django підтримує ORM з SQLite, модель дозволяє

ваємодіяти з об’єктами в базі даних за опомогою методів та класів в Django. Це позбавляє від необхідності прописувати SQL запити, Django робить це самостійно.

Основою автосалону є каталог, кожен автомобіль своїм атрибутом має модель, ціну, кількість у наявності, тип двигуна, рік реєстрації, коробка передач, потужність двигуна, пробіг, тип палива та позиціональні поля – опис і зображення. Для зберігання бажаних для користувача характеристик автомобіля виокристовуєтсья внутрішній фреймворк Django – Session. Він зберігає характеристики до тих пір, коли користувач отримає рекомендовані моделі та знаходиться у системі та підтримує анонімні семії й дозволяє зберігати довільні дані для кожного користувача.

Веб-застосунки, написані на Django, звичайно групують код в окремі файли (рис. 3.7).

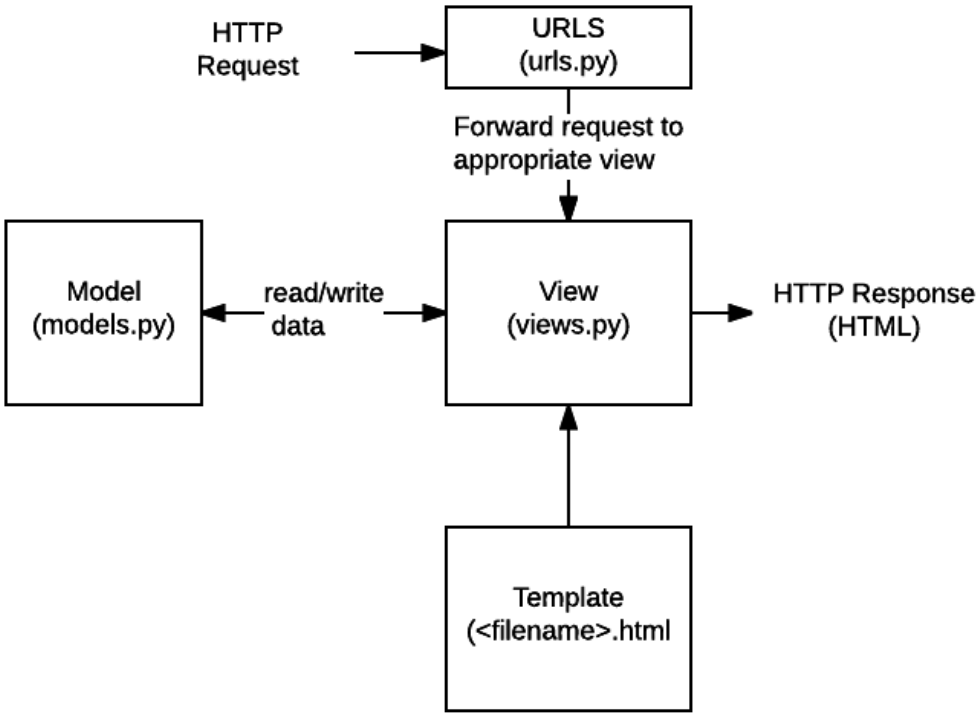

Рис. 3.6. Файли проекту Django

1. *URLs*: Хоча можна обробляти запити з кожної URL-адреси за допомогою однієї функції, набагато зручніше писати окрему функцію для обробки кожного

ресурсу. URL-маршрутизатор використовується для перенапрямку HTTP-запитів у відповідне подання на основі URL-адреси запиту. Крім того, URL-маршрутизатор може витягати дані з URL-адреси відповідно до заданого шаблону й передавати їх у відповідну функцію відображення (view) у вигляді аргументів.

2. *View*: View (англ. "відображення") – це функція оброблювача запитів, що одержує HTTP-запити й повертає відповіді. Функція view має доступ до даних, необхідним для задоволення запитів, і делегує відповіді в шаблони через моделі.

3. *Models*: Моделі є об’єктами Python, які визначають структуру застосунку й надають механізми для керування (додавання, зміни, видалення) і виконання запитів у базу даних.

4. *Templates*: Template (англ. "шаблон") – це текстовий файл, що визначає структуру або розмітку сторінки (наприклад HTML-сторінки), з полями для підстановки, які використаються для висновку актуального вмісту. View може динамічно створювати HTML-сторінки, використовуючи HTML-шаблони й заповнюючи їхніми даними з моделі (model). Шаблон може бути використаний для визначення структури файлів будь-яких типів, не обов'язково HTML.

Діаграма класів програмного продукту зображено на рисунку 3.7.

До складу Django входить веб-сервер, написаний мовою Python, який дозволяє налагоджувати створюваний проект. Для тестування скелету роботи застосунку необхідно запустити міграцію бази даних з використанням ORM. При створенні застосунку Django автоматично додає декілька моделей, які можна використовувати в адмін-панелі.

Для зв’язку з клієнтською частиною потрібні представлення (views) або доступ до ресурсів, які надає сервер які приймають http-запити від веб-клієнтів та повертають http-відповіді. Вони описуються у файлі views.py. Щоб підключити представлення, необхідно додати URL до файлу urls.py. Покроково створюється модель, для неї представлення, яке конфігурується потрібною аутентифікацією та видом запитів через серріалізатори. У результаті є ресурси, які використовуються

клієнтом через POST, GET ,PUT, DELETE запити і отримують результати у JSON форматі.

Після створення сервера з базою даних, від якої надані посилання керування ресурсами, описують відображення цих ресурсів користувачеві черех клієнтський застосунок, реалізований з допомогою HTML та CSS.

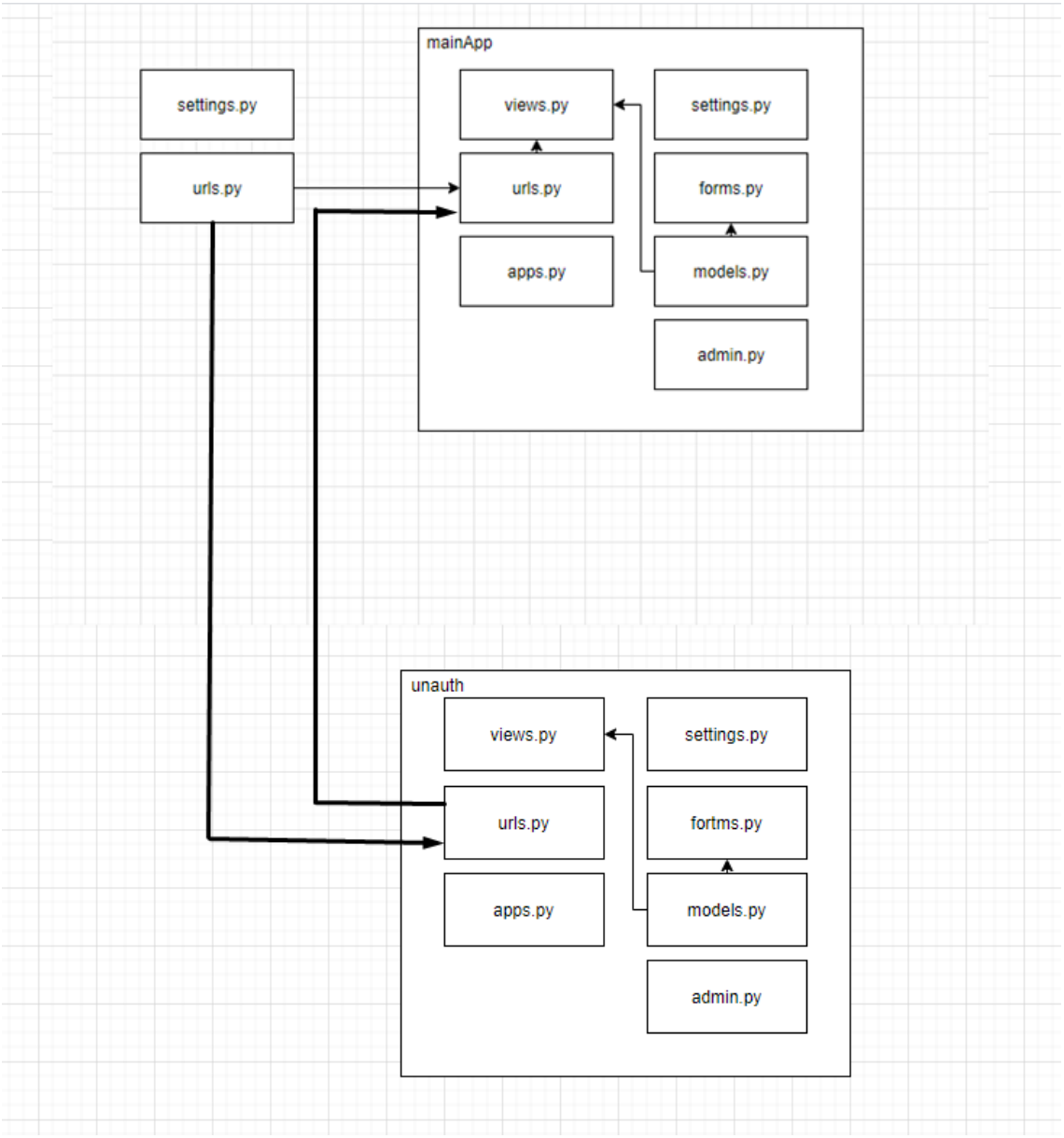

Рис. 3.7. Діаграма класів програмного продукту

Верстка веб-сторінок створена на HTML + CSS з використанням темплейтів. Django має історію з темлейтами. Ідея темплейтів полягає у тому, що у інформаційній системі автосалону, переміщаючись від товару до товару, видно одну й ту ж навігаційну панель (шапку та підвал). Django Template Language

(DTL) дозволяє створювати динамічний HTML, ізолювати елементи, зменшувати кількість рядків, що повторюються та описує представлення, а не програмну логіку.

Тому в нас окремо створений базовий шаблон (меню, хедери) і кожна сторінка являє собою перевизначення блоку "контент". Тобто , фактично, сторінка та сама, міняється тільки її наповнення. Для виведення інформації використаються Django-теги типу {% for el in list %}, які означають, що зміст блоку може бути перевизначено дочірніми темплейтами.

Для реалізації алгоритму k-ближніх сусідів при здійсненні фільтрації та побудові рекомендацій використано бібліотеку машинного навчання Scikit-learn, у якій реалізовано обчислення основних метрик для наборів даних та підтримка дагного алгоритму. За попередню обробку даних у бібліотеці Scikit-learn відповідають класи з пакета *preprocessing*, вхідні дані у якості аргументів приймає метод *fit*, який є у кожній моделі бібліотеки (класу, який її реалізує). У проекті у якості метрики було обрано відстань Евкліда, кількість ближніх сусідів k=5. Це дозволяє визначити 5 найближчих моделей авто до вказаних користувачем характеристик.

3.3. Інтерфейс застосунку

Опишемо інформфейс створеного програмного застосунку.

Розроблена рекомендаційна інформаційна система автосалону складається з одного вікна, яке може міняти стани завдяки викорситанню темплейтів. Спочатку користувач потрапляє на стрінку реєстрації, на якій необхідно ввести дані про себе: електронну пошту, нік, прізвище та ім’я і пароль для входу у систему (рис. 3.8).

Після реєстарції користувачеві пропонують пройти авторизацію (рис. 3.9).

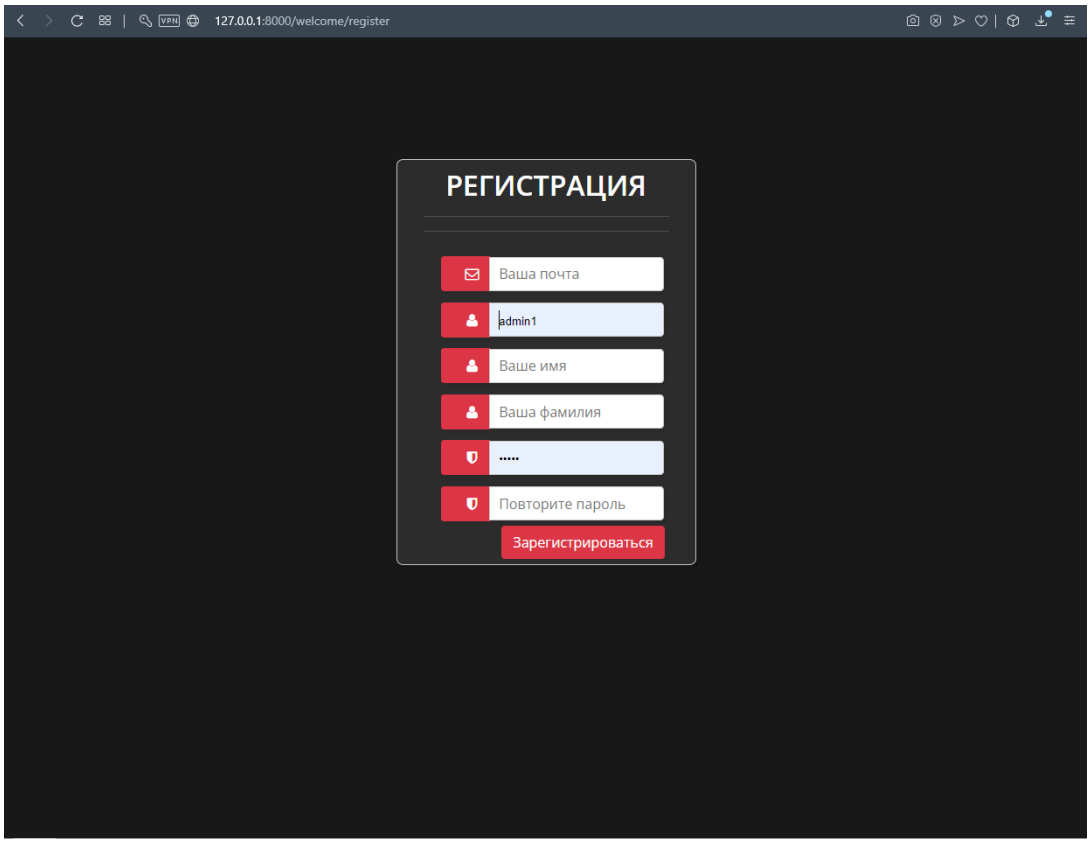

Рис. 3.8. Сторінка реєстрації

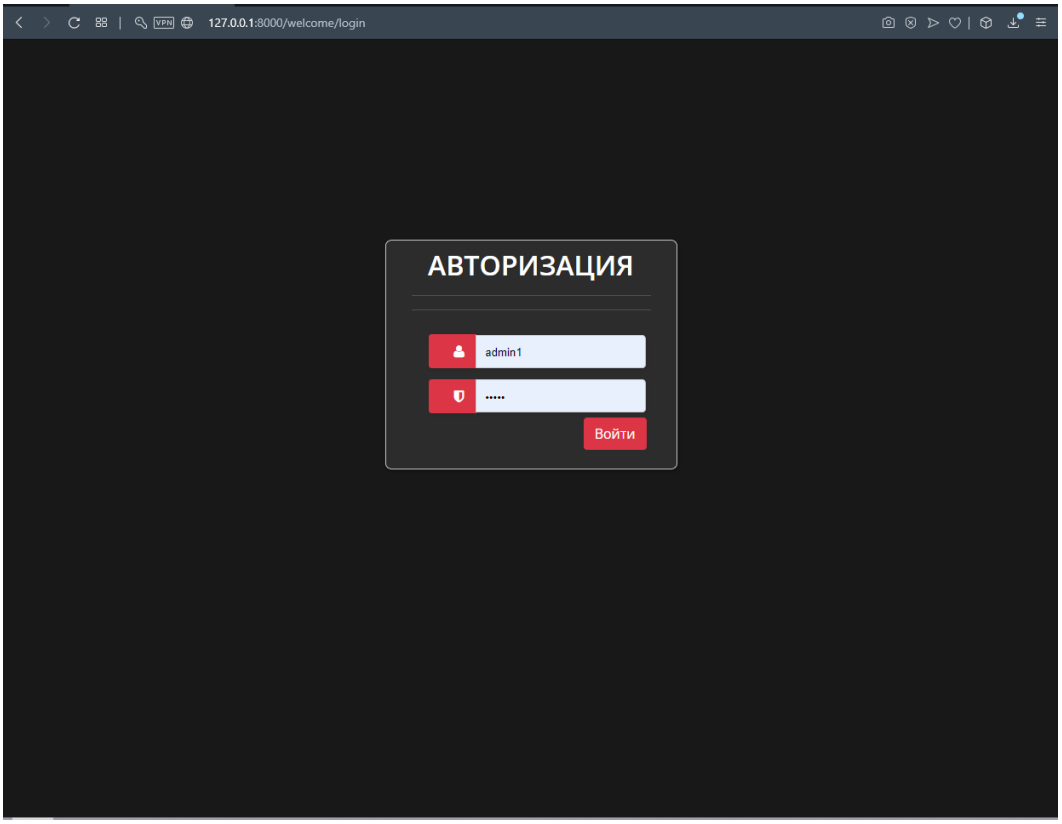

Рисунок 3.9. Сторінка авторизації

Пройшовши авторизацію корисувач потрапляє на головну сторінку застосунку (рис. 3.10).

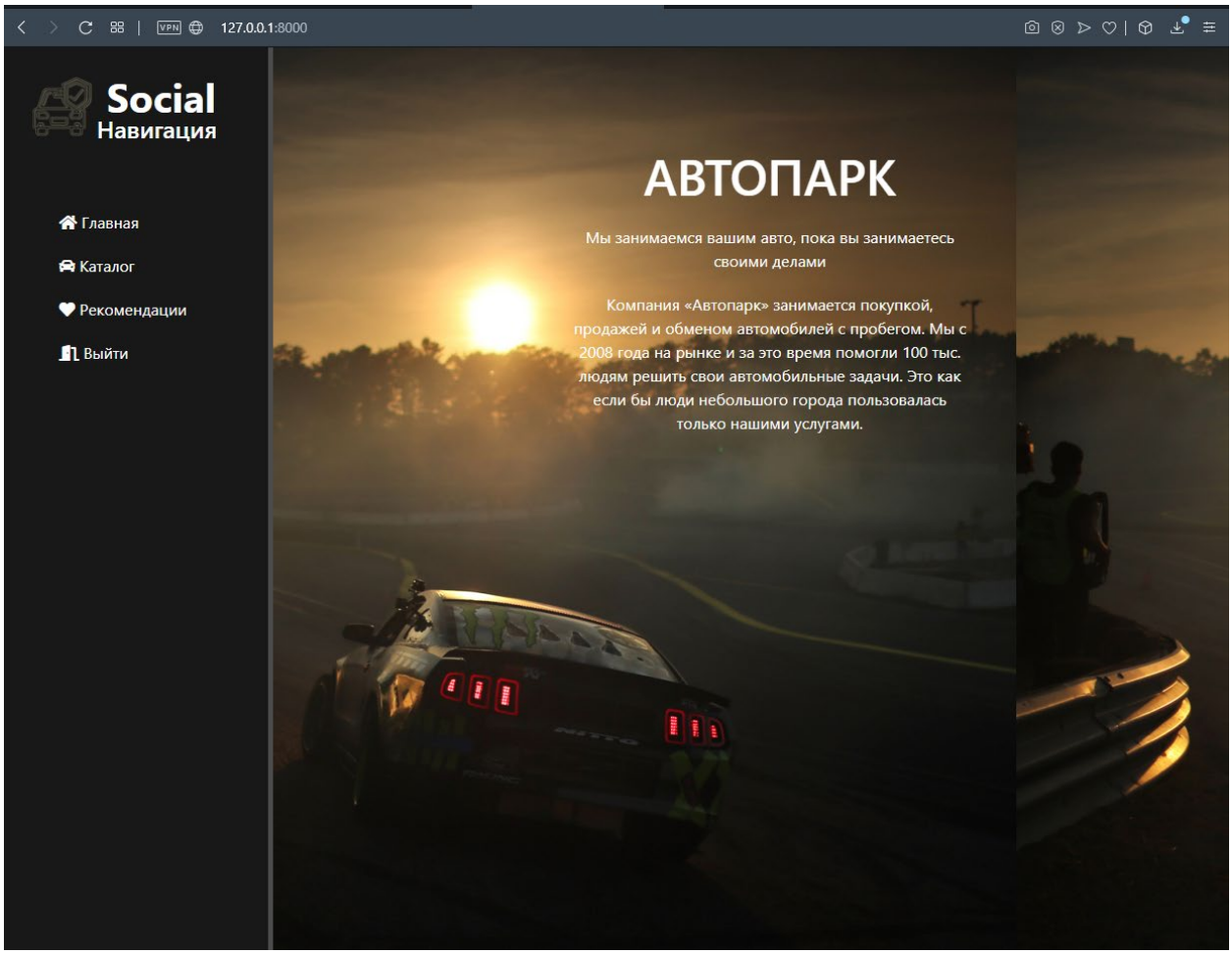

Рисунок 3.10. Головна сторінка

Обравши у панелі зліва на головній сторінці пункт *Каталог*, користувач має можливість для перегляду наявних моделей автомобілів (рис. 3.11).

Обравши у панелі зліва на головній сторінці пункт *Рекомендації*, користувач має можливість перейти до стрінки підбору рекомендацій (рис. 3.12). Ця сторінка містить поля, які дають можливість ввести бажаний діапазон цін (мінімальну та максимальну ціну), рік реєстарції, пробіг. У списках, що розкриваються, є можливість обрати модель автомобіля (рис. 3.13), тип коробки передач (ручна чи автоматична, рис. 3.14) та тип палива (рис. 3.15).

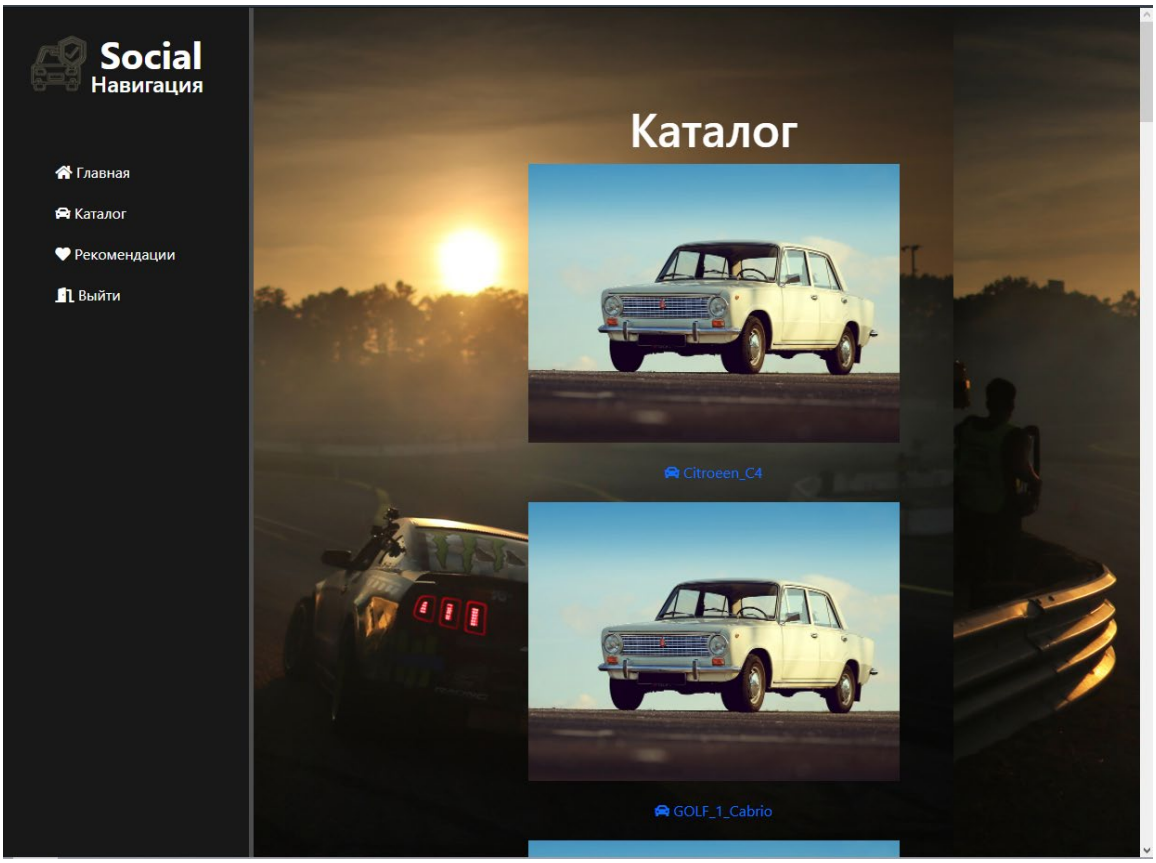

Рисунок 3.11. Сторінка каталогу

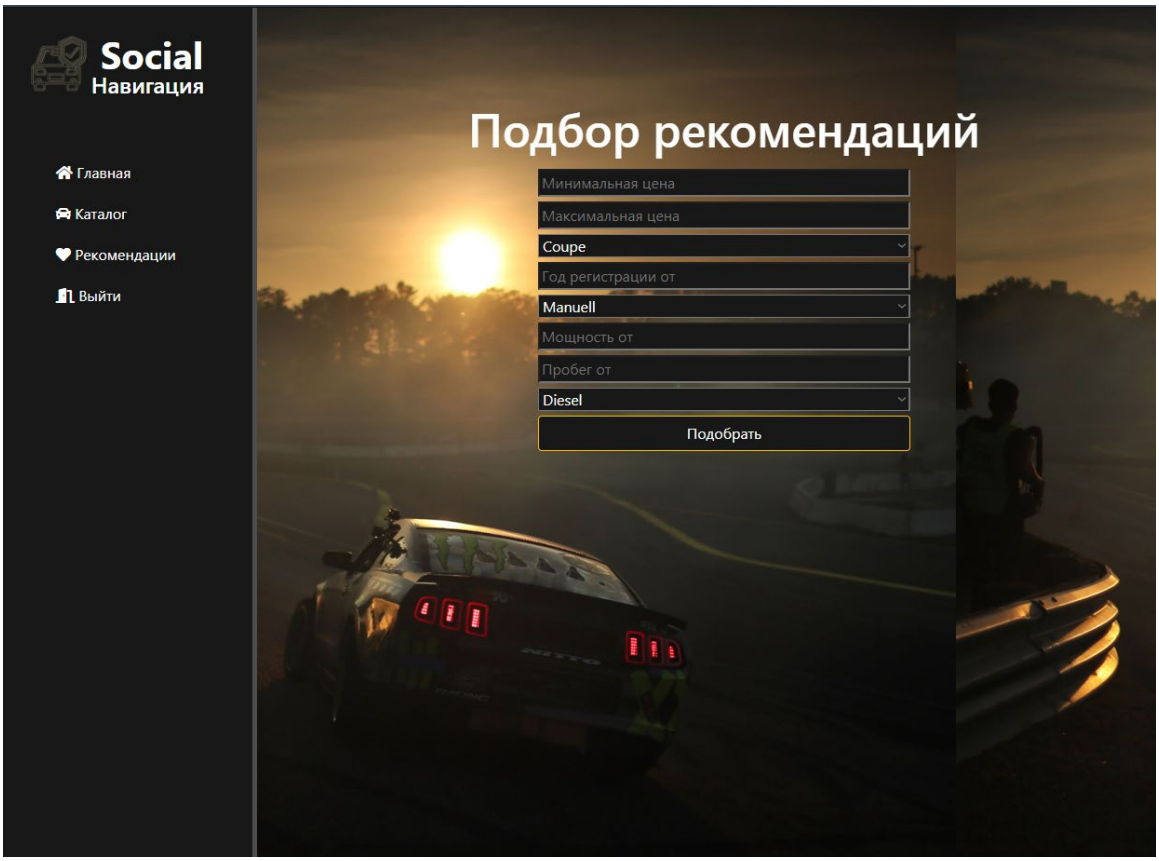

Рисунок 3.12. Сторінка підбору рекомендацій

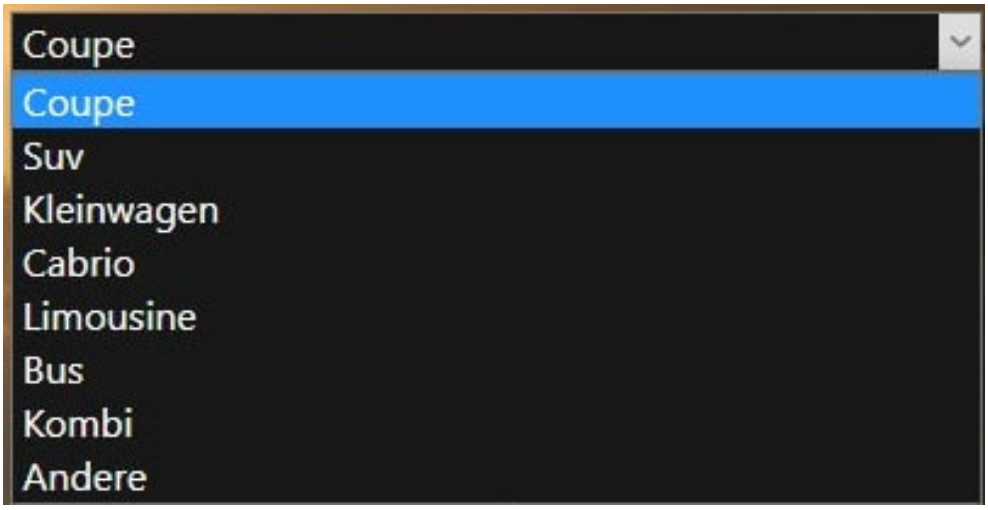

Рисунок 3.13. Список вибору моделі автомобіля

Рисунок 3.14. Список вибору коробки передач

Рисунок 3.15. Список вибору типу палива

Після вказівки бажаних характеристик автомобілів користвач натискає кнопку *Підібрать*, після чого буде відкрите вікно з рекомендованими йому моделями автомобілів (рис. 3.16).

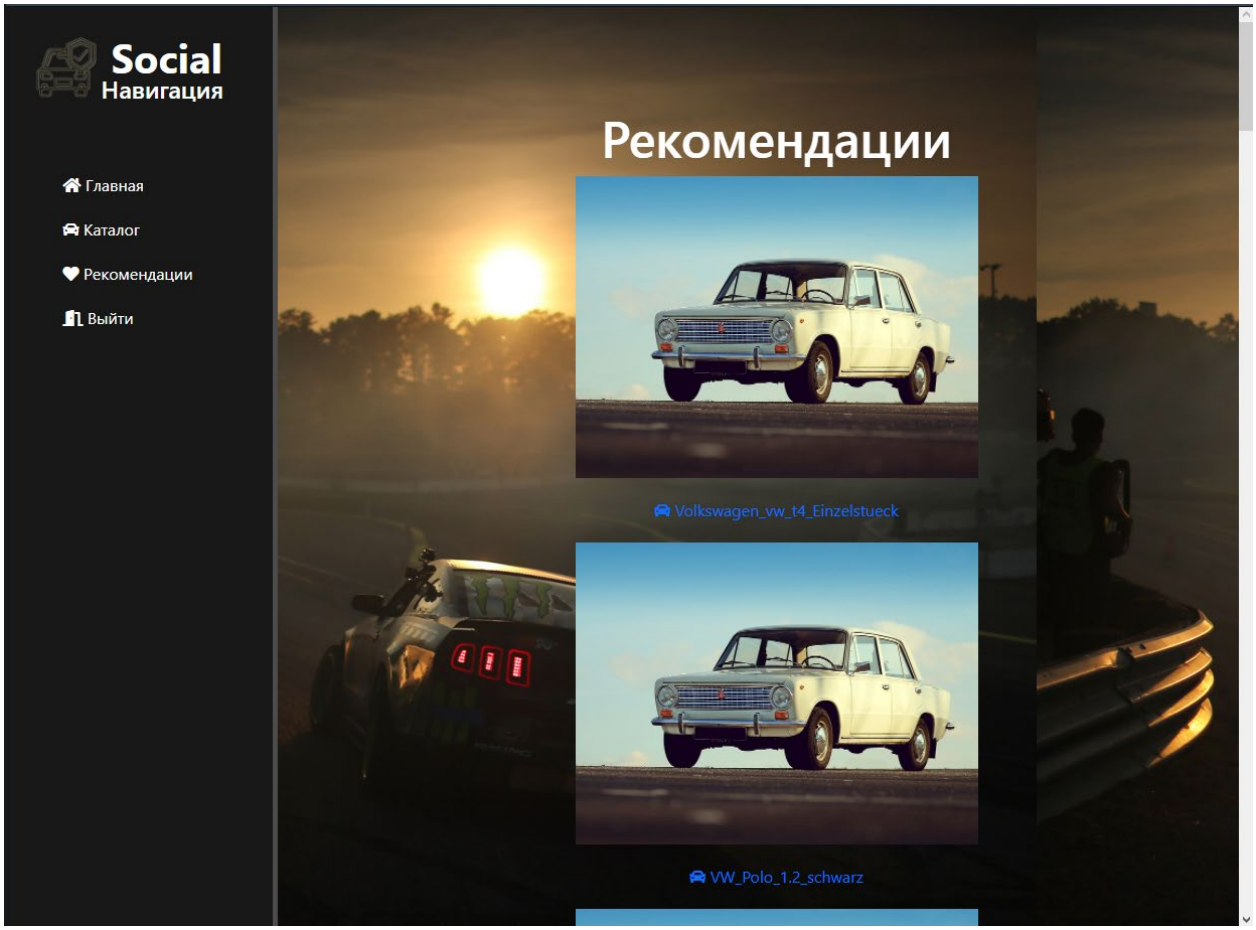

Рисунок 3.16. Сторінка списку рекомендацій

Кожну з рекомендованих моделей можна обрати для більш детального перегляду лівою кнопкою миші (рис. 3.17).

Висновок до розділу 3

Здійснено розробку, програмну реалізацію та тестування рекомендаційної інформаційної системи автосалону на основі колаборативної фільтрації, яка дозволяє вдосконалити та підвищити ефективність обслуговування клієнтів і просування товару в мережі Інтернет, полегшує роботу консультантів і

менеджерів, розширює функціонал інформаційної системи, підвищує якість роботи автосалону й ефективність обслуговування клієнтів.

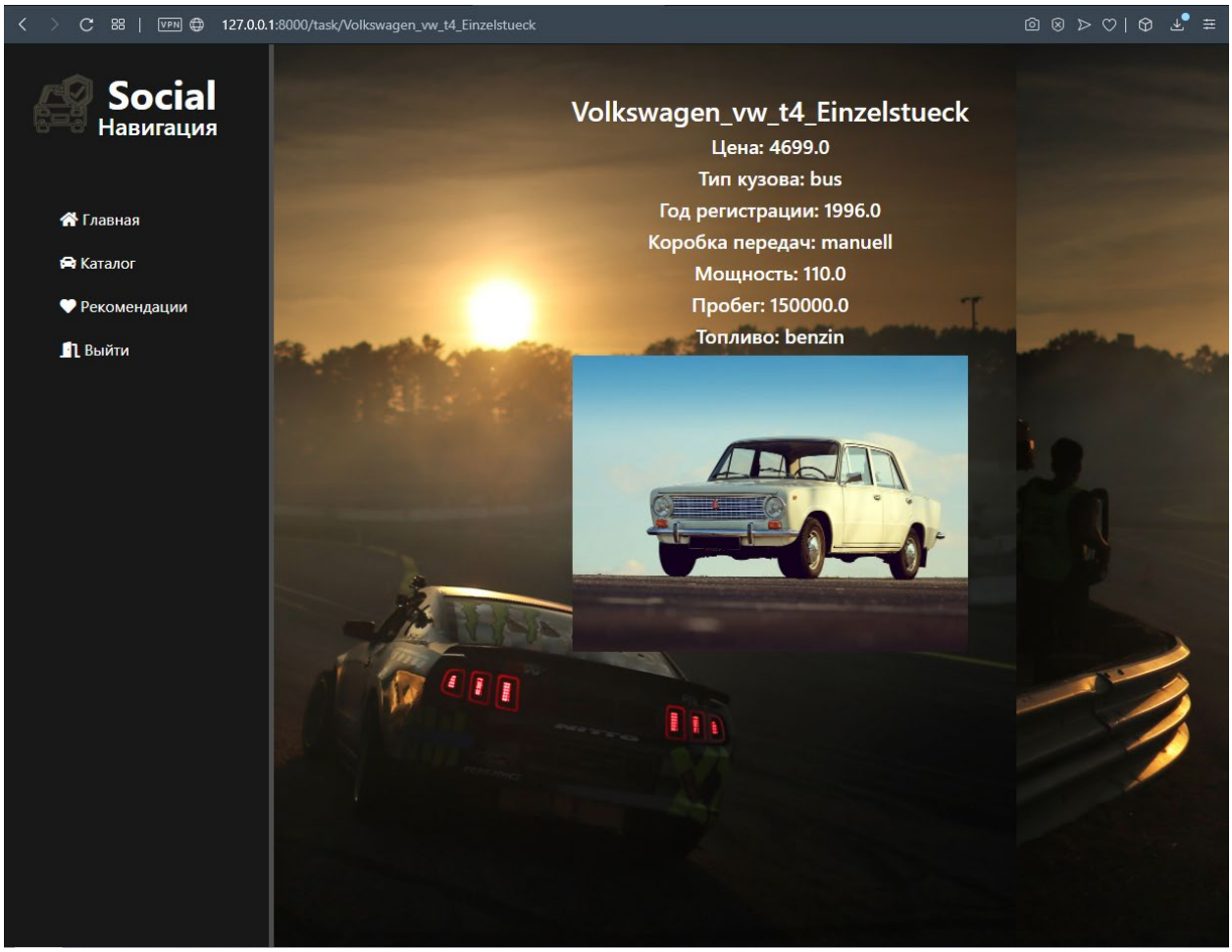

Рисунок 3.17. Сторінка детального перегляду

Кафедра інтелектуальних інформаційних систем
Рекомендаційна інформаційна система автосалону на основі уподобань користувачів

Спеціальний розділ до магістерської кваліфікаційної роботи

Методичний розділ

на тему:

«РЕКОМЕНДАЦІЙНА ІНФОРМАЦІЙНА СИСТЕМА АВТОСАЛОНУ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧІВ»

Спеціальність 124 «Системний аналіз»

124 – МКР – 607. 21830801

Студент _____ Бурлака І.І.
«__» _____ 2022 р.

Консультант _____ Болюбаш Н. М.
канд. пед. наук, доцент
«__» _____ 2022 р.

Миколаїв – 2022

Спеціальний розділ до магістерської кваліфікаційної роботи

Охорона праці та безпека у надзвичайних ситуаціях

на тему:

«РЕКОМЕНДАЦІЙНА ІНФОРМАЦІЙНА СИСТЕМА АВТОСАЛОНУ НА ОСНОВІ УПОДОБАНЬ КОРИСТУВАЧІВ»

Спеціальність 124 «Системний аналіз»

124 – МКР – 607. 21830801

Студент _____ Бурлака І.І.
«__» _____ 2022 р.

Консультант _____ Григор'єва Л. І.
д.б.н., професор
«__» _____ 2022 р.

Миколаїв – 2022

ВИСНОВКИ

У результаті проведеного дослідження виявлено, що продаж автомобілів останнім часом зростає та стає все більш Інтернет-орієнтованим. Діяльність авторинків та автосалонів розширюється за рахунок впровадження веб-ресурсів, якими широко користуються як покупці, так і ті, хто продає автомобілі та запчастини до них. До найбільш популярних в Україні ресурсів, які надають інформацію про автомобілі та їх характеристики і містять відгуки про рівень обслуговування й сервісу відносять: AvtoBazar, Auto.Ria, RST, AutoSite, сайти-агрегатори Auto.Meta, AutoMoto, UAvto, AvtoPoisk, AutoSale. Кращі веб-сайти автосалонів – Towne Ford, Santa Margarita Toyota, “Фалькон-Авто”, “Ю.Р.К”, орієнтовані на надання допомоги покупцям у знаходженні автомобілів, які задовольняють їх потреби, використовуючи при цьому перш за все інформаційну та маркетингову функції. Однак аналіз накопиченої інформації шляхом її фільтрації з метою вивчення поведінки споживачів та їх уподобань і потреб, представлений недостатньо. Тому є потреба у створенні рекомендаційної інформаційної системи автосалону, де передбачено автоматизований збір та аналіз інформації у процесі її функціонування й надання рекомендацій користувачам для прийняття рішення про вибір автомобіля.

Установлено, що рекомендаційна система – це система, що будує рейтинг автомобілів серед великого набору даних, на підґрунті вже існуючої інформації про уподобання користувачів при виборі моделей авто та супутніх товарів і послуг. Виявлено, що у більшості рекомендаційних систем використовуються методи контентної фільтрації, колаборативної фільтрації та гібридної фільтрації, яка передбачає їх поєднання. Контентна фільтрація будує рекомендації, базуючись на інформації про поведінку користувача та його потреби й уподобання. Колаборативна фільтрація будує персональні рекомендації, базуючись на моделі поведінки користувача на основі попередньо зібраної інформації про поведінку інших користувачів із схожими вподобаннями або

характеристиками. Серед алгоритмів колаборативної фільтрації виділяють алгоритми, що базуються на даних користувачів та алгоритми, що базуються на даних предметів. Гібридна фільтрація поєднує сильні сторони контентної та колаборативної фільтрації, однак є складнішою у розробці та підтримці.

Метод колаборативної фільтрації не вимагає конкретизованих запитів, тому для реалізації рекомендаційної інформаційної системи автосалону доцільно взяти за основу метод колаборативної фільтрації, який базується на даних, що містять характеристики автомобілів. Найбільш достовірним підходом до отримання початкових даних є явний зворотній зв'язок, коли інформація отримується від користувача через системний інтерфейс.

Основними задачами, які має вирішувати рекомендаційна система автосалону, є пошук користувачем автомобілів за вказаними характеристиками та надання йому рекомендацій з вибору. Від користувача система отримує вектор бажаних характеристик: $U = \{u_1, u_2, ..., u_n\}$, де n – кількість характеристик. Ці дані необхідно порівняти з векторами характеристик автомобілів $I_k = \{i_1, i_2, ..., i_n\}$ (I_k – вектор характеристик k-го автомобіля), які містить система та найти ті з них, які є найбільш близькими до запиту користувача. Для вибору найбільш підходящих автомобілів було вирішено застосувати метод k-ближніх сусідів, розрахунок мір близькості здійснювати шляхом обчислення відстані Евкліда.

Для розробки рекомендаційної інформаційної системи автосалону було використано мову програмування Python, інтегроване середовище розробки PyCharm, яке забезпечує аналіз коду, має графічний налагоджувач, інтегрований модуль тестування та інтеграцію з системами контролю версій, інструменти для роботи з базами даних, підтримує фреймворк Django, який тиакож було використано, і платформи для розробки на Python та бібліотеки для наукових обчислень Pandas, Numpy, Matplotlib. Для реалізації алгоритмів фільтрації рекомендаційної системи було використано бібліотеку машинного навчання Scikit-learn.

Зберігання даних було реалізоване з використанням реляційної бази даних SQLite, яка не використовує парадигму клієнт-сервер, а є бібліотекою, з якою програма компонується й зберігає всю базу даних (включаючи таблиці, індекси й дані) у єдиному файлі на там, де виконується програма.

Здійснено розробку, програмну реалізацію та тестування рекомендаційної інформаційної системи автосалону на основі колаборативної фільтрації яка дозволяє вдосконалити та підвищити ефективність обслуговування клієнтів і просування товару в мережі Інтернет, розширює функціонал інформаційної системи, підвищує якість роботи автосалону, ефективність обслуговування клієнтів та полегшує роботу консультантів і менеджерів.

Поставлені завдання виконано повністю, однак функціонал розробленого застосунку може бути розширений у подальшому шляхом надання можливості вибору мір близькості з метою виявлення тих, які дають кращий прогноз при виборі моделі автомобіля, застосування пошуку асоціативних правил для знаходження рекомендацій.

СПИСОК ПОСИЛАНЬ

1. Meleshko E. Дослідження методів побудови рекомендаційних систем в мережі Інтернет / E. Meleshko, S. Semenov, V. Khokh // Системи управління, навігації та зв'язку. Збірник наукових праць. – Полтава: ПНТУ, 2018. – Т. 1 (47). – С. 131-136.
2. Лобур М. Побудова асоціативних правил для прогнозування рекомендацій в колаборативних рекомендаційних системах / М. Лобур, Ю. Стех, М. Шварц // Збірник наукових праць УАД. – Львів, 2017. − № 2 (32). − С. 82−86.
3. Schedl M, Zamani H, Chen CW, Deldjoo Y, Elahi M. Current challenges and visions in music recommender systems research. Int J Multi Inform Ret. (2018) 7:95–116. doi: 10.1007/s13735-018-0154-2
4. Schedl M, Gómez E, Urbano J. Music information retrieval: recent developments and applications. Found Trends Inform Ret. (2014) 8:127–261. doi: 10.1561/1500000042
5. Downie JS. Music information retrieval. Ann Rev Inform Sci Techn. (2003) 37:295–340. doi: 10.1002/aris.1440370108
6. Dorfer M, Henkel F, Widmer G. Learning to listen, read, and follow: score following as a reinforcement learning game. In: Proceedings of the 19th International Society for Music Information Retrieval Conference (ISMIR 2018), Paris (2018). p. 784–91.
7. Chou PW, Lin FN, Chang KN, Chen HY. A simple score following system for music ensembles using chroma and dynamic time warping. In: Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval (ICMR 2018), Yokohama: ACM (2018). p. 529–32. doi: 10.1145/3206025.3206090
8. Goto M, Dannenberg RB. Music interfaces based on automatic music signal analysis: new ways to create and listen to music. IEEE Signal Proc Mag. (2019) 36:74–81. doi: 10.1109/MSP.2018.2874360

9. Schedl M. Intelligent user interfaces for social music discovery and exploration of large-scale music repositories. In: Proceedings of the 22nd ACM International Conference on Intelligent User Interfaces (IUI 2017): Workshop on Theory-Informed User Modeling for Tailoring and Personalizing Interfaces (HUMANIZE 2017). Limassol: ACM (2017). p. 7–11. doi: 10.1145/3039677.3039678
10. Oramas S, Nieto O, Barbieri F, Serra X. Multi-label music genre classification from audio, text and images using deep features. In: Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR), Suzhou (2017). p. 23–30.
11. Mayer R, Rauber A. Music genre classification by ensembles of audio and lyrics features. In: Proceedings of the 12th International Society for Music Information Retrieval Conference (ISMIR 2011). Miami, FL (2011). p. 675–80.
12. Sturm BL. Classification accuracy is not enough: on the evaluation of music genre recognition systems. J Intell Inform Syst. (2013) 41:371–406. doi: 10.1007/s10844-013-0250-y
13. Yang YH, Chen HH. Machine recognition of music emotion: a review. Trans Intell Syst Techn. (2013) 3:40. doi: 10.1145/2168752.2168754
14. Huq A, Bello JP, Rowe R. Automated music emotion recognition: a systematic evaluation. J New Music Res. (2010–11) 39:227–44. doi: 10.1080/09298215.2010.513733
15. Knees P, Schedl M. Music similarity and retrieval — an introduction to audio- and web-based strategies. Berlin; Heidelberg: Springer (2016).
16. Karydis I, Lida Kermanidis K, Sioutas S, Iliadis L. Comparing content and context based similarity for musical data. Neurocomputing. (2013) 107:69–76. doi: 10.1016/j.neucom.2012.05.033
17. Schedl M, Knees P, McFee B, Bogdanov D, Kaminskas M. Music recommender systems. In: Ricci F, Rokach L, Shapira B, Kantor PB, editors. Recommender Systems Handbook, 2nd ed. Boston, MA: Springer (2015). p. 453–92.

18. van den Oord A, Dieleman S, Schrauwen B. Deep content-based music recommendation. In: Burges C, Bottou L, Welling M, Ghahramani Z, Weinberger K, editors. Advances in Neural Information Processing Systems 26 (NIPS). Lake Tahoe, NV: Curran Associates, Inc. (2013). p. 2643–51.
19. Bertin-Mahieux T, Ellis DPW, Whitman B, Lamere P. The million song dataset. In: Proceedings of the 12th International Society for Music Information Retrieval Conference. Miami, FL (2011). p. 591–6.
20. Xiaoyuan Su, Taghi M. Khoshgoftaar, A survey of collaborative filtering techniques, Advances in Artificial Intelligence archive, 2009.
21. Recommender Systems - The Textbook | Charu C. Aggarwal | Springer. Springer. 2016. ISBN 9783319296579.
22. "virtual (C# Reference)". docs.microsoft.com.
23. "Installation guidance for SQL Server on Linux". December 21, 2017. Retrieved February 1, 2018.
24. "SQL Server 2008 R2 Express Database Size Limit Increased to 10GB"
25. "What's up with SQL Server 2008 Express editions"
26. Kalen Delaney (2007). Inside Microsoft SQL Server 2005: The Storage Engine
27. "Pages and Extents" (http://msdn.microsoft.com/en-us/library/ms190969.aspx)
28. "Table and Index Organization" (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/ms189051(v=sql.105)?redirectedfrom=MSDN)
29. "Buffer Management" (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2008-r2/aa337525(v=sql.105)?redirectedfrom=MSDN)
30. "Transact-SQL Reference" (https://docs.microsoft.com/en-us/previous-versions/sql/sql-server-2005/ms189826(v=sql.90)?redirectedfrom=MSDN)

Додаток А

Лістинг тпрограмного коду

forms.py

%Створення форми вибору характеристик автомобіля для підбору рекомендацій

```
from django import forms
class SimpleRecomendForm(forms.Form):
    vehicleTypeChoises = (
        ("coupe", u"Coupe"),
        ("suv", u"Suv"),
        ("kleinwagen", u"Kleinwagen"),
        ("cabrio", u"Cabrio"),
        ("limousine", u"Limousine"),
        ("bus", u"Bus"),
        ("kombi", u"Kombi"),
        ("andere", u"Andere")
    )

    gearboxChoises = (
        ('manuell', u'Manuell'),
        ('automatik', u'Automatik')
    )

    fuelTypeChoises = (
        ('diesel',u'Diesel'),
        ('benzin', u'Benzin')
    )

    price_min = forms.CharField(label='Минимальная цена',
widget=forms.TextInput(attrs={'placeholder': 'Минимальная цена'}))
```

```
    price_max = forms.CharField(label='Максимальная цена',
widget=forms.TextInput(attrs={'placeholder': 'Максимальная цена'}))
    vehicleType = forms.ChoiceField(label='Тип двигателя', choices=vehicleTypeChoises)
    yearOfRegistration = forms.CharField(label='Год регистрации от',
                                         widget=forms.TextInput(attrs={'placeholder': 'Год регистрации
от'}))
    gearbox = forms.ChoiceField(label='Коробка передач', choices=gearboxChoises)
    powerPS = forms.CharField(label='Мощность от',
widget=forms.TextInput(attrs={'placeholder': 'Мощность от'}))
    kilometer = forms.CharField(label='Пробег от',
widget=forms.TextInput(attrs={'placeholder': 'Пробег от'}))
    fuelType = forms.ChoiceField(label='Топливо', choices=fuelTypeChoises)

    class Meta:
        fields = ['price_min',
                  'price_max',
                  'vehicleType',
                  'yearOfRegistration',
                  'gearbox',
                  'powerPS',
                  'kilometer',
                  'fuelType',
                  ]
```

urls.py

%Відповідає за механізм урл-роутів

```
from django.urls import path, include from . import views
app_name = "main"
urlpatterns = [
    path('', views.index, name = 'index'),
```

```
    path('logout/', views.logoutUser, name='logout'),
    path('opened/', views.openedTasks, name='opened'),
    path('closed/', views.closedTasks, name='closed'),
    path('new_task/', views.newTask, name='new_task'),
    path(u'task/<id>', views.taskPage, name='task'),
]
```

models.py

%Створення таблиці БД і подальшого використання у зручному форматі

```
from django.db import models
from django.contrib.auth.models import User

class Car(models.Model):
    name = models.TextField('Наименование', null = True)
    price = models.TextField('Цена', null = True)
    vehicleType = models.TextField('Тип кузова', null = True)
    yearOfRegistration = models.TextField('Год регистрации', null = True)
    gearbox = models.TextField('Коробка передач', null = True)
    powerPS = models.TextField('Мощность', null = True)
    kilometer = models.TextField('Пробег', null = True)
    fuelType = models.TextField('Топливо', null = True)

    def str(self):
        return self.name

    class Meta:
        verbose_name = 'Машина'
        verbose_name_plural = 'Машины'
```