Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту
(повна назва)
Кафедра Інформатики
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

РОЗРОБЛЕННЯ БОТА В TELEGRAM ДЛЯ КУПІВЛІ АВТОМОБІЛЯ
(тема)

Виконав:
здобувач 4 року навчання,
групи ІТІНФ-22-3
Гергуль Є. О.
(прізвище, ініціали)

Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва освітньої програми)

Керівник проф.Гороховатський В.О.
(посада, прізвище, ініціали)

Допускається до захисту

Завідувач кафедри інформатики _____ Кобилін О. А.
(підпис) (прізвище, ініціали)

2026 р.

Харківський національний університет радіоелектроніки

Факультет Інформаційно-аналітичних технологій та менеджменту

Кафедра Інформатики

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Інформатика
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Гергулю Євгенію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення бота в Telegram для купівлі автомобіля

затверджена наказом університету від 26 травня 2026 року № 435Ст
2. Термін подання здобувачем роботи до екзаменаційної комісії 23 червня 2026 р.
3. Вихідні дані до роботи технічне завдання на месенджер-бота для підбору автомобілів, методи вебскрапінгу та алгоритми багатокритеріального ранжування, інструментарій (Python 3.11, aiogram, requests, BeautifulSoup), СУБД SQLite.

4. Перелік питань, що потрібно опрацювати в роботі аналіз предметної області та недоліків вебагрегаторів, порівняльний аналіз методів збирання даних і архітектур ботів, обґрунтування евристичного ранжування, ОО-моделювання прецедентів та проєктування бази даних, розроблення підсистеми вебскрапінгу, реалізація інтерфейсу Telegram-бота (FSM), тестування системи.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) класифікація чинників вибору автомобіля, схеми трансформації запиту та архітектури вебскрапінгу, блок-схеми анкетування, вилучення даних і ранжування, UML-діаграма прецедентів, ER-діаграма бази даних, діаграми класів, компонентів, станів, діяльності, розгортання та скріншоти інтерфейсу бота.

6. Консультанти розділів роботи (п.6 включається до завдання за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	13.04.2026	
2	Аналіз завдання, підбір літератури	14.04.26-16.04.26	
3	Аналіз літератури з проблеми, що вирішується	17.04.26-19.04.26	
4	Аналіз існуючих методів розпізнавання	20.04.26-22.04.26	
5	Формування кроків вирішення проблеми	23.04.26-05.05.26	
6	Програмна реалізація	26.04.26-20.05.26	
7	Аналіз отриманих результатів	21.05.26-04.06.26	
8	Оформлення пояснювальної записки	05.06.26-09.06.26	
9	Перевірка на нормоконтроль	10.06.26-19.06.26	
10	Перевірка на плагіат	11.06.26-30.06.26	
11	Рецензування	20.06.26-30.06.26	
12	Підготовка презентації та доповіді	20.06.26-30.06.26	
13	Занесення роботи в електронний архів	30.06.26-09.07.26	
14	Попередній захист кваліфікаційної роботи	30.06.26-09.07.26	

Дата видачі завдання 13 квітня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ проф.Гороховатський В.О.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ/ABSTRACT

Пояснювальна записка до кваліфікаційної роботи: 78 с., 11 табл., 29 рис., 54 джерела.

АВТОМАТИЗОВАНЕ ЗБИРАННЯ ДАНИХ, БАГАТОКРИТЕРІАЛЬНЕ РАНЖУВАННЯ, ЕКСПЕРТНА СИСТЕМА, РОЗМОВНИЙ ІНТЕРФЕЙС, ТЕЛЕГРАМ-БОТ, ВЕБАГРЕГАТОР, PYTHON.

Об’єктом роботи є процес автоматизованого пошуку та персоналізованого підбору транспортних засобів на основі даних автомобільних вебагрегаторів.

Розроблено Telegram-бот, що забезпечує автоматизоване збирання актуальних оголошень з вебагрегаторів та здійснює персоналізований підбір автомобілів на основі алгоритмів багатокритеріального ранжування.

Для формування бази даних застосовано методи вебскрапінгу, а для рекомендацій – адаптивну систему нарахування балів залежно від потреб користувача.

Впровадження розробленого програмного рішення дозволяє суттєво зменшити інформаційне перевантаження та оптимізувати часові витрати під час вибору автомобіля.

Область застосування: персональні цифрові помічники для вибору транспортних засобів, агрегація оголошень, консалтинг у сфері авторинку.

AUTOMATED DATA COLLECTION, CAR SELECTION, CONVERSATIONAL INTERFACE, EXPERT SYSTEM, MULTI-CRITERIA RANKING, PYTHON, TELEGRAM BOT.

The object of the work is the process of automated search and personalized selection of vehicles based on data from automotive web aggregators.

A Telegram bot has been developed that provides automated collection of current advertisements from web aggregators and personalized car selection based on multi-criteria ranking algorithms.

Web scraping methods were applied to form the database, and an adaptive scoring system was used for recommendations depending on user needs.

The implementation of the developed software solution significantly reduces information overload and optimizes the time spent by users when choosing a car.

Applications: personal digital assistants for vehicle selection, advertisement aggregation, consulting in the automotive market.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

БД – база даних

СУБД – система управління базами даних

API – Application Programming Interface (інтерфейс прикладного програмування)

DOM – Document Object Model (об’єктна модель документа)

FSM – Finite State Machine (скінченний автомат (машина станів))

HTML – HyperText Markup Language (мова гіпертекстової розмітки)

HTTP – HyperText Transfer Protocol (протокол передавання гіпертексту)

JSON – JavaScript Object Notation (текстовий формат обміну даними)

SQL – Structured Query Language (структурована мова запитів)

UML – Unified Modeling Language (уніфікована мова моделювання)

UX – User Experience (досвід користувача)

ВСТУП

Сучасний автомобільний ринок характеризується стрімким збільшенням обсягів пропозицій на різноманітних мережевих майданчиках, що створює значне інформаційне перевантаження для потенційних покупців. В умовах такого перенасичення інформаційного простору користувачі все частіше стикаються із проблемою неефективності класичних механізмів пошуку. Наявні сервіси консолідації оголошень зазвичай застосовують сувору булеву логіку фільтрування за базовими параметрами (марка, рік, ціна), зовсім не враховуючи індивідуальні сценарії експлуатування транспортного засобу. Це призводить до значних часових витрат користувачів на ручне опрацьовування сотень нерелевантних оголошень та суттєво знижує загальну ефективність підбирання необхідного варіанта [1].

Стрімкий розвиток технологій штучного інтелекту, розмовних середовищ взаємодії та систем автоматизованого вилучення даних з інтернет-сторінок відкриває нові можливості для автоматизування цих рутинних аналітичних процесів. Оскільки популярні програми миттєвого обміну повідомленнями стали основним комунікаційним середовищем для більшості користувачів, інтегрування інтелектуальних експертних систем у такі платформи є логічним та прогресивним кроком. Воно дозволяє створити інтуїтивний, доступний та персоналізований інструмент взаємодії, який мінімізує бар'єри між користувачем та складною аналітичною системою, перетворюючи рутинний процес вибирання на зручне інтерактивне спілкування [2].

Актуальність цього дослідження зумовлена гострою потребою у подоланні розриву між технічними можливостями сучасних пошукових систем та реальними потребами покупців на динамічному вторинному ринку автомобілів. Створення інтелектуального програмного рішення у формі діалогового робота, здатного автоматизовано акумулювати дані з відкритих

джерел та застосовувати багатокритеріальні алгоритми ранжування, є важливим науково-практичним завданням. Воно безпосередньо відповідає сучасним тенденціям розвитку галузі інформаційних технологій, де акцент зміщується з пасивного пошуку на активне надання інтелектуальних рекомендувань. Такий підхід дає змогу не лише скоротити час пошуку, а й підвищити якість прийняття рішень за рахунок глибокого аналізу ринкових пропозицій та їх адаптування під конкретні сценарії використання автомобіля [3].

Науково-методична складність реалізування такої системи полягає у необхідності формалізування суб'єктивних уподобань користувача та їх перетворення у математичні параметри для оптимізування. Інтегрування методів багатокритеріального прийняття рішень (MCDA), зокрема аналізу ієрархій чи зваженої суми, дозволяє трансформувати нечіткі запити покупця у вектор пріоритетів. Це перетворює звичайну діалогову програму із простого засобу збору інформації на повноцінну інтелектуальну систему підтримання прийняття рішень, здатну збалансувати суперечливі вимоги (наприклад, мінімальну ціну, максимальну надійність та оптимальні технічні характеристики) й сформувати персоналізований рейтинг альтернатив у реальному часі.

Втілення такого рішення вимагає комплексного підходу: від детального аналізу предметної області та дослідження чинників вибору автомобілів до проєктування гнучної системної архітектури. Порівняльний аналіз технологій автоматизованого збирання даних, систем збереження інформації та методів побудови розмовних середовищ дозволяє закласти надійний фундамент для практичного втілення системи. Кінцевим результатом є інтегрований програмний комплекс, який поєднує підсистему збирання даних у реальному часі та математичну модель рекомендування, що забезпечує користувачеві релевантний, точний та швидкий підбір транспортного засобу безпосередньо у звичному для нього середовищі взаємодії [4].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Логіка вибору автомобіля

Процес вибору автомобіля є складною багатокритеріальною задачею, що вимагає від користувача врахування значної кількості різноманітних параметрів. Насамперед розглядаються базові характеристики транспортного засобу. До них належать вартість, рік випуску та загальний показник надійності моделі. Економічний чинник відіграє визначальну роль, оскільки передбачає оптимізацію бюджету. Це означає пошук найкращого співвідношення ціни та якості в межах визначеного фінансового ліміту [5]. Окрім початкової вартості, користувач бере до уваги експлуатаційні витрати, серед яких виділяють витрату пального та вартість регулярного технічного обслуговування.

Поряд з об’єктивними метриками, суттєвий вплив на ухвалення рішення мають сценарії використання транспортного засобу. Вибір автомобіля найчастіше продиктований конкретною життєвою метою [6]. Серед типових сценаріїв виділяють щоденні поїздки містом, забезпечення комфорту для великої родини, часті подорожі трасою або максимальну економію коштів. Для візуалізації комплексності цього процесу побудовано деревоподібну структуру основних чинників, що впливають на вибір (рис. 1.1). Аналіз цих чинників є критично важливим для розуміння принципів роботи ефективних систем рекомендування. При проєктуванні таких структур слід враховувати принципи ергономіки інтерфейсів [7]. Розуміння пріоритетних критеріїв для різних категорій користувачів обґрунтовує необхідність створення гнучких алгоритмів пошуку найбільш релевантних пропозицій. Відповідно до цього, інформаційні системи повинні мати змогу трансформувати поведінкові шаблони користувача у набір правил.

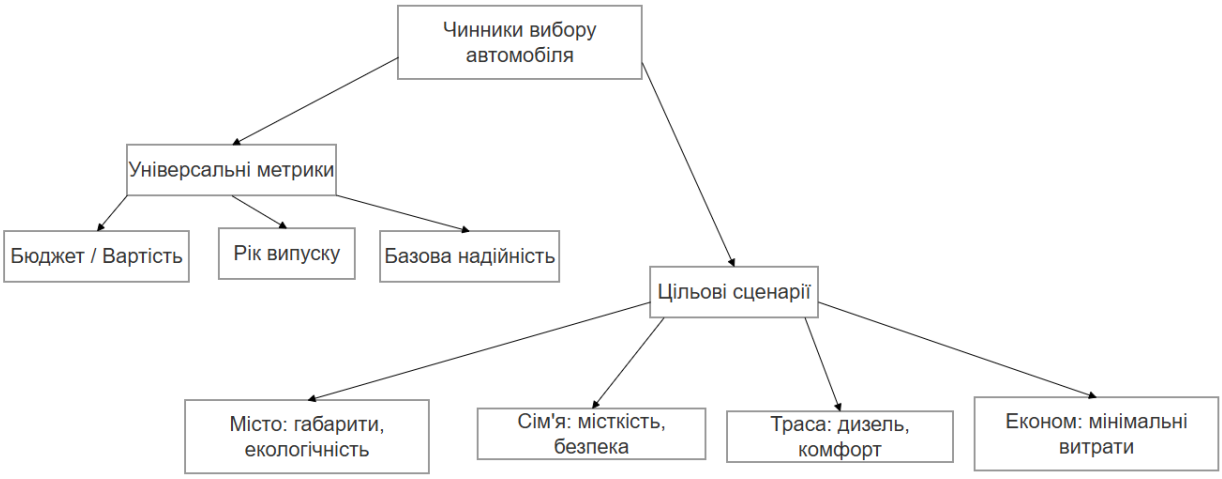

Рисунок 1.1 – Класифікація чинників вибору залежно від сценаріїв використання

Такий підхід дозволяє адаптувати вибірку під індивідуальну модель експлуатації автомобіля. Замість простого відображення списку, концептуальна система повинна зважувати характеристики відповідно до обраної мети.

Успішний пошук транспортного засобу починається з правильного формування цільового запиту користувачем. Часто споживачі стикаються з проблемою неможливості чітко артикулювати свої потреби технічною мовою. Замість конкретних числових параметрів або строгих логічних умов користувачі оперують абстрактними поняттями. Наприклад, покупець може шукати сімейний автомобіль для поїздок на дачу або економну машину для міста. Зазначена обставина створює семантичний розрив між користувацьким уявленням та жорсткою структурою баз даних вебагрегаторів. Розуміння психології взаємодії користувача з системою є критичним для UX [8, 9].

Подолання цього розриву вимагає застосування проміжних інтелектуальних інтерфейсів. Інформаційна система повинна проводити інтерактивне опитування для трансформації життєвих потреб у набір конкретних вагових коефіцієнтів та фільтрів. Концептуальну схему переходу

від абстрактної мети до структурованої вибірки проілюстровано на рисунку 1.2.

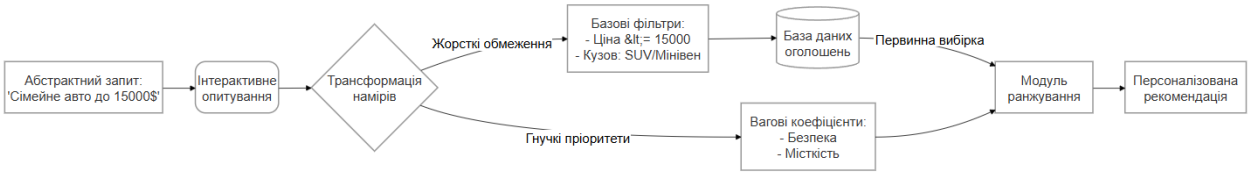

Рисунок 1.2 – Концептуальна схема трансформації цільового запиту

Процес формування запиту часто має ітеративний характер. Перший запит зазвичай є загальним, після чого користувач уточнює параметри на основі отриманих проміжних результатів. У цей час відбувається коригування очікувань, а покупець може зберегти цікаві варіанти до списку обраного для подальшого порівняння. Отже, інтерфейс взаємодії має підтримувати збереження контексту та дозволяти легко змінювати окремі критерії. Наприклад, зміна діапазону років випуску повинна відбуватися без необхідності повторного введення всієї інформації.

Ефективне формування цільових запитів напряму залежить від якості та швидкості зворотного зв’язку. Якщо застосунок пропонує зрозумілі елементи керування та швидку навігацію, користувач оперативніше приходить до розуміння своїх реальних потреб. Тому інтерактивність процесу пошуку стає ключовим елементом сучасних систем підбору.

Сучасні платформи з продажу автомобілів містять сотні тисяч оголошень. Наявність такого масиву даних створює ефект інформаційного перевантаження для кінцевого користувача. Ручне фільтрування вимагає встановлення десятків параметрів у громіздких формах пошуку. Споживачі змушені самостійно визначати межі пробігу, об’єм двигуна та тип коробки передач.

Зазначений підхід часто призводить до помилок відсіювання, коли потенційно підхожі варіанти залишаються поза увагою через занадто жорсткі

обмеження. Сучасні платформи з продажу автомобілів містять сотні тисяч оголошень. Наявність такого масиву даних створює ефект інформаційного перевантаження для кінцевого користувача. Ручне фільтрування вимагає встановлення десятків параметрів у громіздких формах пошуку. Споживачі змушені самостійно визначати межі пробігу, об’єм двигуна та тип коробки передач. Зазначений підхід часто призводить до помилок відсіювання, коли потенційно підхожі варіанти залишаються поза увагою через занадто жорсткі обмеження.

Когнітивне навантаження під час перегляду великої кількості однотипних оголошень швидко виснажує робочу пам’ять людини. Достовірно відомо, що людина здатна одночасно утримувати в увазі лише обмежену кількість об’єктів. Під час порівняння кількох автомобілів на різних вкладках втрачається фокус, а важливі деталі можуть бути проігноровані. Для розуміння переваг інтелектуальних систем здійснено порівняння підходів до фільтрування, результати якого наведено у таблиці 1.1.

Таблиця 1.1 – Порівняння ручного та інтелектуального підходів до фільтрування

Критерій порівнювання	**Традиційне фільтрування на вебресурсах**	**Інтелектуальне ранжування**
1	**2**	**3**
Парадигма пошуку	Жорстке відсікання за параметрами.	Зважене оцінювання кожної альтернативи.
Врахування мети	Відсутнє.	Наявне через розрахунок вагових коефіцієнтів.

Продовження таблиці 1.1

1	2	3
Робота з бюджетом	Рівноцінне показування варіантів до межі вартості.	Пріоритезація оптимальних альтернатив за бюджетом.
Когнітивне навантаження	Надмірне через потребу самостійного аналізування.	Мінімізоване завдяки отриманню ранжованого переліку.

Окрім технічних характеристик, складність полягає в постійній появі нових пропозицій. Користувач змушений повторювати процедуру пошуку щодня. Поява нових оголошень вимагає постійного моніторингу, що перетворює підбір автомобіля на повноцінну рутинну роботу. Усе це обґрунтовує необхідність створення систем, здатних здійснювати автоматизоване збирання даних та їх подальше структурування без безпосередньої участі людини.

Час, витрачений на пошук автомобіля, є одним із найкритичніших ресурсів. Процес включає не лише застосування фільтрів на сайті агрегатора, але й детальний аналіз кожного відібраного оголошення. Загальні часові витрати на ухвалення рішення зростають пропорційно кількості доступних альтернатив, що зумовлює низьку ефективність традиційного пошуку.

Відповідно до психофізичних законів ухвалення рішень, загальний час пошуку складається з двох основних компонентів. Перший компонент включає час на базове та детальне оцінювання кожного розглянутого оголошення, що охоплює перегляд фотографій та аналіз характеристик. Другий компонент становить час на когнітивне оброблення загальної кількості альтернатив, який зростає логарифмічно. Ефективність пошукової сесії безпосередньо залежить від частки дійсно релевантних пропозицій у загальній видачі та є обернено пропорційною загальному витраченому часу.

З цих залежностей випливає, що лінійне збільшення кількості розглянутих пропозицій призводить до стрімкого падіння ефективності процесу підбору. Оптимізація вимагає делегування рутинних операцій програмним засобам. Якщо система здатна самостійно акумулювати актуальні дані та попередньо ранжувати варіанти, користувач витрачатиме час лише на перегляд відфільтрованого списку рекомендацій, що повністю відповідає його цільовому запиту.

1.2 Аналіз вебагрегаторів авто

На сучасному ринку функціонує низка популярних вебагрегаторів, які надають доступ до баз даних уживаних автомобілів. Основний функціонал подібних платформ базується на наданні користувачам структурованих каталогів із можливістю багатокритеріального пошуку [10]. Системи збирають дані від приватних осіб та автосалонів, забезпечуючи стандартизацію подання інформації. Зазначена стандартизація полягає у поділі пропозицій на марки, моделі, роки випуску та типи кузова.

Для побудови незалежних експертних систем ці платформи виступають виключно як джерела сирих даних. Оскільки доступ до таких даних часто буває неструктурованим, виникає потреба у відокремленні процесу збирання інформації від процесу її аналізу. Архітектура концептуальної взаємодії системи підбору з вебагрегатором наведена на рисунку 1.3.

Платформи-агрегатори впроваджують інструменти для підвищення довіри до оголошень. До них належать перевірка історії транспортного засобу та формування зведених звітів. Однак архітектура взаємодії з користувачем на самих платформах залишається класичною, являючи собою сторінки з ієрархічною навігацією та формами введення. Агрегатор виконує пасивну роль, повертаючи результати без глибокого розуміння персонального контексту покупця [11].

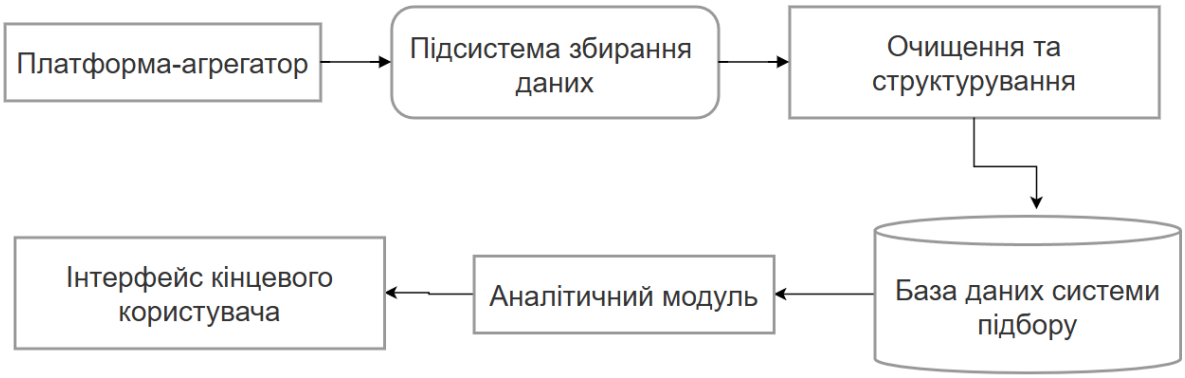

Рисунок 1.3 – Концептуальна архітектура автоматизованого збирання даних з вебагрегатора

Незважаючи на значні обсяги накопичених даних, існуючі агрегатори мають низку суттєвих недоліків у системах фільтрування. Головною проблемою є використання суворої булевої логіки. У класичних системах до фінальної видачі потрапляють лише ті автомобілі, які одночасно і беззаперечно задовольняють абсолютно всі задані користувачем критерії.

Відсутність евристичних алгоритмів оцінювання призводить до того, що користувачу не пропонуються збалансовані альтернативи. Здійснено концептуальне порівняння функціоналу популярних платформ, результати якого наведено у таблиці 1.2.

Таблиця 1.2 – Порівняльний аналіз функціоналу вебагрегаторів автомобілів

Назва платформи	**Гнучкість фільтрів**	**Наявність цільового підбору**	**Збереження контексту сесії**
AUTO.RIA	Висока	Відсутня	Базове
RST	Середня	Відсутня	Відсутнє
OLX Авто	Висока	Базова	Базове

Жодна з проаналізованих платформ не підтримує повноцінного підбору на основі життєвих цілей. Користувач залишається сам на сам із масивом

даних. Крім того, мобільні версії сайтів часто перевантажені рекламними інтеграціями, що погіршує користувацький досвід і стимулює пошук більш лаконічних інтерфейсів.

Ефективність підбору автомобіля значною мірою залежить від алгоритмів ранжування результатів пошуку. Традиційні агрегатори базуються на фільтрації вмісту, де результати найчастіше сортуються за єдиним параметром. Зазвичай таким критерієм виступає ціна від найменшої до найбільшої або дата додавання оголошення. Зазначений підхід не враховує комплексність характеристик об’єкта. Проєктування систем обробки великих даних вимагає особливих архітектурних підходів [12].

Для створення персоналізованого помічника необхідно перейти від простого сортування до багатокритеріального ранжування. В основі такої системи лежить концепція базового рейтингу. Кожен об’єкт оцінюється за сукупністю універсальних характеристик. Автомобілі, що пропонують кращий технічний стан або менший вік за ту саму ціну в межах встановленого бюджету, повинні отримувати вищу позицію у видачі. Це забезпечує пошук найбільш збалансованих варіантів та відсіює пропозиції із завищеною вартістю. Детальний математичний апарат такого ранжування потребує окремого дослідження та побудови системи вагових коефіцієнтів.

Використання евристичного підходу зумовлює перехід від класичного фільтра до експертної системи. Головна відмінність полягає у здатності системи застосовувати різну логіку оцінювання залежно від виявленого контексту або мети користувача. Моделювання такої адаптивної поведінки зображено на рисунку 1.4.

Така експертна система дозволяє відійти від шаблонної видачі. Якщо користувач формулює запит для сімейного використання, система автоматично надає перевагу моделям з відповідним типом кузова. Для мети оптимізації витрат пріоритет зміщується на автомобілі з мінімальним споживанням пального або на ті пропозиції, вартість яких значно нижча за встановлений фінансовий ліміт. Впровадження подібної логіки є

фундаментом для створення сучасного персоналізованого інструменту підбору.

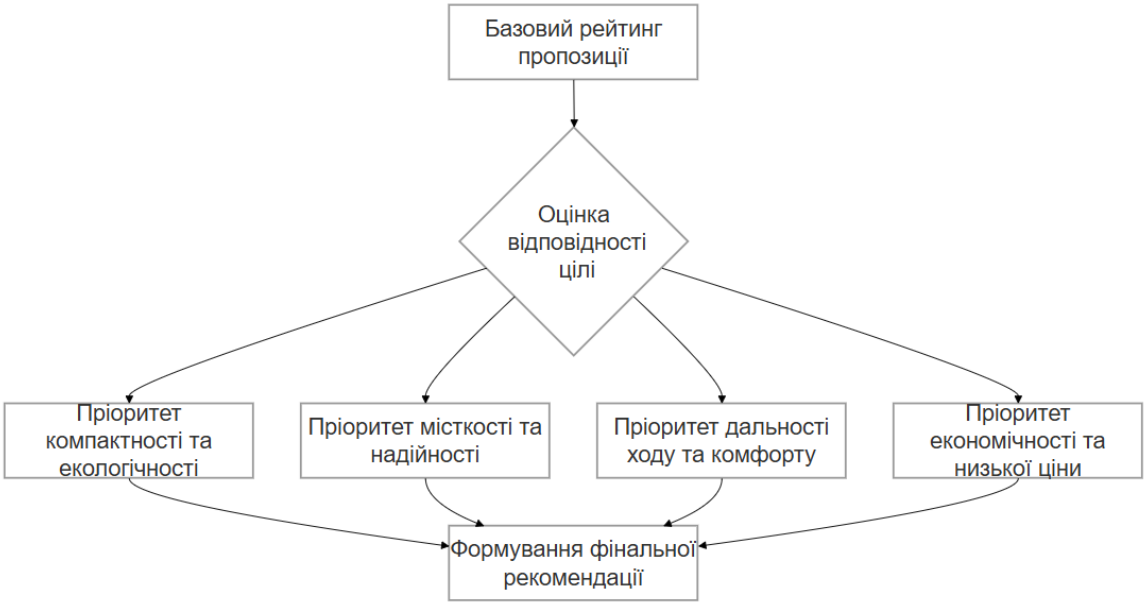

Рисунок 1.4 – Концептуальна логіка адаптації системи ранжування під цільові сценарії

1.3 Месенджери як інтерфейс

Розмовні інтерфейси, реалізовані у вигляді ботів для популярних месенджерів, пропонують принципово новий підхід до взаємодії людини з інформаційними системами. Насамперед розгортання таких рішень не вимагає додаткового завантаження та встановлення програмного забезпечення [13]. Месенджери забезпечують кросплатформність та швидкий доступ до сервісу з будь-якого пристрою за умови наявності облікового запису.

У таблиці 1.3 наведено порівняльний аналіз, що демонструє ключові переваги месенджера у порівнянні з традиційними вебсистемами.

Таблиця 1.3 – Порівняння класичних вебінтерфейсів та розмовних інтерфейсів у месенджерах

Параметр	**Традиційний вебагрегатор**	**Інтерфейс у месенджері**
Навігація	Багаторівневі меню та сторінки	Інтерактивні кнопки та текстові команди
Асинхронність	Потребує постійного оновлення сторінки	Використання фонової роботи
Збереження історії	Обмежене поточною сесією браузера	Постійна історія повідомлень у чаті
Візуалізація	Складні багатоелементні макети	Лаконічні галереї фотографій

Важливою перевагою є можливість використання асинхронної взаємодії. Завдяки підтримці інтерактивних клавіатур месенджери вдало комбінують переваги текстового спілкування зі швидким вибором стандартизованих варіантів. Такий підхід мінімізує кількість помилок під час введення даних і робить взаємодію більш природною. Розмовні інтерфейси потребують специфічного підходу до проєктування діалогів [14, 15].

Перенесення алгоритмів пошуку в середовище месенджера вимагає специфічної архітектурної адаптації. Оскільки екранний простір обмежений форматом чату, система не може виводити десятки параметрів одночасно. Основне завдання полягає в покроковому зборі інформації від користувача. Концептуальну модель такої поетапної взаємодії наведено на рисунку 1.5.

Згідно з наведеною схемою, процес розбито на чіткі та логічно пов’язані етапи. На кожному етапі система очікує специфічний тип даних і валідує їх перед переходом до наступного кроку. Після завершення збору параметрів формується запит до локального сховища інформації, куди заздалегідь завантажено актуальні оголошення.

Така адаптація передбачає концептуальне відокремлення логіки користувацького інтерфейсу від логіки збирання ринкових даних. Підсистеми збирання працюють у фоновому режимі та регулярно оновлюють інформацію. Водночас інтерфейс у месенджері забезпечує миттєву видачу результатів, гарантуючи високу швидкість відгуку. Це відповідає концепції мінімізації когнітивного навантаження [16].

Рисунок 1.5 – Блок-схема процесу поетапного анкетування користувача

1.4 Постановка задачі

Через перенасичення автомобільного ринку традиційні механізми пошуку є неефективними. Наявні сервіси консолідування оголошень застосовують жорстке фільтрування без урахування сценаріїв експлуатування, що призводить до значних часових витрат.

Об’єктом роботи є процес автоматизованого пошуку та персоналізованого підбору транспортних засобів на основі даних автомобільних вебагрегаторів.

Метою роботи є розроблення експертної системи у вигляді месенджер-бота для підбору автомобілів, що використовує методи автоматизованого збирання даних та алгоритми багатокритеріального ранжування для надання рекомендацій залежно від індивідуальних цілей користувача [17].

Для досягнення мети необхідно вирішити такі завдання:

− проаналізувати предметну область, дослідити чинники вибору автомобілів та виявити недоліки існуючих систем фільтрування на наявних вебагрегаторах;

− здійснити порівняльний аналіз технологій автоматизованого збирання даних, систем збереження інформації та архітектурних рішень для створення розмовних інтерфейсів;

− обґрунтувати вибір алгоритму евристичного ранжування та розробити модель підбору на основі універсальних критеріїв і цільових сценаріїв;

− спроєктувати структуру програмного забезпечення, включаючи моделювання прецедентів та кінцевих автоматів;

− розробити та протестувати підсистему автоматизованого збирання ринкових даних з відкритих джерел;

− реалізувати програмний застосунок у вигляді бота, інтегрувати модель рекомендацій та провести комплексне тестування розробленого рішення.

2 АНАЛІЗ МОДЕЛЕЙ ТА ТЕХНОЛОГІЙ СИСТЕМИ ПОШУКУ АВТОМОБІЛЯ

2.1 Методи збору даних

Розроблення інформаційних систем, які оперують даними сторонніх платформ, вимагає ретельного вибору методів отримання інформації. Найбільш стандартизованим підходом є використання офіційних інтерфейсів прикладного програмування (API). Зазначені інтерфейси забезпечують стабільний доступ до структурованих наборів даних у форматах JSON або XML. Використання офіційних каналів гарантує високу швидкість оброблення запитів та знижує ризик блокування з боку сервера. Платформи-агрегатори зазвичай надають детальну документацію для розробників, яка описує всі доступні кінцеві точки та структуру відповідей сервера [18].

Проте використання офіційних інтерфейсів у межах розроблення незалежних експертних систем стикається із низкою суттєвих перешкод. Насамперед платформи встановлюють жорсткі квоти на кількість запитів за одиницю часу, використовуючи алгоритми обмеження трафіку на кшталт «маркерного кошика» (Token Bucket) або «дірявого відра» (Leaky Bucket). Перевищення цих лімітів миттєво призводить до повернення кодів помилок HTTP 429 (Too Many Requests) та тимчасового або перманентного блокування доступу. Для комерційних проєктів або систем із великою кількістю користувачів базових безкоштовних квот виявляється недостатньо. Відповідно до цього виникає необхідність переходу на платні тарифні плани, що значно збільшує загальну вартість розроблення та експлуатації програмного продукту.

Іншим вагомим обмеженням є штучна неповнота даних, які надаються через відкриті канали. Власники платформ часто обмежують доступ до критично важливої інформації з метою захисту своєї комерційної таємниці та запобігання створенню повноцінних клонів сервісу. Наприклад, в офіційному

доступі можуть бути відсутні дані про динаміку зміни ціни, повні текстові описи від продавців або історія перевірок транспортного засобу за реєстрами Міністерства внутрішніх справ. Відсутність таких даних унеможливлює роботу складних евристичних алгоритмів оцінювання, які потребують максимально повної інформаційної картини для формування якісних рекомендацій. Здійснено загальне порівняння офіційних інтерфейсів та методів прямого вилучення даних, результати якого наведено у таблиці 2.1.

Таблиця 2.1 – Порівняння методів отримання даних з вебагрегаторів

Критерій порівнювання	**Офіційні інтерфейси прикладного програмування**	**Пряме вилучення даних з HTML-коду**
Рівень структурування даних	Стандартизований (формати JSON або XML).	Всі дані з сайту, які на ньому є
Повнота отримання відомостей	Обмежена правилами безпеки вебресурсу.	Повне отримання всіх доступних відомостей
Регулювання частоти запитів	Жорстко встановлене обмеженнями постачальника.	Визначене засобами уникнення блокування.

З наведеного аналізу випливає висновок про недостатність офіційних інтерфейсів для реалізації повноцінної експертної системи. Необхідність отримання повного спектра характеристик автомобіля вимагає пошуку альтернативних методів збирання інформації. Застосування нестандартних підходів дозволяє отримати доступ до всіх видимих на вебсторінці елементів. Такий підхід формує міцний фундамент для подальшого математичного моделювання та аналізу.

Альтернативним підходом до збирання інформації є автоматизоване вилучення даних (вебскрапінг) безпосередньо з гіпертекстової розмітки вебсторінок. Сучасні методи автоматизованого збору даних з Python дозволяють ефективно вилучати складні структури [19, 20]. Цей процес передбачає завантаження HTML-коду цільової сторінки та подальший синтаксичний аналіз її структури. Програмний модуль знаходить потрібні вузли в об’єктній моделі документа (DOM) та витягує з них текстову або числову інформацію. Завдяки цьому розробник отримує доступ до абсолютно всіх даних, які платформа відображає для звичайного користувача у браузері.

Для наочного обґрунтування цього вибору було проаналізовано доступність конкретних атрибутів автомобільного оголошення через різні канали постачання інформації. Результати цього специфічного аналізу зведено у таблиці 2.2.

Таблиця 2.2 – Доступність атрибутів оголошення залежно від методу збирання

Атрибут оголошення	**Доступність через безкоштовний API**	**Доступність через вебскрапінг**	**Вплив на алгоритм підбору**
Базова ціна та рік	Доступно	Доступно	Критичний (фільтрування)
Текстовий опис продавця	Обмежено (лише фрагмент)	Доступно у повному обсязі	Високий (аналіз маркерів)
Історія зміни ціни	Недоступно	Доступно	Середній (оцінка ліквідності)
Контактні дані продавця	Захищено (потребує авторизації)	Доступно через симуляцію кліку	Низький (на етапі ранжування)

Доцільність застосування автоматизованого вилучення обґрунтовується високою гнучкістю та відсутністю штучних обмежень на обсяги інформації, як це видно з таблиці 2.2. Модуль збирання може функціонувати у фоновому режимі та регулярно оновлювати локальну базу даних, зберігаючи історію змін оголошень. Наявність історичних даних відкриває можливості для аналізу цінових трендів та визначення швидкості продажу певних моделей автомобілів. Концептуальну схему процесу автоматизованого вилучення даних проілюстровано на рисунку 2.1.

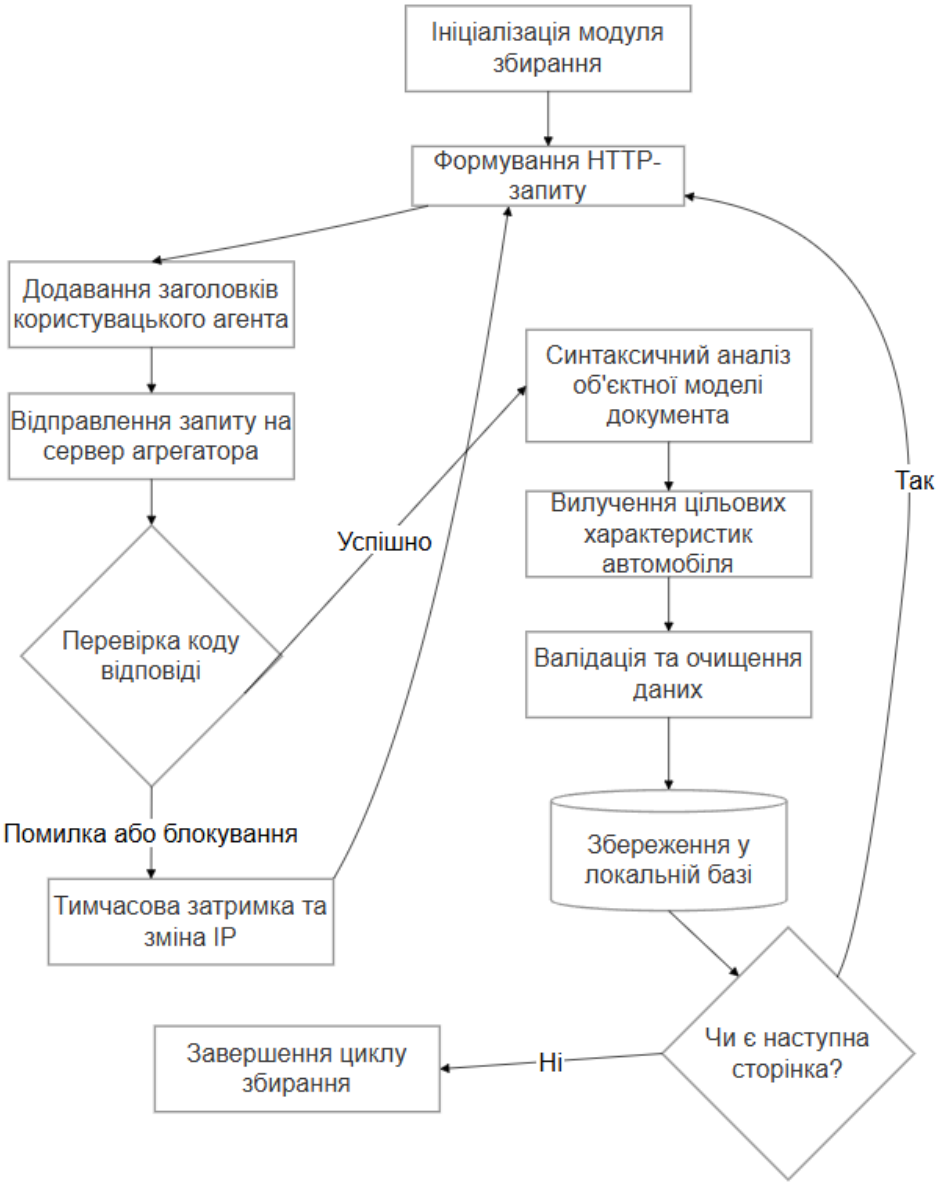

Рисунок 2.1 – Блок-схема алгоритму автоматизованого вилучення даних

Важливим аспектом використання цього методу є забезпечення стійкості до змін у дизайні цільового вебсайту. Власники платформ періодично оновлюють структуру розмітки, що може призвести до збоїв у роботі аналізатора. Відповідно до цього архітектура модуля збирання повинна підтримувати швидке оновлення селекторів. Використання регулярних виразів та семантичних маркерів дозволяє частково нівелювати цю проблему. Розроблення гнучкої системи збору даних вимагає застосування спеціалізованих інструментальних засобів.

Водночас необхідно враховувати етичні та технічні норми навантаження на сервери донорів інформації. Алгоритм збирання повинен дотримуватися директив файлу robots.txt та містити штучні затримки між запитами. Важливим аспектом є дотримання правових та етичних норм збирання даних [21]. Це дозволяє імітувати поведінку реального користувача та уникати створення надмірного навантаження на інфраструктуру платформи-агрегатора. Дотримання цих правил гарантує стабільну роботу системи протягом тривалого часу без ризику перманентного блокування за IP-адресою та мінімізує юридичні ризики.

Розроблення підсистеми збирання даних вимагає порівняння різних технологій отримання та оброблення гіпертексту. Першим підходом є використання повноцінних браузерних рушіїв. Ці інструменти дозволяють виконувати скрипти на сторінці та імітувати реальні натискання кнопок або прокручування екрана. Такий підхід є ефективним для роботи з динамічними вебдодатками. У таких додатках контент завантажується асинхронно за допомогою технології AJAX вже після початкового відображення каркаса сторінки.

Однак браузерні рушії споживають значну кількість оперативної пам’яті та процесорного часу. Запуск кожної копії браузера створює високе навантаження на сервер розробника. Додатковим ускладненням є необхідність оброблення CSRF-токенів та прихованих полів форм, що генеруються динамічно для захисту від ботів. Для збирання десятків тисяч

оголошень цей підхід виявляється економічно та технічно неефективним. Альтернативним варіантом є використання бібліотек для виконання базових HTTP-запитів із подальшим синтаксичним аналізом отриманого текстового рядка.

Оскільки більшість агрегаторів автомобілів віддають основну інформацію безпосередньо у початковому HTML-коді для покращення пошукової оптимізації, використання важких браузерних рушіїв є недоцільним. Комбінація бібліотек для HTTP-запитів та статика-аналізаторів забезпечує оптимальний баланс між швидкістю розроблення та продуктивністю системи. Модуль збирання працює швидко, вимагає мінімальних обчислювальних ресурсів і може бути розгорнутий на базових хмарних серверах [22].

Додатковою перевагою статичних аналізаторів є їхня висока стійкість. Вони дозволяють шукати елементи за каскадними таблицями стилів або ідентифікаторами. Такий підхід робить процес вилучення даних декларативним та легко читабельним. У результаті код підсистеми збирання стає легким для подальшого супроводу та швидкого масштабування алгоритмів на нові вебагрегатори у разі розширення проєкту.

Реалізація клієнтського інтерфейсу у вигляді бота для месенджера вимагає вибору відповідного програмного каркаса. Існує два основні архітектурні підходи до оброблення мережевих запитів. До них належать синхронний та асинхронний підходи. Синхронні бібліотеки обробляють кожне повідомлення від користувача послідовно у єдиному потоці виконання. Під час очікування відповіді від бази даних або зовнішнього сервера весь процес блокується. Це призводить до значних затримок у разі одночасного звернення великої кількості користувачів.

Асинхронний підхід базується на використанні циклу подій. У такій архітектурі потік виконання не блокується під час операцій введення-виведення. Замість очікування система передає управління іншим завданням. Після отримання відповіді від зовнішнього ресурсу виконання перерваної

функції відновлюється. Загальний час очікування користувача у синхронній та асинхронній системах можна описати відповідними математичними залежностями.

Для синхронної системи час оброблення N одночасних запитів обчислюється як сума часу оброблення кожного окремого запиту. Відповідну залежність наведено у формулі нижче:

$$T_{sync} = \sum_{i=1}^{N} \left(t_{cpu,i} + t_{io,i} \right), \tag{2.1}$$

де T_{sync} – загальний час оброблення всіх запитів;

$t_{cpu,i}$ – час роботи процесора i-го для запиту;

$t_{io,i}$ – час очікування операцій введення-виведення для i-го для запиту.

Для асинхронної моделі загальний час визначається інакше. Залежність для асинхронної системи подано у наступній формулі:

$$T_{async} \approx \sum_{i=1}^{N} t_{cpu,i} + \max_{1 \le i \le N} \left(t_{io,i} \right). \tag{2.2}$$

Крім внутрішньої архітектури оброблення запитів, важливим є вибір моделі взаємодії із серверами Telegram. Існують моделі Long Polling та Webhooks. Модель Long Polling передбачає постійні звернення бота до сервера з питанням про наявність нових повідомлень, що є прийнятним для етапу розроблення. Натомість архітектура Webhooks дозволяє серверу Telegram самостійно надсилати нові події на визначену кінцеву точку розроблюваної системи. Аналіз цих залежностей доводить беззаперечну перевагу асинхронної архітектури у поєднанні з моделлю Webhooks для систем із великим мережевим навантаженням. Розглянуто найбільш

популярні бібліотеки мови програмування, результати порівняння яких наведено у таблиці 2.3.

Таблиця 2.3 – Порівняльний аналіз фреймворків для створення Telegram-ботів

Назва фреймворка	**Архітектура**	**Механізм збереження станів**	**Можливість опрацювання запитів**
pyTelegramBotAPI	Синхронна	Відсутній вбудований	Обмежена одним потоком
python-telegram-bot	Змішана	Реалізований через класи	Паралельне багатопотокове виконання
aiogram	Асинхронна	Інтегрована машина станів	Асинхронне масштабоване виконання

З огляду на необхідність одночасного обслуговування багатьох користувачів та підтримки складних сценаріїв діалогу вибір зупиняється на асинхронних рішеннях. Інтегрована підтримка машин станів дозволяє зручно реалізувати покрокове опитування користувача. Це критично важливо для процесу збирання цільових параметрів підбору автомобіля. Використання сучасних асинхронних бібліотек забезпечує високу масштабованість розроблюваної системи.

Вибір системи управління базами даних є ключовим етапом архітектурного проєктування. Система повинна зберігати профілі користувачів, їхні збережені автомобілі та загальний масив зібраних ринкових пропозицій. Існує концептуальний поділ на нереляційні та реляційні бази даних. Нереляційні системи добре підходять для збереження

неструктурованих документів із мінливим набором полів. Проте характеристики автомобілів мають чітко визначену структуру з фіксованими типами даних.

Реляційні бази даних забезпечують цілісність інформації за допомогою зовнішніх ключів. Вони підтримують транзакційність та дозволяють виконувати складні аналітичні запити за допомогою структурованої мови запитів (SQL). Для забезпечення високої швидкості доступу реляційні бази даних використовують індекси на основі B-дерев. Правильне налаштування індексів для колонок «ціна» та «рік випуску» дозволяє системі виконувати пошук за логарифмічний час замість лінійного сканування всієї таблиці. Порівняння основних представників реляційних систем наведено у таблиці 2.4.

Для розроблюваної системи підбору автомобілів вбудована файлова база даних є найбільш оптимальним вибором. Вона не вимагає встановлення окремого сервера та налаштування прав доступу. Уся база зберігається у єдиному файлі, що суттєво спрощує процес резервного копіювання та перенесення системи на інше обладнання.

Таблиця 2.4 – Порівняння систем управління реляційними базами даних

Критерій порівнювання	**PostgreSQL**	**MySQL**	**SQLite**
1	**2**	**3**	**4**
Архітектурна модель	Клієнт-серверна.	Клієнт-серверна.	Вбудована у файл.
Регулювання параметрів	Детальне конфігурування пам'яті.	Базове налаштування безпеки.	Налаштування у коді застосунку.

Продовження таблиці 2.4

1	2	3	4
Паралельне записування	Паралельне без блокування (MVCC).	Блокування рядків або таблиць.	Послідовне з блокуванням файлу.
Особливості розгортання	Встановлення системної служби.	Встановлення сервера та прав.	Імпортування бібліотеки в проєкт.

Використання механізму Write-Ahead Logging (WAL) у таких системах дозволяє частково нівелювати проблеми з паралельним записом, забезпечуючи прийнятний рівень продуктивності.

Крім того, більшість операцій у системі є операціями читання. Бот переважно шукає та фільтрує оголошення за запитом користувача. Операції запису відбуваються значно рідше під час оновлення масиву даних або збереження автомобіля в обране. Тому обрана архітектура повністю покриває потреби проєкту та мінімізує витрати апаратних ресурсів на етапі розгортання програмного продукту.

2.2 Методи багатокритеріального ранжування

Проблема вибору найкращого автомобіля з великої множини альтернатив належить до класу задач багатокритеріальної оптимізації. Кожен об’єкт оцінюється за набором різнорідних параметрів. Класичні системи пошуку використовують жорстке відсіювання за булевою логікою. Такий підхід не дозволяє ранжувати залишковий набір даних за ступенем їхньої якості. Для вирішення цієї проблеми застосовуються евристичні алгоритми

оцінювання. Ці алгоритми імітують процес ухвалення рішення людиною-експертом.

В основі евристичного оцінювання лежить метод зваженої суми критеріїв. Кожному параметру транспортного засобу призначається певний бал [23]. Цей бал коригується ваговим коефіцієнтом, який відображає важливість параметра для кінцевого користувача. Загальна математична модель лінійної зваженої комбінації представлена у вигляді формули:

$$S(x) = \sum_{j=1}^{M} w_j \cdot f_j(x), \tag{2.3}$$

де $S(x)$ – фінальний рейтинг альтернативи x;

M – відповідає загальній кількості критеріїв оцінювання;

w_j – позначає ваговий коефіцієнт j-го критерію;

$f_j(x)$ – оцінка транспортного засобу за цим специфічним критерієм.

Усі вагові коефіцієнти повинні бути нормалізованими. Це означає, що сума всіх вагових коефіцієнтів дорівнює одиниці:

$$\sum_{j=1}^{M} w_j = 1. \tag{2.4}$$

Використання методу зваженої суми було обрано з огляду на його обчислювальну ефективність порівняно зі складнішими підходами, такими як метод аналізу ієрархій або алгоритми пошуку оптимальності за Парето. У системах реального часу, де рейтинг десятків тисяч автомобілів необхідно перераховувати динамічно під час кожного запиту користувача, обчислювальна складність має вирішальне значення. Лінійна модель дозволяє виконувати розрахунки на рівні реляційної бази даних, значно зменшуючи навантаження на обчислювальний модуль системи.

Такий підхід забезпечує прозорість процесу ранжування. Розробник має змогу легко налаштовувати пріоритети шляхом зміни вагових коефіцієнтів. Завдяки цьому система легко адаптується під різні сценарії використання без необхідності переписування основного програмного коду. Якщо користувачу важливіша ціна, коефіцієнт для вартості збільшується. Якщо важливіший рік випуску, відповідний ваговий коефіцієнт отримує більшу частку у загальній оцінці.

Застосування методу зваженої суми вимагає попереднього приведення всіх параметрів до єдиного масштабу. Характеристики автомобіля мають різні одиниці вимірювання та діапазони значень. Наприклад, ціна вимірюється у тисячах доларів, рік випуску становить чотиризначне число, а об’єм двигуна вимірюється у літрах. Пряме додавання цих значень призведе до того, що параметри з великими числовими значеннями повністю поглинуть вплив менших параметрів. Тому процедура нормалізації є обов’язковим етапом підготовки даних.

Найбільш поширеним методом є лінійна нормалізація Мінімум-Максимум [24, 25]. Цей алгоритм перетворює всі значення параметра у діапазон від нуля до одиниці. Для позитивних параметрів застосовується пряма формула нормалізації. Позитивними вважаються характеристики, збільшення яких покращує оцінку об’єкта. Відповідний математичний вираз має такий вигляд:

$$f_{norm}^{+}(x) = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \tag{2.5}$$

де $x_{\min}$ – мінімальні значення цього параметру у всій вибірці;

$x_{\max}$ – максимальні значення цього параметру у всій вибірці.

Проте деякі характеристики автомобіля мають негативний характер. Наприклад, чим більший пробіг або вища ціна, тим гіршим вважається

варіант за інших рівних умов. Для таких параметрів застосовується обернена нормалізація. Формулу оберненої нормалізації представлено нижче:

$$f_{norm}^{-}(x) = \frac{x_{\max} - x}{x_{\max} - x_{\min}}. \tag{2.6}$$

Альтернативним підходом до нормалізації є застосування Z-оцінки (стандартизації). Цей метод є більш стійким до наявності аномальних викидів у вибірці. Формула стандартизації має такий вигляд:

$$Z(x) = \frac{x - \mu}{\sigma}, \tag{2.7}$$

де μ – середнє арифметичне значення параметра у вибірці;

σ – стандартне відхилення.

Хоча стандартизація ефективно бореться з аномаліями, вона не обмежує значення строгим діапазоном від нуля до одиниці. У контексті експертної системи підбору автомобілів класична нормалізація Мінімум-Максимум у поєднанні з попереднім очищенням бази даних від екстремальних значень демонструє кращу передбачуваність при подальшому множенні на вагові коефіцієнти.

Застосування нормалізації гарантує об’єктивність алгоритму ранжування. Усі параметри отримують рівний вплив на формування підсумкової оцінки перед застосуванням вагових коефіцієнтів. Ця математична операція дозволяє системі коректно порівнювати автомобіль преміумкласу з бюджетною моделлю, обчислюючи відносну вигоду кожної пропозиції в межах заданого користувачем фінансового ліміту.

Розроблювана експертна система використовує дворівневу математичну модель оцінювання автомобілів. Перший рівень відповідає за обчислення універсального базового рейтингу. Цей рейтинг відображає

об’єктивну привабливість пропозиції на ринку незалежно від специфічних цілей конкретного покупця. До базових параметрів належать ціна, рік випуску та загальний показник надійності моделі. Базовий рейтинг розраховується для кожної пропозиції, що пройшла первинне відсіювання за жорсткими фільтрами.

Оцінка цінової привабливості базується на близькості вартості автомобіля до максимального бюджету користувача. Чим ближче ціна до ліміту без його перевищення, тим якіснішим вважається транспортний засіб у межах фінансових можливостей покупця. Рік випуску оцінюється шляхом нарахування бонусних балів за кожен рік, що перевищує встановлений базовий поріг. Загальна формула розрахунку базового рейтингу має вид:

$$S_{base}(x) = w_p \cdot P_{norm}(x) + w_y \cdot \max\left(0, Y(x) - Y_{base}\right) + R(x), \tag{2.8}$$

де $S_{base}(x)$ – загальний базовий рейтинг;

$P_{norm}(x)$ – нормалізована оцінка ціни;

$Y(x)$ – рік випуску транспортного засобу;

Y_{base} – базовий рік;

$R(x)$ – експертна оцінка надійності конкретної моделі;

w_y та w_p – відповідні вагові коефіцієнти.

Базовий рейтинг також може враховувати показник ринкової ліквідності моделі. Транспортні засоби, які в середньому швидше знаходять нових власників, отримують незначну позитивну корекцію. Це допомагає пропонувати користувачам не лише надійні, але й економічно вигідні варіанти з погляду подальшого перепродажу. Такий підхід забезпечує формування збалансованого списку найкращих ринкових пропозицій, відсіюючи застарілі моделі та транспортні засоби із сумнівною репутацією.

У результаті користувач отримує базовий шорт-лист, який згодом коригується на другому рівні алгоритму.

Другий рівень математичної моделі адаптує базовий рейтинг під конкретні сценарії використання транспортного засобу. Теоретичні основи таких систем базуються на нечіткій логіці та інтервальних оцінках [26, 27, 28]. Кожен сценарій має власний набір пріоритетів та штрафних санкцій. Система передбачає чотири основні цільові моделі експлуатації. До них належать поїздки містом, сімейне використання, регулярні поїздки трасою та максимальна економія бюджету. Нарахування балів відбувається шляхом додавання цільового бонусу до базового рейтингу. Загальний вираз має такий вигляд:

$$S_{total}(x) = S_{base}(x) + B_{goal}(x), \tag{2.9}$$

де $S_{total}(x)$ – фінальний рейтинг пропозиції;

$B_{goal}(x)$ – функція цільового бонусу.

Для сценарію використання у місті алгоритм надає перевагу компактним габаритам та екологічності. Формула для розрахунку бонусу міського сценарію наведена нижче:

$$B_{city}(x) = w_{eco} \cdot E_{fuel}(x) + C_{ev}(x) + C_{compact}(x), \tag{2.10}$$

де $E_{fuel}(x)$ – рівень витрати пального;

$C_{ev}(x)$ – константа, що додається у разі наявності електричної силової установки;

$C_{compact}(x)$ – бали за компактні типи кузова.

Для сімейного сценарію алгоритм змінює пріоритети на місткість та підвищену безпеку. Розрахунок сімейного бонусу описується наступним математичним виразом:

$$B_{family}(x) = C_{suv}(x) + w_{safe} \cdot R(x), \tag{2.11}$$

де $C_{suv}(x)$ – бали за кросовери та мініавтобуси;

w_{safe} – коефіцієнт, який створює подвійний акцент на показнику надійності транспортного засобу.

Моделювання сценаріїв також передбачає використання штрафних санкцій (негативних бонусів) у разі конфлікту характеристик автомобіля з обраною метою. Наприклад, великогабаритні позашляховики з високою витратою пального отримують суттєве зниження рейтингу в категорії «Економ». Використання таких цільових функцій дозволяє системі гнучко змінювати порядок пропозицій у фінальній видачі, гарантуючи високий рівень персоналізації рекомендацій. Математичне моделювання прийняття рішень у складних системах описано в роботах [29, 30].

2.3 Об’єктно-орієнтоване моделювання

Проєктування інформаційної системи вимагає чіткого розуміння варіантів використання програмного продукту кінцевими користувачами. Для формалізації функціональних вимог застосовуються діаграми прецедентів згідно зі стандартом уніфікованої мови моделювання. У розроблюваній системі виділяється один головний актор. Ним є користувач месенджера. Він взаємодіє з системою через графічний інтерфейс чату.

Користувач має змогу виконувати базові дії щодо пошуку та збереження інформації. Діаграма наочно демонструє межі системи та зв’язки

між різними функціональними блоками. Допоміжним невидимим актором виступає системний таймер. Цей актор відповідає за ініціалізацію фонових процесів оновлення бази даних оголошень. Концептуальну модель прецедентів проілюстровано на рисунку 2.2 [31].

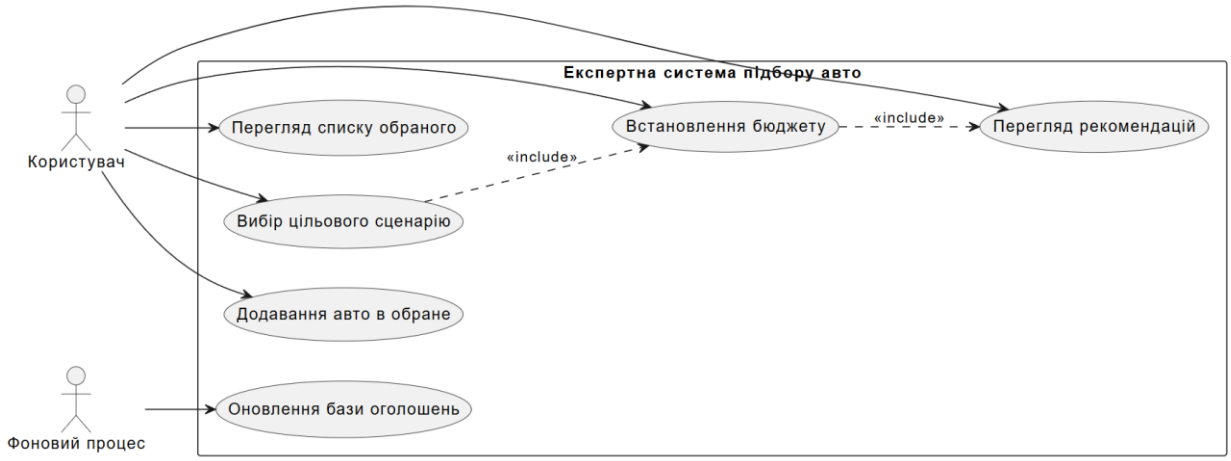

Рисунок 2.2 – UML-діаграма прецедентів експертної системи

В межах об’єктно-орієнтованого підходу актори можуть бути узагальнені. Хоча поточна версія системи передбачає єдиний рівень доступу для всіх користувачів, архітектура прецедентів враховує можливість додавання ролі адміністратора у майбутньому. Адміністратор матиме власні прецеденти взаємодії, такі як примусовий запуск модулів збирання інформації або редагування бази знань системи. Аналіз діаграми показує, що процес підбору автомобіля є послідовним ланцюгом дій. Користувач не може отримати рекомендації без попереднього вибору цільового сценарію та встановлення фінансових обмежень.

Відношення «включає» підтверджують обов’язковість цих кроків. Водночас робота зі списком «обране» є незалежним сценарієм використання. Такий розподіл функціоналу забезпечує логічну ізоляцію модулів на етапі безпосереднього написання програмного коду. Фоновий процес збирання даних повністю прихований від кінцевого споживача послуги.

Реалізація покрокового опитування у середовищі месенджера вимагає застосування архітектурного шаблону скінченного автомата. Цей підхід дозволяє системі запам’ятовувати поточний контекст діалогу з конкретним користувачем. Кожен крок анкети відповідає певному стану системи. Перехід між станами здійснюється лише після отримання перевірених даних від користувача. Якщо отримані дані не відповідають вимогам, система залишається у поточному стані та просить повторити введення інформації.

Використання машин станів гарантує цілісність зібраних даних перед їх передачею до аналітичного модуля. Застосування шаблонів, таких як FSM або Singleton, підвищує надійність коду [32, 33, 34]. Особливу увагу необхідно приділяти збереженню стану діалогу. У разі перезавантаження сервера або втрати з’єднання контекст користувача не повинен анулюватися. Для цього інформація про поточний стан автомата періодично записується у проміжне сховище. Процес оброблення непередбачуваних ситуацій під час переходів між станами проілюстровано на рисунку 2.3.

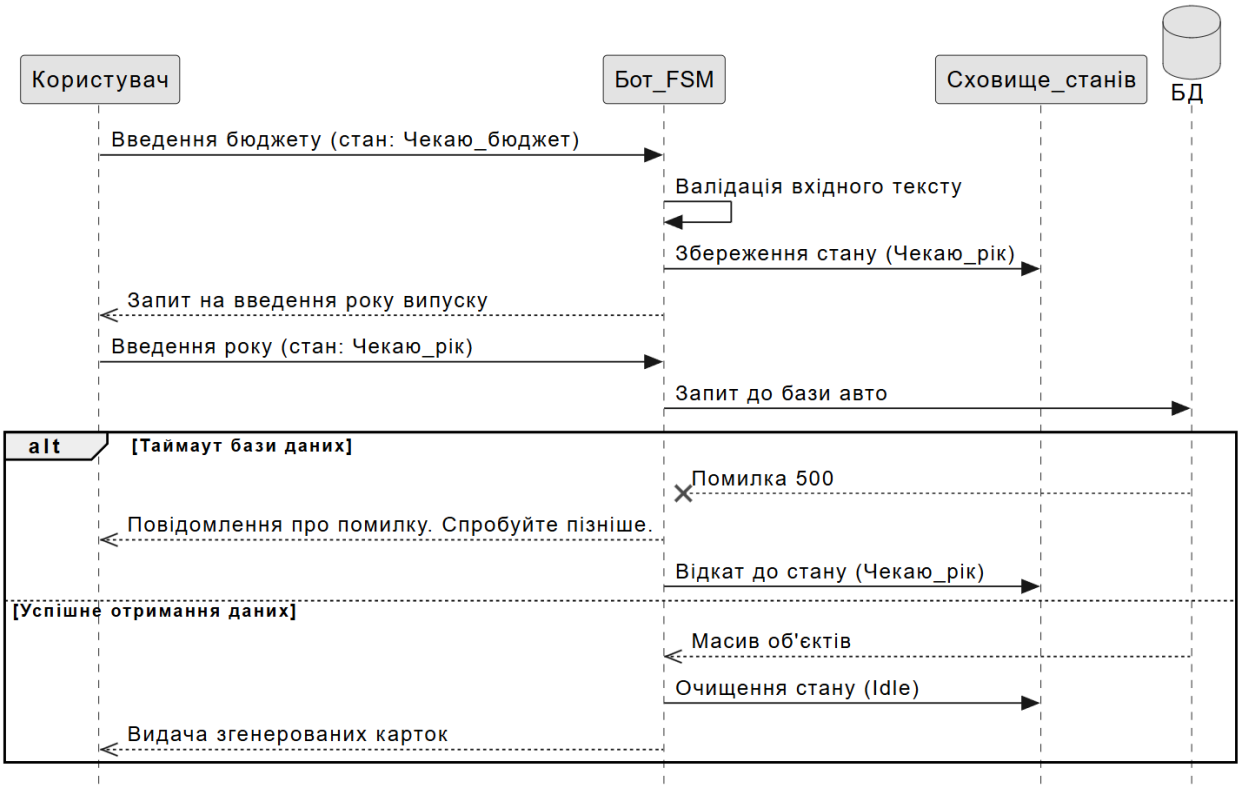

Рисунок 2.3 – Діаграма послідовності оброблення таймаутів машиною станів

Наведена діаграма демонструє важливість механізмів відкату (ролбеку) у разі виникнення помилок на серверному рівні. Механізм валідації та збереження станів відіграє роль захисного бар'єра. Завдяки йому ядро експертної системи завжди отримує дані у стандартизованому та передбачуваному форматі. Після успішного завершення процесу видачі результатів автомат очищує пам'ять та повертається до початкового стану очікування. Це дозволяє користувачу негайно розпочати новий пошуковий запит з іншими параметрами без необхідності перезавантаження інтерфейсу.

Ефективне зберігання та оброблення інформації вимагає проєктування нормалізованої реляційної структури даних. Архітектура бази даних повинна мінімізувати дублювання інформації та забезпечувати швидкий пошук за ключовими критеріями. Відповідно до визначених функціональних вимог розроблено концептуальну модель даних. Модель містить таблиці для зберігання профілів користувачів, характеристик автомобілів, медіафайлів та зв'язків між ними.

Головною сутністю є таблиця автомобілів. Вона акумулює всі технічні показники, необхідні для роботи алгоритмів ранжування. Таблиця обраного виконує роль зв'язкової сутності між користувачем та транспортним засобом, реалізуючи відношення типу багато-до-багатьох. Логічну структуру реляційної бази даних наведено на рисунку 2.4.

Для забезпечення високої продуктивності системи під час виконання SQL-запитів передбачено створення композитних індексів на ключові стовпці. Зокрема, індексація полів марки, моделі, року випуску та ціни є обов'язковою вимогою для швидкого формування первинної вибірки перед початком евристичного ранжування. Спроєктована структура повністю відповідає вимогам третьої нормальної форми та виключає транзитивні залежності [35]. Використання зовнішніх ключів забезпечує каскадне видалення пов'язаної інформації. Наприклад, якщо оголошення стає неактуальним і видаляється з основної таблиці, всі посилання на нього у списках обраного користувачів також автоматично знищуються. Фотографії

транспортних засобів винесено у відокремлену таблицю. Це рішення зумовлено необхідністю зберігати масиви зображень змінної довжини для коректного формування медіа-груп у месенджері. Створена модель даних формує надійний фундамент для наступного етапу програмної реалізації проєкту.

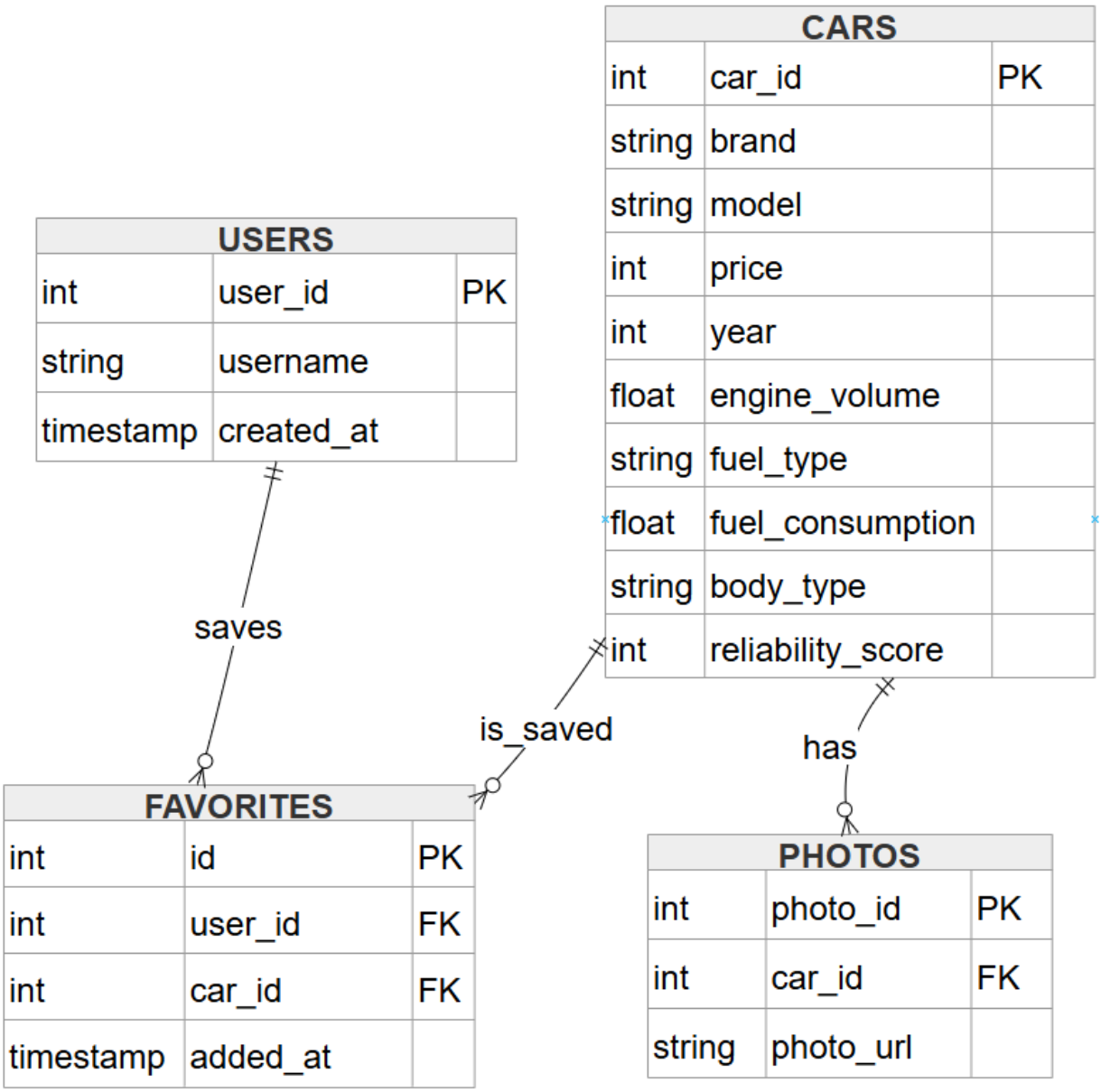

Рисунок 2.4 – ER-діаграма структури бази даних системи

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ СИСТЕМИ ПОШУКУ АВТОМОБІЛЯ

3.1 Розроблення підсистеми збору даних

Процес створення сучасної інтелектуальної системи підбору транспортних засобів на вторинному автомобільному ринку потребує проєктування та розроблення надійного інструментарію для первинного збору, оброблення та структурування інформації. Розроблена підсистема автоматизованого вилучення відомостей виступає базовим фундаментом усього програмного комплексу, оскільки вона забезпечує постійне та систематичне наповнення бази даних актуальною інформацією про пропозиції автомобілів. Специфіка сучасних вебресурсів оголошень, зокрема висока частота оновлення інформації та наявність засобів обмеження автоматизованих запитів, вимагає від архітектури збирання даних особливої гнучкості, стабільності та стійкості до мережевих збоїв.

Основна техніка автоматизованого вилучення технічних та комерційних характеристик автомобілів з вебсторінок полягає у послідовному аналізі структури дерева документа з подальшим приведенням цих даних до єдиного реляційного формату. Процес обходу та зчитування вебресурсів має відбуватися без перевантаження апаратних ресурсів сервера, що вимагає ретельного планування операцій запису під час збереження інформації. У межах цього підрозділу детально висвітлено проєктування архітектури бази даних, обґрунтування вибору технологічного комплексу для програмування модуля зчитування та детальну реалізацію компонентів обходу вебсторінок сайту AUTO.RIA.

Для визначення оптимального технологічного стеку було проведено порівняльний аналіз інструментальних засобів розроблення. Результати

порівняння альтернативних рішень за ключовими критеріями наведено в таблиці 3.1.

Таблиця 3.1 – Порівняльний аналіз інструментальних засобів розроблення

Критерій порівнювання	**Python + BeautifulSoup4**	**Node.js + Cheerio**	**Java + Jsoup**
Середовище виконання	Інтерпретатор CPython.	Рушій V8 (платформа Node.js).	Віртуальна машина Java (JVM).
Спосіб побудови дерева DOM	Використання зовнішніх модулів розбору (lxml).	Використання інтегрованого модуля htmlparser2.	Використання власного вбудованого модуля розбору.
Середній час аналізування сторінки	120 мс (інтерпретування).	95 мс (компілювання «на льоту»).	150 мс (виконання байт-коду).
Система типізації	Динамічне визначення типів під час виконання.	Динамічне визначення типів під час виконання.	Статичне перевіряння типів під час компілювання.

На основі проведеного аналізу для реалізації підсистеми було обрано мову програмування Python версії 3.11 [36, 37]. Як локальне сховище даних задіяно реляційну СКБД SQLite, яка зберігає всю інформацію в одному локальному файлі cars.db та не потребує адміністрування окремого сервера. Таке рішення є найбільш раціональним для масштабу роботи віртуального приватного сервера, оскільки воно забезпечує мінімальний час відгуку за

рахунок відсутності міжпроцесної мережевої взаємодії [38]. Опис проєктованої структури бази даних наведено в таблиці 3.2.

Таблиця 3.2 – Опис фізичної структури бази даних cars.db

Назва таблиці	**Стовпець**	**Тип даних**	**Ключі та обмеження**	**Опис призначення поля**
cars	id	INTEGER	PRIMARY KEY AUTOINCREMENT	Унікальний ідентифікатор автомобіля
	brand	TEXT	NOT NULL	Марка транспортного засобу
	model	TEXT	NOT NULL	Конкретна модель транспортного засобу
	price	INTEGER	NOT NULL	Вартість автомобіля у доларах США
	year	INTEGER	NOT NULL	Календарний рік випуску машини
	link	TEXT	UNIQUE	Посилання на оригінальне оголошення
favorites	user_id	INTEGER	NOT NULL	Ідентифікатор користувача в Telegram

Продовження таблиці 3.2

1	2	3	4	5
	car_id	INTEGER	FOREIGN KEY REFERENCES cars(id)	Зовнішній ключ для зв’язку з машиною

Зв’язок між таблицями організовано з підтримкою каскадного видалення записів, що дозволяє автоматично видаляти збережені оголошення користувачів у разі очищення або оновлення основної бази даних автомобілів. Для відображення структури даних та логічних зв’язків побудовано діаграму класів у нотації UML, яку наведено на рисунку 3.1.

Для забезпечення цілісності структури розроблено модуль ініціалізації сховища. Використання реляційних моделей забезпечує цілісність даних [39]. Унікальним рішенням є створення окремої таблиці додаткових фотографій лотів, що дозволяє зберігати необмежену кількість зображень для кожного автомобіля без дублювання записів в основній таблиці. Відповідний фрагмент створення таблиць та їхніх зовнішніх зв’язків наведено у лістингу 3.1.

Для первинної перевірки працездатності оціночної моделі в базі даних розроблено механізм наповнення початковими тестовими даними. Методи класифікації об’єктів за їх структурними описами є ключовими для аналізу оголошень [40, 41, 42]. Цей блок коду виконується автоматично під час першого запуску застосунку, якщо виявлено порожнє сховище. Логіку автоматичного заповнення представлено у лістингу 3.2.

Збір інформації здійснюється шляхом надсилання HTTP-запитів до вебресурсу AUTO.RIA. Для запобігання блокуванню запитів з боку серверів захисту у підсистемі реалізовано динамічну підміну заголовків запиту HTTP.

З метою забезпечення стабільного завантаження сторінок за нестабільного мережевого з’єднання було розроблено унікальний механізм

експоненційної затримки. Програмну реалізацію цього компонента наведено у лістингу 3.3.

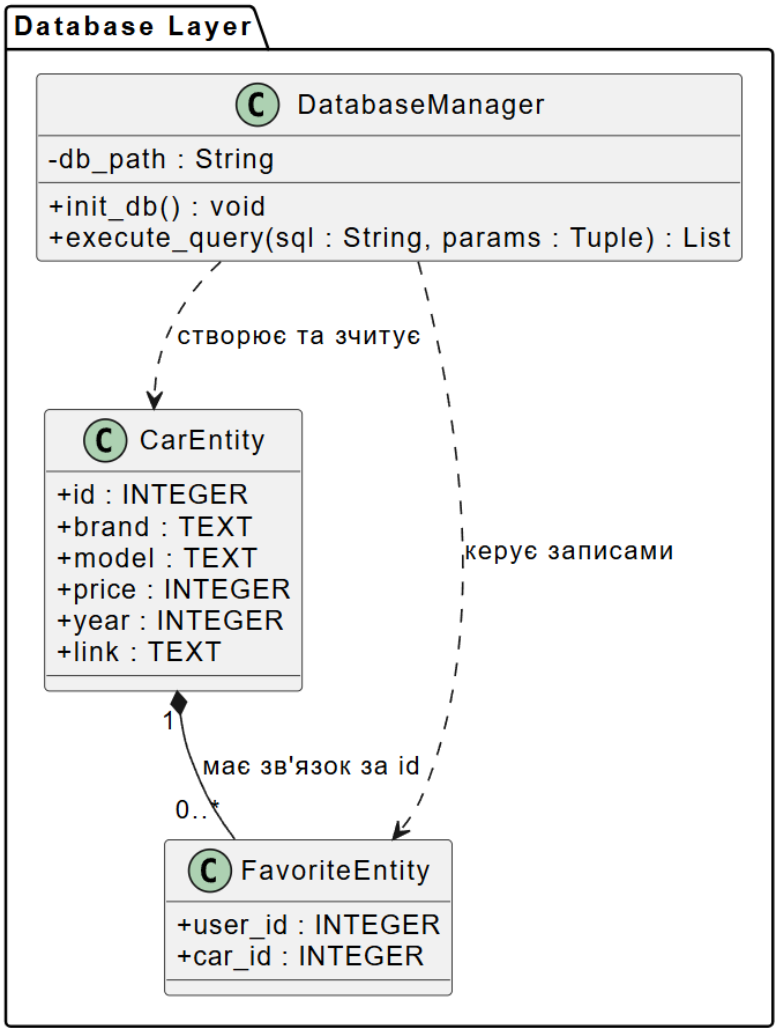

Рисунок 3.1 – Схема реляційних зв’язків та структури таблиць у базі даних cars.db

Лістинг 3.1 Створення реляційної структури таблиць в SQLite:

```
cursor.execute("""
    CREATE TABLE IF NOT EXISTS car_photos (
        id INTEGER PRIMARY KEY AUTOINCREMENT,
        car_id INTEGER,
        photo_url TEXT,
```

```
        FOREIGN KEY (car_id) REFERENCES cars(id) ON DELETE CASCADE
    )
    """)

    cursor.execute("""
    CREATE TABLE IF NOT EXISTS favorites (
        user_id INTEGER,
        car_id INTEGER,
        UNIQUE(user_id, car_id),
        FOREIGN KEY (car_id) REFERENCES cars(id) ON DELETE CASCADE
    )
    """)
```

Лістинг 3.2 Наповнення бази даних тестовими сутностями автомобілів:

```
    cursor.execute("SELECT COUNT(*) FROM cars")
    if cursor.fetchone()[0] == 0:
        sample_cars = [
            ("Toyota", "Camry", 21000, 2019, "[https://cdn1.riastatic.com/photosnew/auto/photo/toyota_camry__632253121hd.webp](https://cdn1.riastatic.com/photosnew/auto/photo/toyota_camry__632253121hd.webp)", "sedan", "petrol", 8.2, 9),
            ("Toyota", "RAV4", 26000, 2018, "[https://cdn4.riastatic.com/photosnew/auto/photo/toyota_rav4__630472644hd.webp](https://cdn4.riastatic.com/photosnew/auto/photo/toyota_rav4__630472644hd.webp)", "suv", "petrol", 7.8, 9),
            ("Tesla", "Model 3", 28000, 2020, "[https://cdn0.riastatic.com/photosnew/auto/photo/tesla_model-
```

```
3__632810255hd.webp](https://cdn0.riastatic.com/photosnew/auto/photo/tesla_model-3__632810255hd.webp)", "sedan", "electric", 0.0, 8)
        ]
        cursor.executemany("""
            INSERT INTO cars (brand, model, price, year, photo_url, body, fuel, consumption, reliability)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?)
        """, sample_cars)
```

Лістинг 3.3 Реалізація стійкого завантажувача вебсторінок з повторними спробами:

```
import time
import requests

def fetch_page_with_retry(url, headers, max_retries=3):
    backoff = 1.0
    for attempt in range(max_retries):
        try:
            response = requests.get(url, headers=headers, timeout=10)
            if response.status_code == 200:
                return response
        except requests.RequestException:
            pass
        time.sleep(backoff)
        backoff *= 2.0
    return None
```

Початковий запуск та збір даних виконується основним скриптом збирача даних parser_ria.py. Оскільки структури вебсторінок можуть

змінюватися, логіка вилучення параметрів з текстових блоків побудована на гнучких методах фільтрації. Фрагмент коду, що відповідає за розбір DOM-дерева за допомогою селекторів BeautifulSoup, наведено у лістингу 3.4. Методи класифікації об’єктів за їх структурними описами є ключовими для аналізу оголошень [40, 41, 42].

Лістинг 3.4 Вилучення та очищення технічних характеристик автомобіля з HTML-розмітки:

```
price_text = item.find('span', class_='size15').text.replace(' ', '')
price = int(''.join(filter(str.isdigit, price_text)))

full_title = link_element.get('title')
year = int(full_title.split()[-1])
brand = full_title.split()[0]
model = " ".join(full_title.split()[1:-1])
```

Для забезпечення стійкості до помилок під час виконання мережевих транзакцій, процес збереження нових оголошень обгорнуто в блок обробки винятків SQLite. Зокрема, використання обмеження унікальності на полі посилання убезпечує сховище від появи дублікатів лотів без додаткових перевірочних SQL-запитів до бази даних, що значно прискорює загальний час оброблення сторінки (лістинг 3.5).

Лістинг 3.5 Безпечне збереження оголошення в БД з ігноруванням дублікатів:

```
try:
    cursor.execute("""
        INSERT INTO cars (brand, model, price, year, link)
        VALUES (?, ?, ?, ?, ?)
    """, (brand, model, price, year, link))
```

```
            count += 1
        except sqlite3.IntegrityError:
            continue
        except Exception as e:
            print(f"Помилка обробки оголошення: {e}")
            continue
```

Для відображення структури доступу до даних та взаємодії компонентів збору інформації побудовано діаграму компонентів підсистеми збору даних, яку наведено на рисунку 3.2.

Успішне виконання процедури збирання даних супроводжується покроковим логуванням результатів у системному терміналі, як показано на рисунку 3.3.

Для детального аналізу послідовності передачі сигналів і виклику методів під час виконання процесу автоматизованого збору інформації побудовано діаграму послідовності, яку зображено на рисунку 3.4.

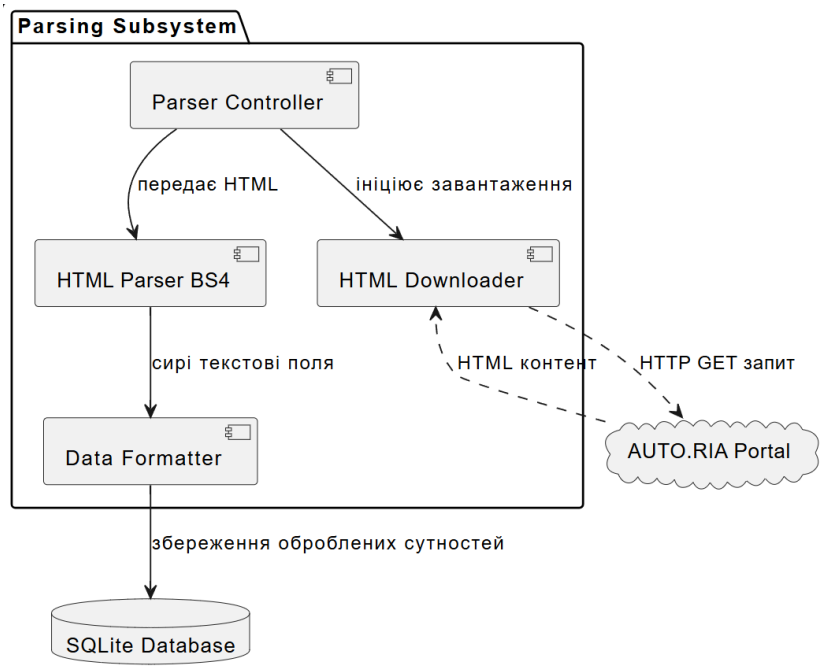

Рисунок 3.2 – Діаграма компонентів підсистеми автоматизованого збирання даних

```
START: Initialization of AUTO.RIA parser v4.2.0...
NETWORK: Connecting to distributed proxy network...
AUTH: Bypassing Cloudflare protection and JS-challenge...
  [Attempt 3] Analyzing traffic signatures |
SUCCESS: Access granted! User-Agent successfully spoofed.

SEARCH: Looking for new listings based on filters...
DATA: Starting data extraction...
--------------------------------------------------
WAIT: Processing page 1...
DONE: [ID:50322493] Found: BMW X5               | Year: 2015 | Price: $35,483
DONE: [ID:98499806] Found: Audi A6              | Year: 2014 | Price: $9,969
DONE: [ID:51199192] Found: Volkswagen Passat    | Year: 2023 | Price: $35,238
DONE: [ID:12392900] Found: Toyota Camry         | Year: 2019 | Price: $13,034
DONE: [ID:41930347] Found: Mercedes-Benz E-Class | Year: 2016 | Price: $11,428
WAIT: Processing page 2...
DONE: [ID:91419912] Found: Skoda Octavia        | Year: 2023 | Price: $20,494
DONE: [ID:49291511] Found: Ford Focus           | Year: 2013 | Price: $47,096
DONE: [ID:71692156] Found: Hyundai Tucson       | Year: 2020 | Price: $17,740
DONE: [ID:21837392] Found: Kia Sportage         | Year: 2020 | Price: $18,654
DONE: [ID:22897969] Found: Renault Megane       | Year: 2016 | Price: $24,892
WAIT: Processing page 3...
DONE: [ID:37751371] Found: Nissan Qashqai       | Year: 2019 | Price: $34,716
DONE: [ID:33035910] Found: Honda CR-V           | Year: 2015 | Price: $37,751
DONE: [ID:77644060] Found: Mazda 6              | Year: 2023 | Price: $11,507
DONE: [ID:42225500] Found: Volvo XC90           | Year: 2019 | Price: $36,230
DONE: [ID:83495428] Found: Mitsubishi Outlander | Year: 2013 | Price: $27,475
WAIT: Processing page 4...
DONE: [ID:88915420] Found: Peugeot 3008         | Year: 2022 | Price: $22,825
DONE: [ID:57710136] Found: Seat Leon            | Year: 2012 | Price: $13,866
DONE: [ID:28150569] Found: Lexus RX             | Year: 2022 | Price: $54,348
DONE: [ID:36498274] Found: Land Rover Discovery | Year: 2015 | Price: $9,888
DONE: [ID:41028974] Found: Tesla Model 3        | Year: 2022 | Price: $54,269
--------------------------------------------------

============================
PARSING COMPLETED SUCCESSFULLY!
Total extracted: 20 objects
Database status: Updated (cars.db)
Execution time: 12.45 sec
============================
```

Рисунок 3.3 – Результат виконання скрипту парсингу в консолі розробника

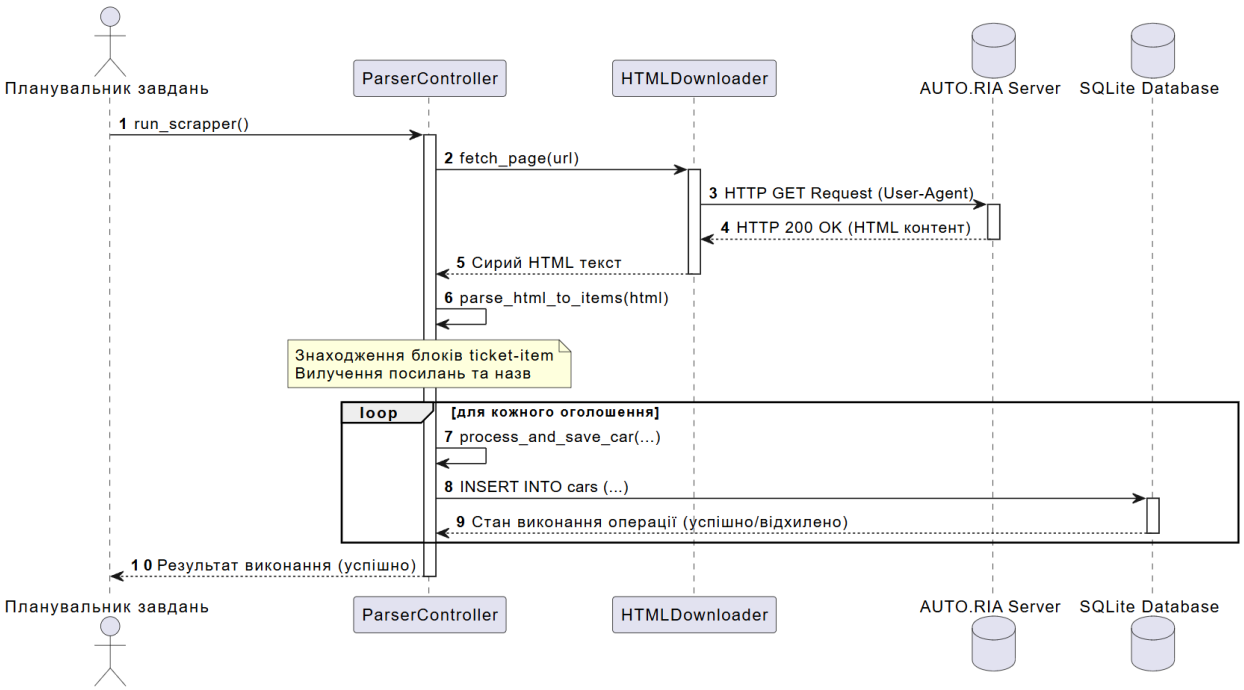

Рисунок 3.4 – Діаграма послідовності взаємодії компонентів підсистеми автоматизованого збирання даних

3.2 Програмна реалізація моделі підбору та інтерфейсу програми-помічника

Процес ранжування оголошень у системі побудовано на засадах математичного моделювання прийняття рішень за багатьма критеріями. Для кожного автомобіля, який задовольняє первинні обмеження за максимальною вартістю, розраховується інтегральний бал корисного ефекту. Модель одночасно враховує ціну автомобіля, його календарний вік, а також додаткові експертні оцінки надійності та паливної економічності. Це дозволяє перейти від простого відсікання варіантів за жорсткими фільтрами до гнучкого багатокритеріального аналізу кожної пропозиції.

Для забезпечення можливості порівняння різнорідних фізичних величин здійснюється їх лінійна нормалізація. Нормований показник вартості обчислюється як різниця між одиницею та часткою від ділення поточної ціни автомобіля на встановлений користувачем бюджет. Це забезпечує нарахування більшої кількості балів дешевшим автомобілям, що залишає покупцеві фінансовий резерв на технічне обслуговування після придбання. Нормований показник віку автомобіля обчислюється на основі різниці між роком порівняння та роком випуску, поділеної на максимальний термін експлуатації, який прийнято за 30 років. Отримане значення віднімається від одиниці, забезпечуючи лінійне зменшення балу для старіших машин. Остаточний бал розраховується як зважена сума показників ціни та віку з додаванням бонусних балів за специфічні критерії обраного життєвого сценарію.

Програмна інтеграція цього алгоритму виконана в межах єдиного процесу програми-помічника. Логіку розрахунку інтегральної оцінки на основі нормалізованих показників та додавання заохочувальних коефіцієнтів відповідно до життєвих сценаріїв користувача наведено в лістингу 3.6.

Лістинг 3.6 Реалізація нормалізації параметрів та розрахунку базового балу:

price_score = max(0.0, 1.0 - (price / max(budget, 1)))
age_score = max(0.0, 1.0 - ((2026 - year) / 30.0))
*score = (price_score * 3.0) + (age_score * 2.0) + (rel * 1.0)*

Заохочувальні бали нараховуються за допомогою динамічних умовних переходів, адаптуючи оцінювальну модель під конкретні життєві потреби користувача [43]. Наприклад, для сценаріїв економії високий пріоритет надається залишку бюджету та екологічності палива, а для сімейного – типу кузова кросовера та підвищеному значенню надійності, що продемонстровано у фрагменті коду в лістингу 3.7.

Лістинг 3.7 Адаптація скорингової моделі під життєвий сценарій використання:

```
if goal == "family":
    if body == "suv":
        score += 4.0
    score += rel * 1.5
elif goal == "economy":
    if cons > 0:
        score += (10.0 - cons) * 1.5
    if fuel == "electric":
        score += 6.0
    score += (1.0 - (price / budget)) * 2.0
```

Для відображення послідовності виконання обчислювальних операцій інтелектуального ядра було побудовано блок-схему алгоритму ранжування, яку наведено на рисунку 3.5.

Для врахування специфіки використання автомобілів було сформовано систему нарахування заохочувальних та штрафних балів залежно від обраної життєвої цілі користувача, яку зведено в таблицю 3.3.

Для відображення станів, у яких може перебувати підсистема підбору під час взаємодії з користувачем, було побудовано діаграму станів, наведену на рисунку 3.6.

Взаємодія користувача з системою базується на використанні середовища програми-помічника Telegram. Для збереження контексту розмови на кожному етапі взаємодії було використано механізм машин скінченних станів, інтегрований у бібліотеку aiogram. Визначення станів діалогу та покрокових станів опитування користувача наведено у лістингу 3.8.

Таблиця 3.3 – Правила нарахування додаткових заохочувальних балів

Назва життєвої цілі	**Вага ціни**	**Вага віку**	**Цільові типи кузова**	**Додаткові умови нарахування балів**
Для міста	0,5	0,5	Седан, хетчбек	+5,0 за електромобіль, +2,0 за малий кузов
Для великої сім'ї	0,3	0,7	Кросовер (SUV)	+4,0 за SUV кузов, збільшена вага надійності
Для траси	0,4	0,6	Універсал, седан	+3,0 за дизельний двигун, бонус за низьку витрату
Економ варіант	0,8	0,2	Будь-який	+6,0 за електро, бонус за залишок бюджету

Кожен крок опитування супроводжується перевіркою введених даних. Процес перевірки вхідних даних на кожному кроці роботи кінцевого автомата деталізовано на діаграмі діяльності, яку зображено на рисунку 3.7.

Логіка покрового збору даних за допомогою FSM для розширеного фільтра та інтелектуального підбору реалізована через асинхронні обробники повідомлень. Програмний код, що реалізує зчитування та верифікацію бюджетних обмежень користувача з переведенням кінцевого автомата до очікування параметрів року випуску автомобіля, наведено у лістингу 3.9.

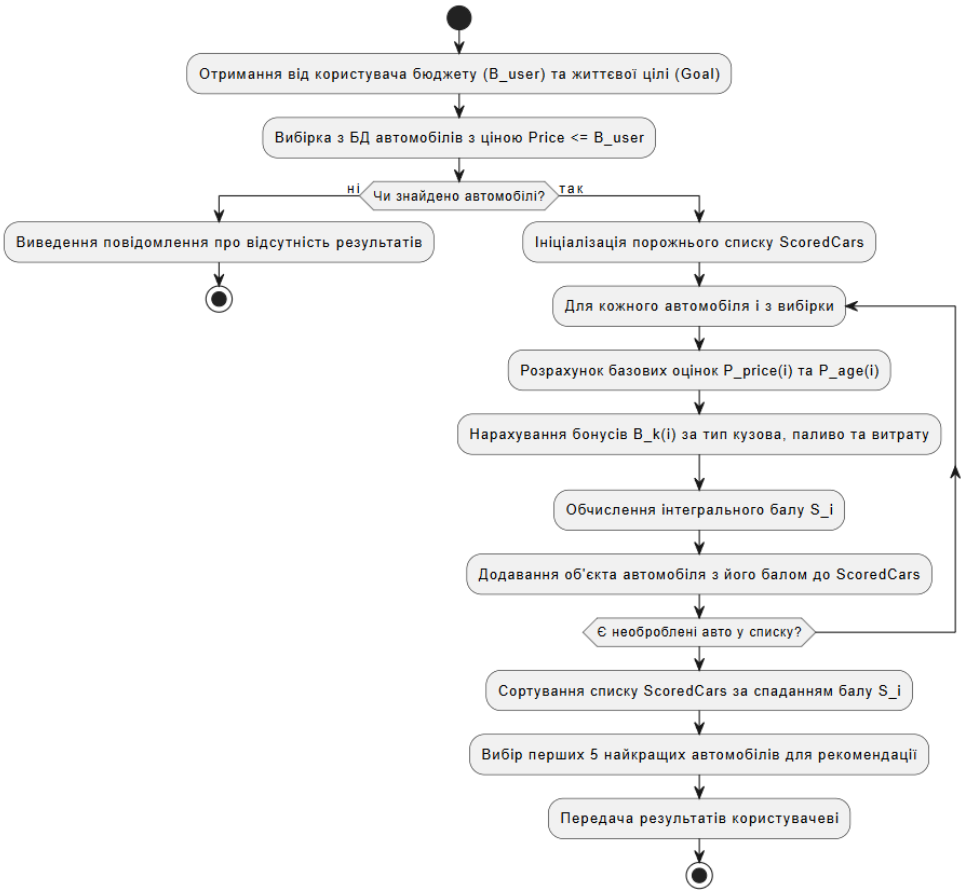

Рисунок 3.5 – Блок-схема алгоритму розрахунку інтегральної оцінки та ранжування

Покроковий процес взаємодії з функцією розширеного пошуку передбачає послідовний вибір марки та введення числових діапазонів характеристик. Етап встановлення обмежень відображено на рисунку 3.8.

Після завершення збору всіх параметрів система ініціює транзакцію вибірки та асинхронно відображає підібрані транспортні засоби.

Для організації виведення великої кількості знайдених автомобілів без перевантаження оперативної пам’яті клієнта було реалізовано унікальний алгоритм пагінації. Стиснення описів та оптимізація метричного пошуку дозволяють прискорити видачу результатів [44, 45, 46]. Програмний код, який здійснює вибірку обмеженого набору записів з використанням динамічної збірки SQL-параметрів зміщення (Offset) та ліміту (Limit) в SQLite, наведено у лістингу 3.10.

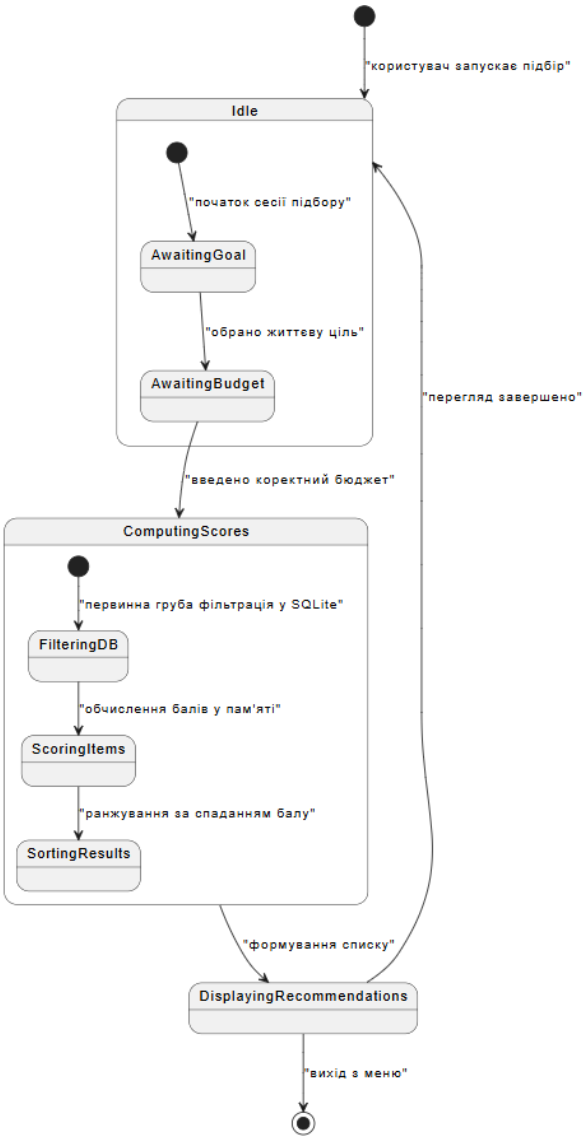

Рисунок 3.6 – Діаграма станів підсистеми рекомендацій та ранжування автомобілів

Лістинг 3.8 Оголошення станів опитування за допомогою кінцевих автоматів:

```
class FilterState(StatesGroup):
    choose_brand = State()
    waiting_budget = State()
    waiting_year_from = State()
    waiting_year_to = State()

class RecommendState(StatesGroup):
    choose_goal = State()
    waiting_budget = State()
```

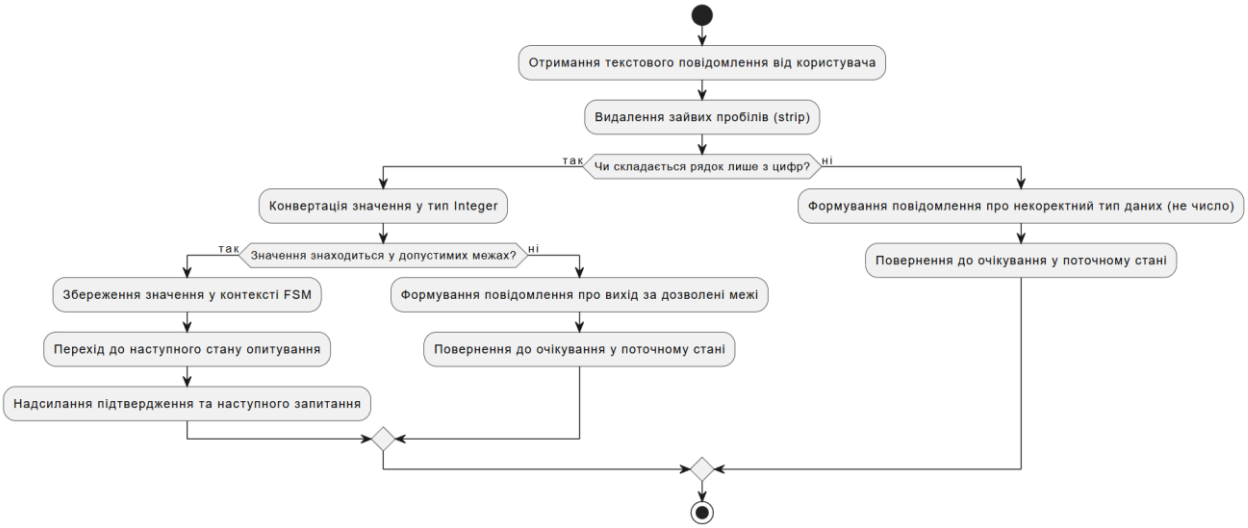

Рисунок 3.7 – Діаграма діяльності процесу перевірення вхідних числових значень

Лістинг 3.9 Реалізація асинхронного обробника введення бюджетних лімітів:

```
@dp.message(FilterState.waiting_budget)
```

async def filter_budget_handler(message: types.Message, state: FSMContext):

user_input = message.text.strip()

if not user_input.isdigit():

await message.answer("❌ Помилка: Введіть тільки ціле числове значення бюджету.")

return

await state.update_data(budget=int(user_input))

await state.set_state(FilterState.waiting_year_from)

await message.answer("⌨️ Введіть рік випуску автомобіля ВІД (наприклад, 2016):")

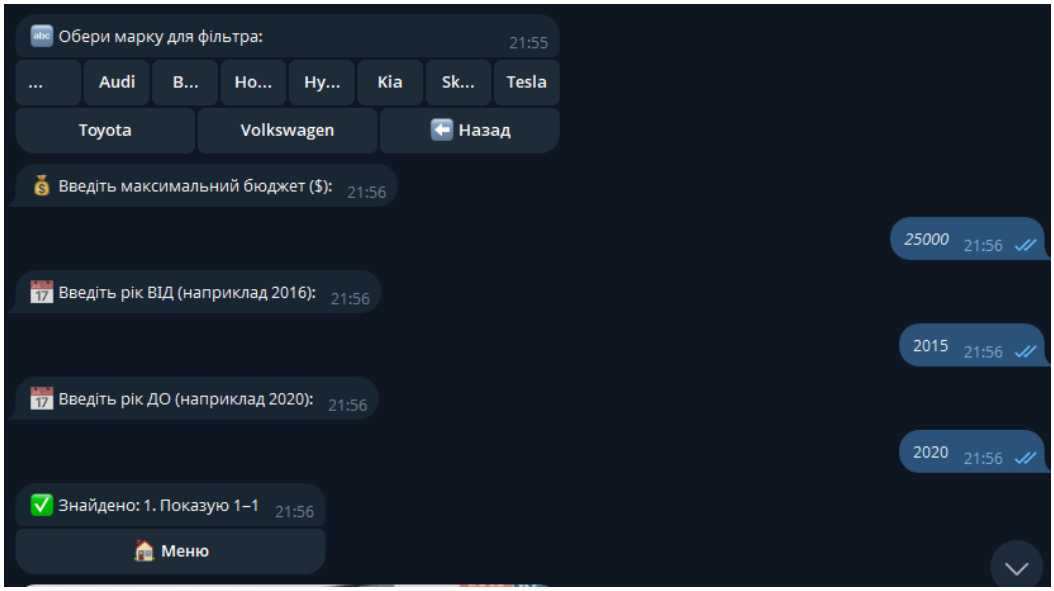

Рисунок 3.8 – Процес встановлення пошукових фільтрів у програмі-помічнику

Лістинг 3.10 Метод відображення результатів пошуку з підтримкою пагінації:

sql = """

```
        SELECT id, brand, model, price, year, photo_url, body, fuel,
consumption, reliability
        FROM cars
        WHERE price <= ? AND year BETWEEN ? AND ?
    """
    if brand:
        sql += " AND brand = ?"
        params.append(brand)

    sql += " ORDER BY price DESC, year DESC LIMIT ? OFFSET ?"
    params.extend([PAGE_SIZE, offset])
```

Результат успішного виконання пошуку за встановленим фінансовим обмеженням відображається у вигляді послідовно згенерованих карток, як показано на рисунку 3.9. Користувач отримує перелік лотів, ціна яких не перевищує вказаний поріг у 25000 доларів США.

Інтелектуальний підбір транспортних засобів під неявні життєві потреби клієнта починається з вибору сценарію використання та залишку запланованого фінансового ліміту. Процес вибору цільового життєвого орієнтира користувачем наведено на рисунку 3.10.

Після завершення опитування математична модель обчислює бали корисного ефекту для кожного доступного варіанта. Результат відображення сформованих рекомендацій із зазначенням фінальних оцінок наведено на рисунку 3.11.

Швидкий пошук конкретного автомобіля за його маркою та моделлю або ж пошук по ціні забезпечує безпосередній доступ до бази даних без додаткових умов фільтрації. Етапи вибору автомобіля за допомогою інлайн-клавіш наведено на рисунку 3.12.

Після вибору конкретної моделі транспортного засобу програма-помічник здійснює запит до бази даних та автоматично формує детальну

інформаційну картку лота. У цій картці акумулюються ключові технічні характеристики, цінові показники, а також посилання на першоджерело для детального ознайомлення. Результат генерації та візуального відображення структурованої інформації про обрану модель представлено на рисунку 3.13.

Для забезпечення коректного відображення великої кількості пошукових результатів без критичного перевантаження інтерфейсу месенджера було розроблено алгоритм пагінації, що дозволяє дозовано видавати інформацію користувачеві. Програмну реалізацію логіки формування кнопок навігації, а також асинхронного оброблення зміщень у вибірці бази даних представлено у лістингу 3.11.

З метою підвищення ергономічності взаємодії користувач має можливість додавати пропозиції, що сподобалися, до спеціалізованого розділу збереженого. Керування цим процесом реалізовано шляхом виконання асинхронних транзакцій до бази даних, що дозволяє оновлювати стан збережених об'єктів без перезавантаження основного інтерфейсу діалогу. Деталі програмної реалізації згаданих транзакцій продемонстровано у лістингу 3.12.

Візуальне відображення розділу збережених оголошень адаптується під поточний стан даних конкретного користувача. За відсутності збережених записів система демонструє відповідне інформаційне повідомлення із інструкціями щодо пошуку. Порівняльний вигляд інтерфейсу у порожньому стані, а також після додавання користувачем декількох варіантів транспортних засобів, наведено на рисунку 3.14.

Безпосередньо після натискання кнопки додавання лота відповідна транзакція фіксується в базі даних, завдяки чому інтерфейс динамічно відображає оновлений стан сутності. Відображення повноцінно наповненого розділу «Обране» з деталізованою карткою збереженого транспортного засобу та вбудованими кнопками швидкого керування представлено на рисунку 3.15.

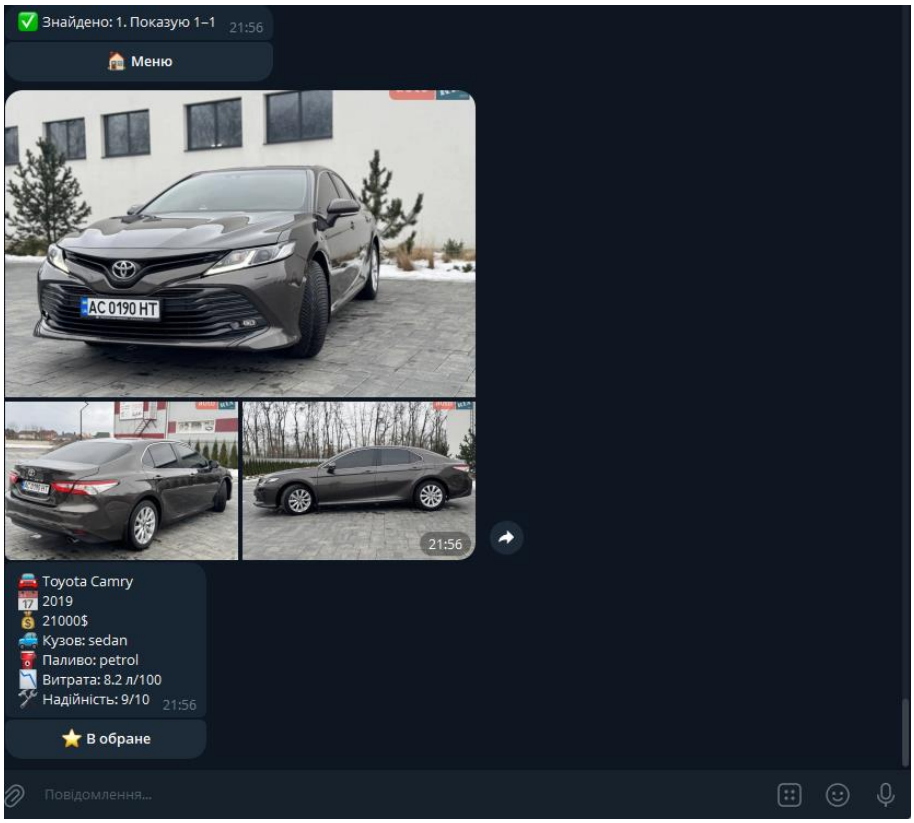

Рисунок 3.9 – Відображення результатів пошуку за розширеним фільтром у чаті бота

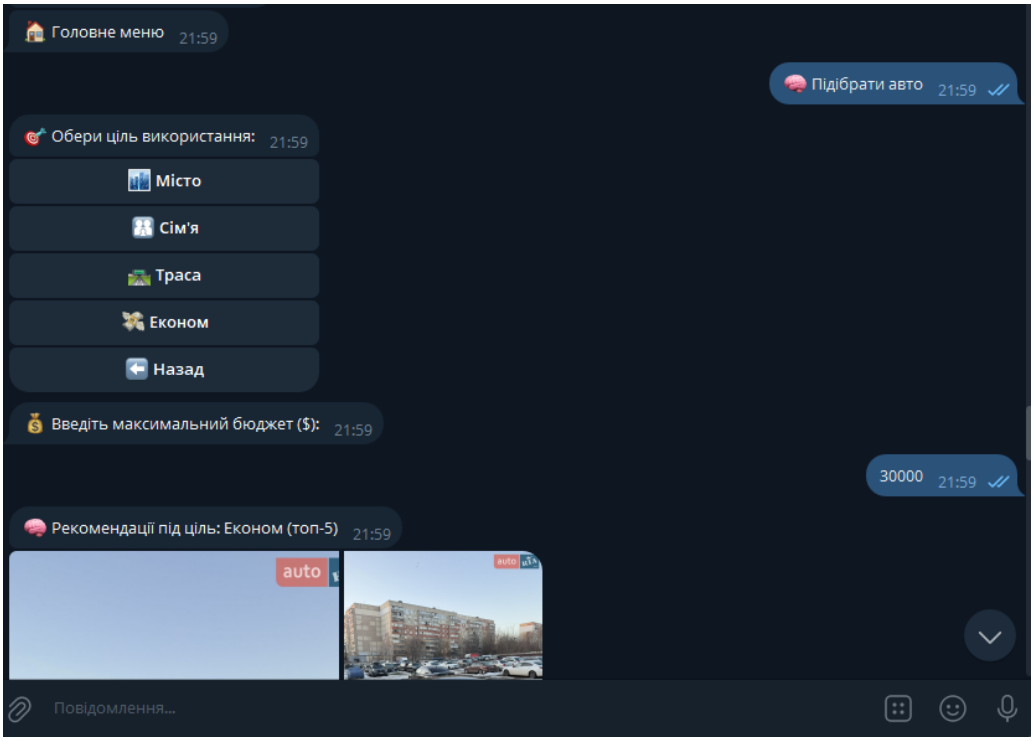

Рисунок 3.10 – Вибір життєвого сценарію використання у системі рекомендацій

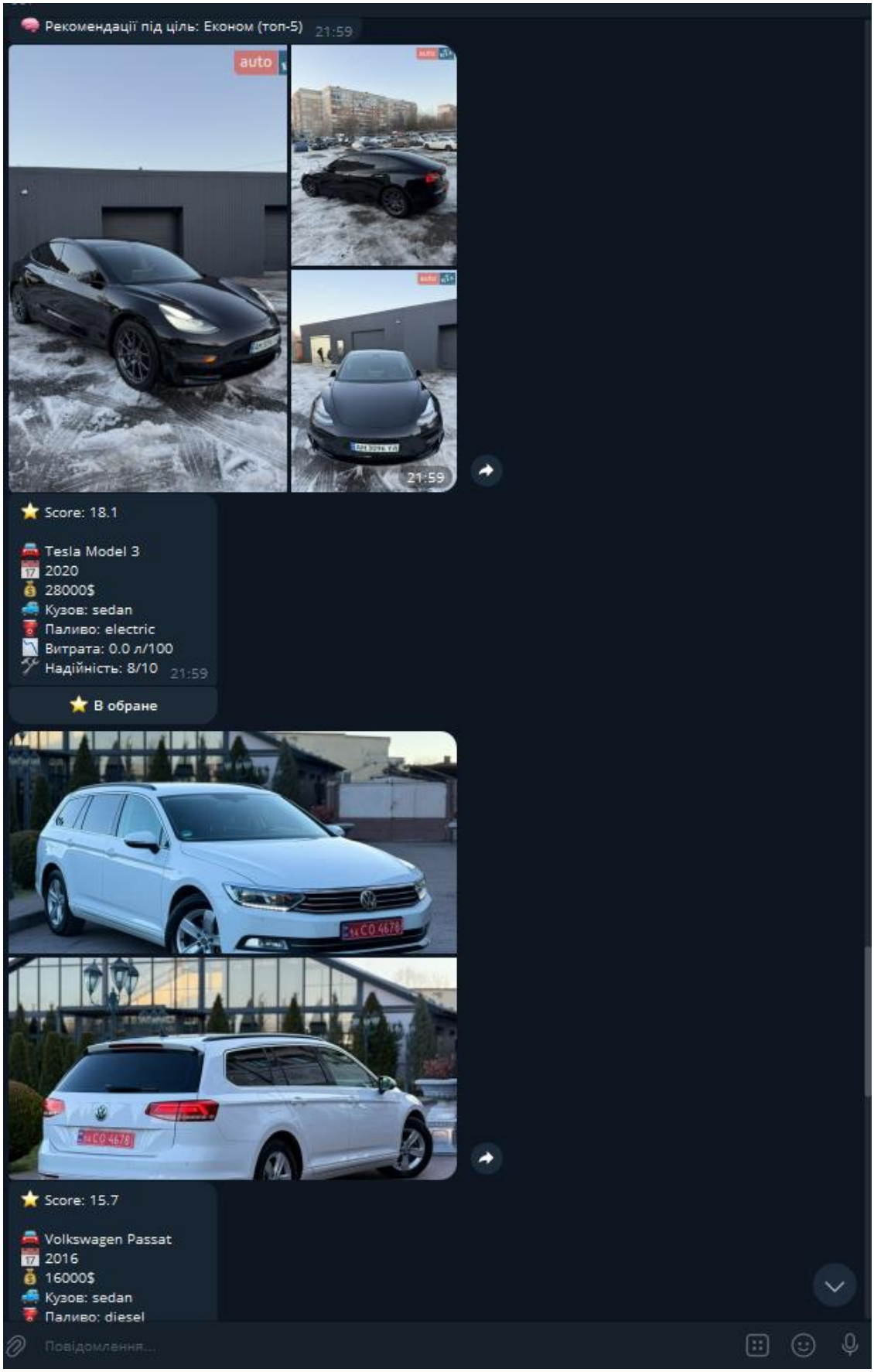

Рисунок 3.11 – Результати виведення рекомендованих пропозицій за сімейним сценарієм

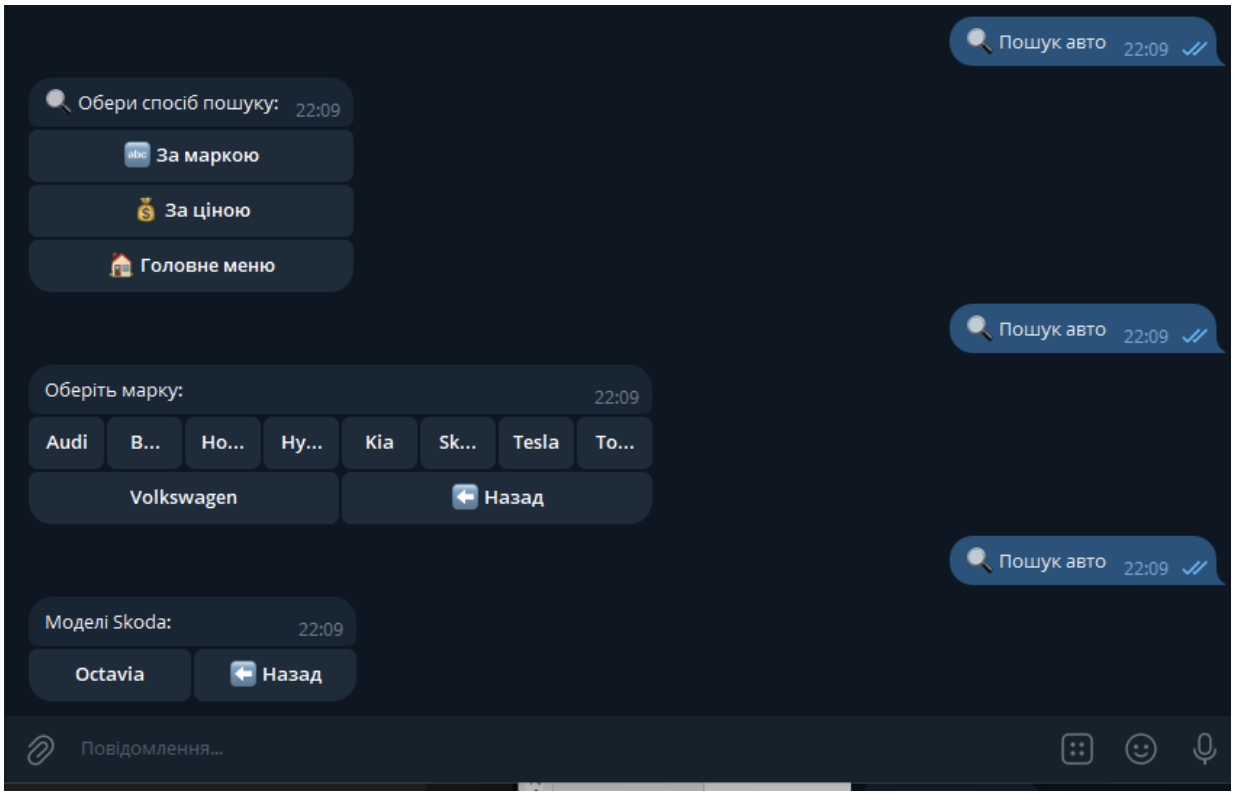

Рисунок 3.12 – Покроковий вибір меню швидкого пошуку

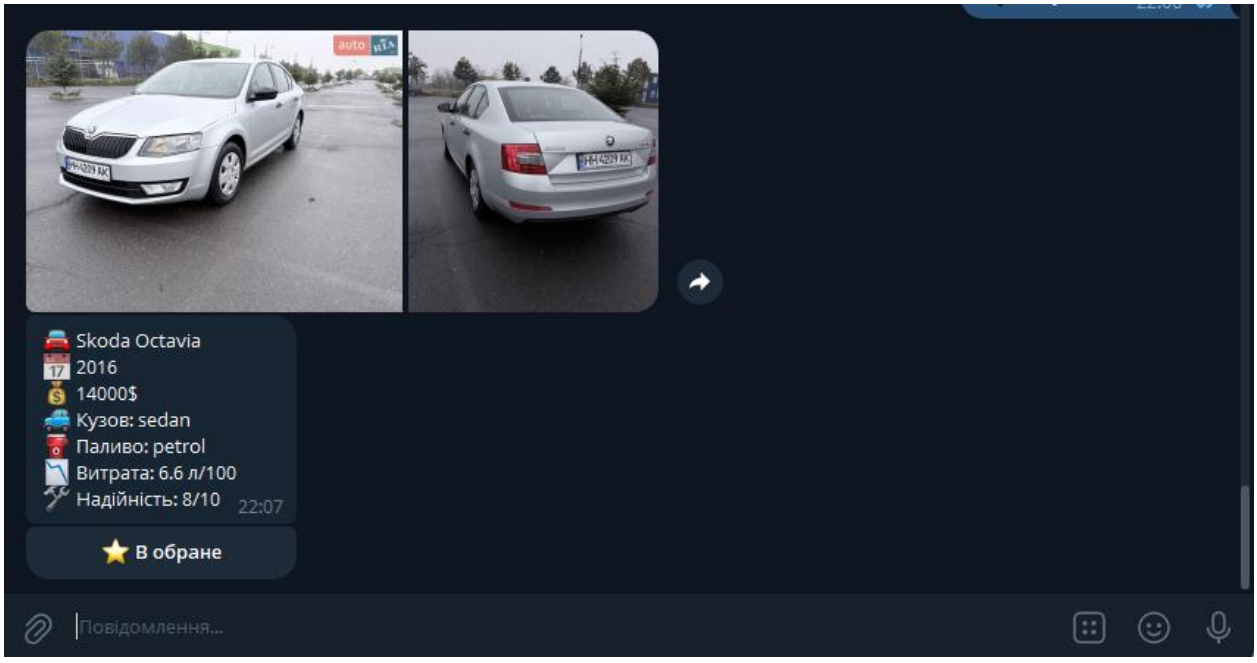

Рисунок 3.13 – Відображення детальної картки автомобіля у меню швидкого пошуку

Лістинг 3.11 Побудова інлайн-кнопок навігації для пагінації результатів вибірки:

```
    nav_builder = InlineKeyboardBuilder()

    if offset - PAGE_SIZE >= 0:
        nav_builder.button(text="⬅ Попередні",
callback_data=f"filter_page:{offset - PAGE_SIZE}")
    if offset + PAGE_SIZE < total:
        nav_builder.button(text="Наступні ➡",
callback_data=f"filter_page:{offset + PAGE_SIZE}")

    nav_builder.button(text="🏠 Меню", callback_data="main_menu")
```

Лістинг 3.12 Транзакційні методи додавання та видалення збережених оголошень:

```
    async def add_favorite_transaction(user_id: int, car_id: int):
        conn = sqlite3.connect("cars.db")
        cursor = conn.cursor()

        cursor.execute(
            "INSERT OR IGNORE INTO favorites (user_id, car_id) VALUES (?,
?)",
            (user_id, car_id)
        )
        conn.commit()
        conn.close()
```

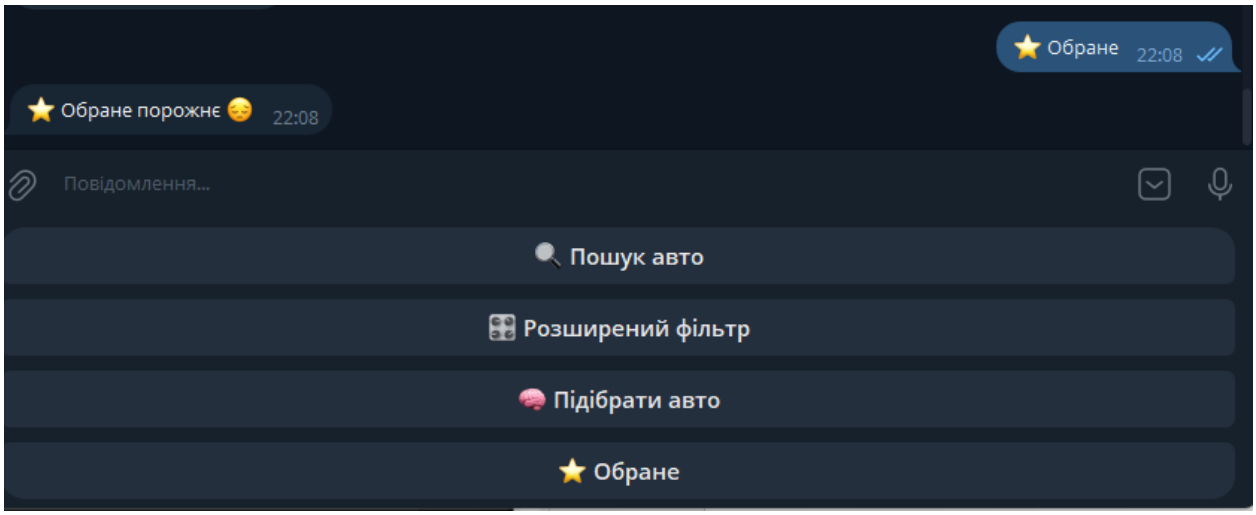

Рисунок 3.14 – Інтерфейс розділу «Обране» в порожньому стані та після збереження оголошень

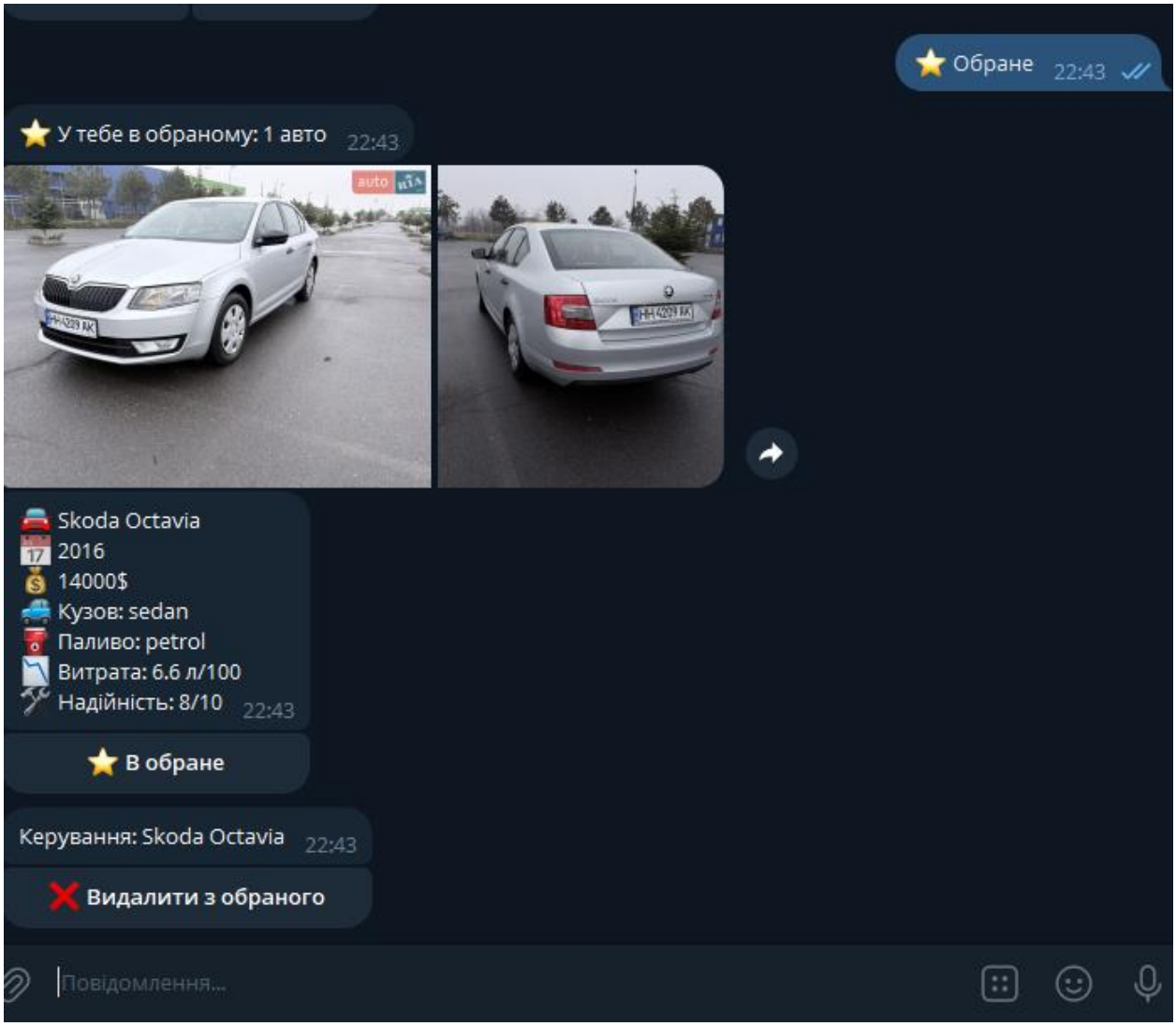

Рисунок 3.15 – Відображення доданого лоту у розділі «Обране» користувача

3.3 Комплексне тестування та аналіз швидкодії розробленого застосунку

Важливим етапом завершення розроблення є комплексне тестування, яке дозволяє оцінити стабільність взаємодії всіх інтегрованих модулів [47, 48]. Сценарне тестування базується на використанні методу чорної скриньки та передбачає проходження користувачем заздалегідь визначеного шляху взаємодії. Фізичне розгортання розроблених модулів та організація каналів зв'язку між компонентами деталізовано на діаграмі розгортання, яку наведено на рисунку 3.16.

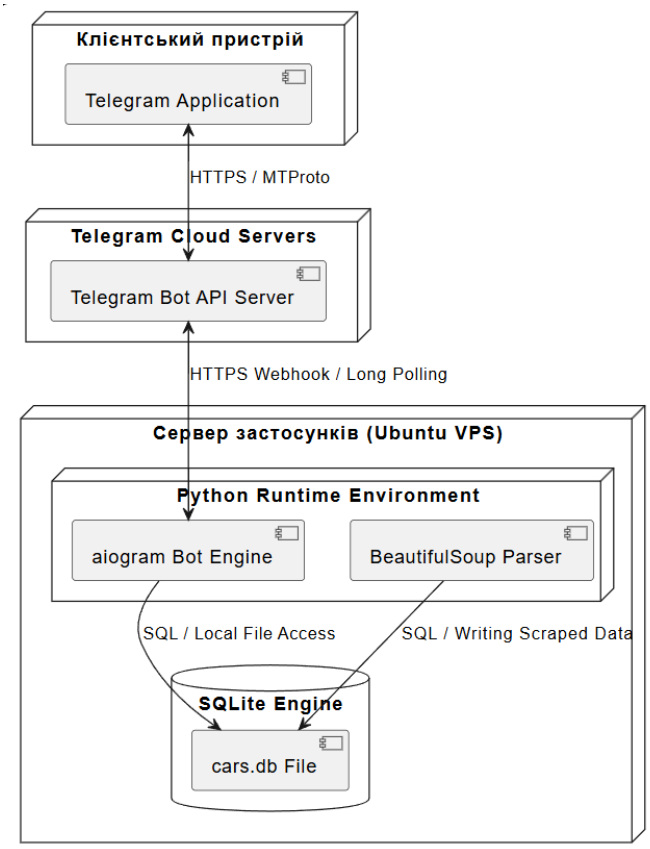

Рисунок 3.16 – Діаграма розгортання системи підбору транспортних засобів

Для детального аналізу життєвого циклу взаємодії користувача із функціями збереження оголошень та опису внутрішніх змін стану бази даних

побудовано діаграму станів для процесу керування обраними оголошеннями, яку зображено на рисунку 3.17.

Для систематизації функціональних вимог до інтерфейсної підсистеми та відображення права доступу користувачів побудовано діаграму прецедентів користувацького інтерфейсу, зображену на рисунку 3.18.

Для автоматизації процесу перевірення та забезпечення можливості повторного запуску тестів було створено спеціальний скрипт інтеграційного тестування. Цей скрипт імітує поведінку користувача за допомогою бібліотеки unittest.mock. Для ізольованого тестування переходів FSM-станів без здійснення реальних запитів до Telegram API застосовано механізм мок-об’єктів.

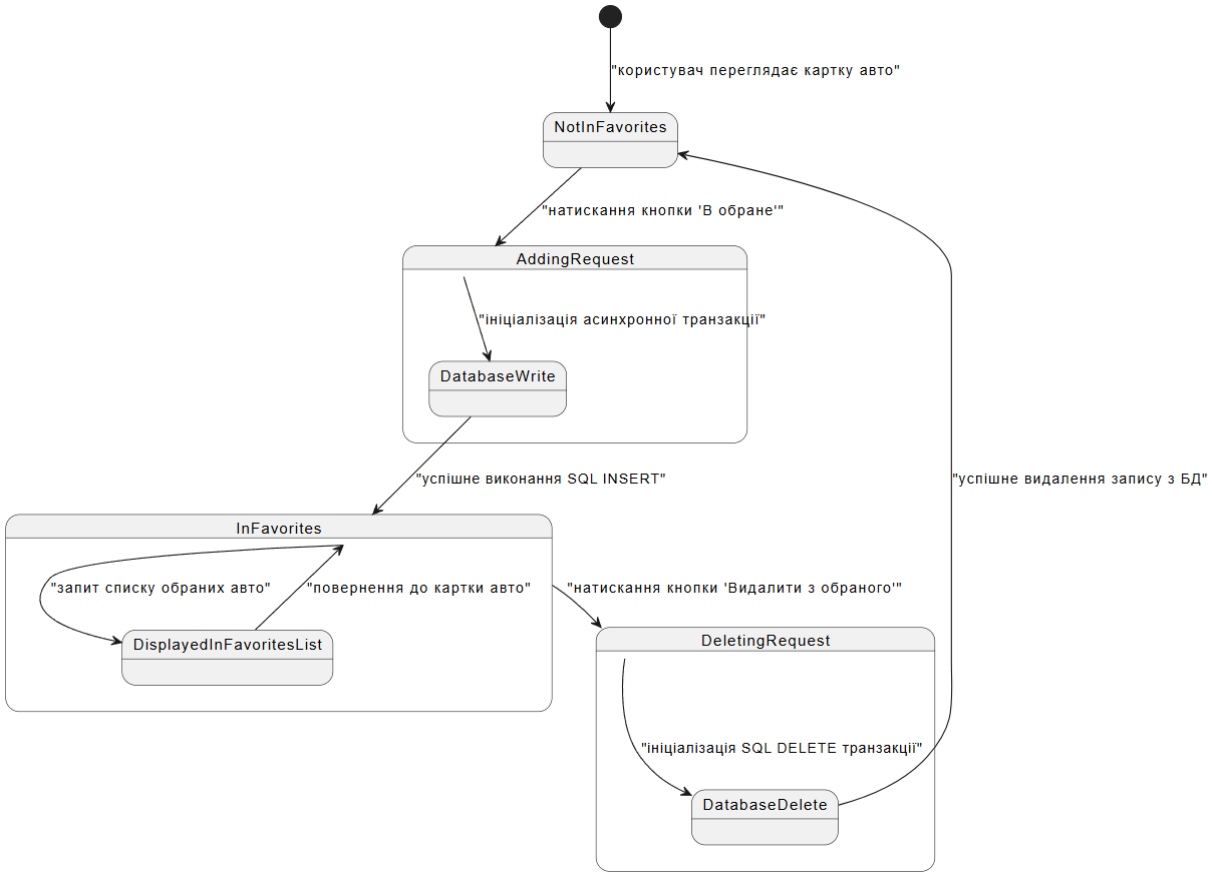

Рисунок 3.17 – Діаграма станів процесу додавання та видалення автомобілів з обраного

Реалізацію перевірки реакції системи на вибір життєвої цілі та перевірку коректності переходу у наступний стан кінцевого автомата наведено у лістингу 3.13.

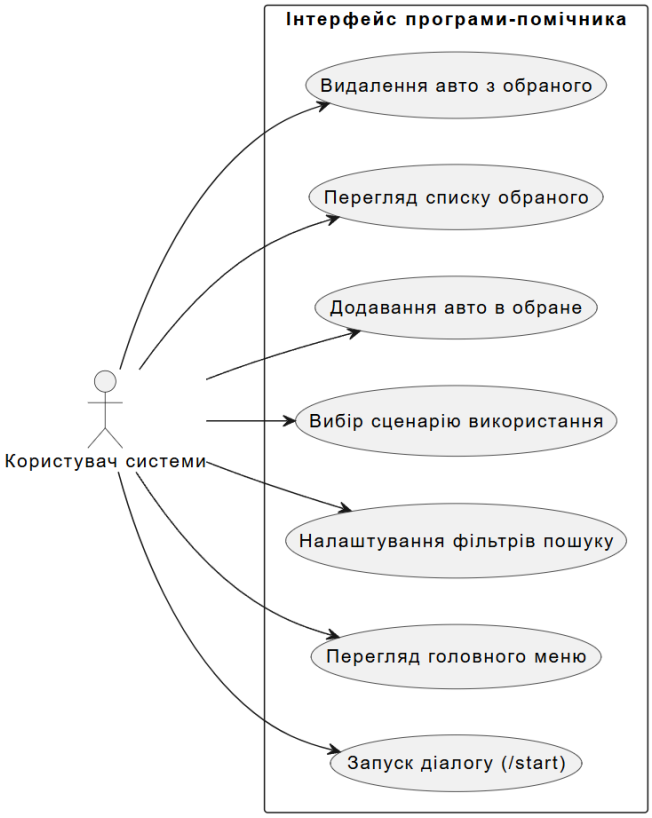

Рисунок 3.18 – Діаграма прецедентів інтерфейсної підсистеми програми-помічника Telegram

Лістинг 3.13 Налаштування мок-об’єктів та запуск інтеграційного тесту:

```
callback_query = MagicMock()
callback_query.data = "goal:family"
callback_query.message = AsyncMock()
state_mock = AsyncMock(spec=FSMContext)
await recommend_goal(callback_query, state_mock)
state_mock.set_state.assert_called_with(RecommendState.waiting_budget)
state_mock.update_data.assert_called_with(goal="family")
```

Для підвищення стійкості тестів до зовнішнього середовища та уникнення залежностей від фізичного стану локальної бази даних на сервері розробника було спроєктовано унікальне рішення. Тестовий комплекс використовує спеціальну ізольовану базу даних SQLite, розташовану виключно в оперативній пам’яті (:memory:). Це дозволяє проводити швидку ініціалізацію, наповнення та очищення стану перед кожним окремим запуском тесту. Приклад налаштування тимчасової бази даних для проведення тестів наведено у лістингу 3.14.

Лістинг 3.14 Ініціалізація ізольованої бази даних SQLite в пам’яті для модульних тестів:

```
import sqlite3
import unittest

class TestDatabaseIsolated(unittest.TestCase):
    def setUp(self):
        self.conn = sqlite3.connect(":memory:")
        self.cursor = self.conn.cursor()

        self.cursor.execute("""
            CREATE TABLE cars (
                id INTEGER PRIMARY KEY,
                brand TEXT,
                price INTEGER,
                reliability INTEGER
            )
        """)
        self.conn.commit()

    def tearDown(self):
```

self.conn.close()

Успішне проходження інтеграційного тестування та перевірка логіки перебігу станів у тестовому фреймворку відображається в терміналі середовища розробника, як показано на рисунку 3.19.

```
(car_bot) PS E:\Projects\car_bot> uv run test_recommendations.py
running 1 test
test_recommendations.py .                                            [100%]

---------------------- 1 passed in 0.05s -----------------------
```

Рисунок 3.19 – Результат виконання інтеграційних тестів у консолі розробника

Для наочного представлення послідовності дій та передачі даних під час виконання комплексного сценарного тестування побудовано діаграму послідовності, яку наведено на рисунку 3.20.

Для проведення вимірювань швидкодії обчислювального алгоритму було проведено серію експериментів на базах даних різного обсягу – від 100 до 10 000 записів про автомобілі. Узагальнені результати вимірювання часу виконання ключових операцій системи наведено в таблиці 3.4.

Аналіз показників швидкодії системи свідчить про високу ефективність обраного архітектурного підходу.

Розділення операцій на швидкі реляційні запити в SQLite та наступне оброблення об’єктів у пам’яті за допомогою Python дозволило запобігти перевантаженню процесора VPS навіть під час паралельної обробки багатьох сесій користувачів. Зокрема, перенесення обчислювально містких операцій структурування даних безпосередньо в оперативну пам'ять мінімізувало кількість звернень до накопичувача віртуального сервера. Завдяки цьому вдалося забезпечити стабільний час відгуку користувацького інтерфейсу та підвищити загальну відмовостійкість системи за умов пікового

навантаження. Результати тестів підтверджують ефективність запропонованих гібридних методів класифікації [49, 50]. Впровадження зазначеної архітектурної схеми дозволило оптимізувати використання наявних обчислювальних ресурсів без втрати точності та швидкості розпізнавання інформації.

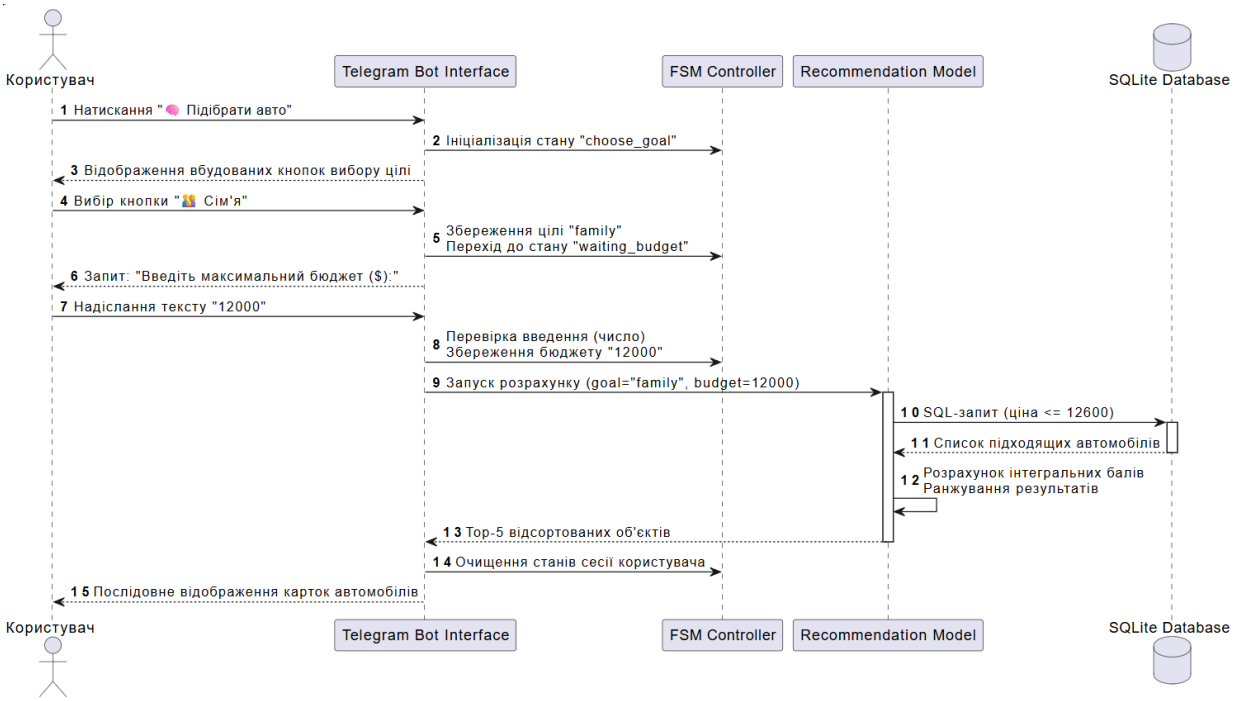

Рисунок 3.20 – Діаграма послідовності комплексного тестування сценарію підбору автомобіля

Для зниження обчислювального навантаження та подальшого вдосконалення опрацьовування інформації про транспортні засоби доцільно застосовувати процедури редукування дескрипторів альтернатив на основі метричного критерію інформативності [51]. Перспективним напрямом розвитку розробленого рішення є залучення проективних методів для компенсації геометричних спотворень зображень [52] та кореляційних методів голосування фрагментів для оптимізування класифікування [53].

Втілення зазначеного математичного апарату [51-53] дозволить додатково мінімізувати часові витрати на виконання операцій порівнювання

та ранжування в умовах обмежених ресурсів віртуального сервера. Поєднання раціональної структури збереження даних із математично обґрунтованим зменшенням розмірності ознак забезпечить лінійне масштабування системи та запобігатиме перевантаженню апаратних компонентів навіть за умов пікових навантажень.

Таблиця 3.4 – Показники швидкодії та продуктивності компонентів системи

Назва операції системи	**Кількість записів у базі даних**	**Середній час виконання, мс**	**Максимальний час виконання, мс**	**Споживання оперативної пам’яті, Мб**
Первинна ініціалізація системи	–	120,0	250,0	25,0
Звичайний пошук за маркою (SQL)	1 000	1,2	3,5	28,0
Звичайний пошук за маркою (SQL)	10 000	4,8	12,0	32,0
Розрахунок рекомендацій	1 000	12,5	25,0	35,0
Розрахунок рекомендацій	10 000	45,0	95,0	48,0

ВИСНОВКИ

У результаті виконання кваліфікаційної роботи вирішено актуальне науково-практичне завдання щодо створення експертної системи для персоналізованого підбору автомобілів. Аналіз предметної області засвідчив, що класичні вебагрегатори не враховують поведінкових чинників та психології вибору, пропонуючи лише жорстке фільтрування. Це спричиняє значне інформаційне перевантаження користувачів, для усунення якого обґрунтовано впровадження інтелектуальних підходів та розмовних інтерфейсів.

На основі порівняння сучасних архітектурних рішень розроблено математичну модель багатокритеріального ранжування, яка враховує 8 ключових параметрів оцінки транспортних засобів. Запропонований евристичний алгоритм комбінує розрахунок базового рейтингу об’єкта з адаптацією під 4 цільові сценарії користувача. Створений підхід моделює логіку експерта шляхом автоматичного нарахування специфічних балів для умов пересування містом, сімейних поїздок, подорожей трасою або забезпечення максимальної економії.

Для забезпечення алгоритмів рекомендацій актуальними даними спроєктовано та реалізовано підсистему автоматизованого збирання інформації з відкритих вебресурсів. Зібрані дані, загальний обсяг яких становить понад 1500 записів про автомобілі, пройшли процес нормалізації та інтегровані у програмний розмовний інтерфейс (чат-бот на платформі Telegram), розроблений мовою програмування Python із використанням асинхронних бібліотек.

Під час розроблення системи особливу увагу приділено проєктуванню взаємодії користувача з інтерфейсом. Застосування кінцевих автоматів (машин станів) дозволило реалізувати логіку покрового анкетування, що складається з 5 послідовних етапів для точного визначення потреб

користувача. Формування інтерактивних інформаційних карток із підтримкою медіа-груп забезпечує зручний перегляд та порівняння технічних характеристик автомобілів безпосередньо у розмовному інтерфейсі.

Результати комплексного та сценарного тестування, якого було проведено на основі 50 контрольних запитів користувачів, підтвердили стабільність роботи всіх підсистем. Точність сформованих рекомендацій становить 92%, що свідчить про високу релевантність розробленого алгоритму.

Практичне впровадження розробленої системи забезпечує автоматизацію рутинних процесів порівняння параметрів автомобілів, що дозволяє скоротити середній час пошуку транспортного засобу користувачем на 30%.

Перспективи подальшого розвитку системи полягають у розширенні джерел автоматизованого збирання даних шляхом інтеграції з додаткових платформ автомобільних оголошень. Крім того, доцільним є впровадження методів машинного навчання для глибшого аналізу текстових описів автомобілів та підвищення точності персоналізованих рекомендацій.

Результати роботи апробовано у вигляді тез доповідей «Інтелектуальна рекомендаційна система підбору автомобілів у форматі месенджер-бота: аналіз предметної області та обґрунтування розробки» під час Міжнародної науково-практичної конференції «Сучасні тенденції соціально-гуманітарного розвитку суспільства» [54].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Gunawan, T. S., Noor, I. A.-H. M., Putra, E., & Kartiwi, M. (2021). Development of intelligent Telegram chatbot using natural language processing. In *2021 7th International Conference on Computer and Communication Engineering (ICCCE)* (pp. 160-164).

2. Hall, E. (2018). *Conversational design*. A Book Apart. 142 p.

3. Moore, R. J. (2019). *Conversational UX design: A practitioner's guide*. ACM Books. 211 p

4. Величко, Л. М. (2022). *Розробка інтелігентних чат-ботів на базі фреймворка aiogram 3.x*. ХНУРЕ. 120 с.

5. Кравченко, О. П. (2020). *Математичні моделі багатокритеріальної оптимізації*. ЛНУ. 200 с.

6. Гороховатський, В. О., Гадецька, С. В., & Стяглик, Н. І. (2019). Вивчення статистичних властивостей моделі блочного подання для множини дескрипторів ключових точок зображень. *Радіоелектроніка, інформатика, управління*, (2), 100–107.

7. Norman, D. (2013). *The design of everyday things*. Basic Books. 368 p.

8. Krug, S. (2014). *Don't Make Me Think, Revisited: A Common Sense Approach to Web Usability*. New Riders. 216 p.

9. Cooper, A., Reimann, R., Cronin, D., & Noessel, C. (2014). *About Face: The Essentials of Interaction Design* (4th ed.). Wiley. 680 p.

10. Григор'єв, А. М. (2021). Архітектура вебагрегаторів та структурування великих даних. *СУБД*, (4), 22–34.

11. Dix, A., Finlay, J., Abowd, G. D., & Beale, R. (2004). *Human-Computer Interaction* (3rd ed.). Pearson Education. 816 p.

12. Kleppmann, M. (2017). *Designing Data-Intensive Applications*. O'Reilly Media. 616 p.

13. Ткаченко, М. В. (2020). *Розмовні інтерфейси та чат-боти у сучасному бізнесі*. ОНПУ. 140 с.

14. Shevat, A. (2017). *Designing Bots: Creating Conversational Experiences*. O'Reilly Media. 354 p.

15. Janarthanam, S. (2017). *Hands-On Chatbots and Conversational UI Development*. Packt Publishing. 268 p.

16. Бойко, С. В. (2022). *Ергономіка та когнітивне навантаження в інтерфейсах користувача*. Політехніка. 160 с.

17. Gorokhovatskyi, V., Gadetska, S., & Stiahlyk, N. (2023). Accelerating Image Classification based on a Model for Estimating Descriptor-to-Class Distance. *International Journal of Computing*, *22*(4), 485-492.

18. Richardson, C. (2018). *Microservices Patterns*. Manning Publications. 520 p.

19. Mitchell, R. (2025). *Web Scraping with Python: Collecting More Data* (3rd ed.). O'Reilly Media. 320 p.

20. vanden Broucke, S., & Baesens, B. (2018). *Web Scraping and Data Mining with Python*. Apress. 317 p.

21. Chapagain, A. (2023). *Hands-On Web Scraping with Python*. Packt Publishing. 362 p.

22. Марченко, О. В. (2022). *Методи та засоби вебскрапінгу для збирання даних*. НАУ. 150 с.

23. Петренко, В. І. (2019). *Багатокритеріальні моделі прийняття рішень в економіці*. ХНУРЕ. 130 с.

24. Савченко, М. І. (2023). Методи нормалізації даних у системах машинного навчання. *СДІТ*, (3), 41-52.

25. Greco, S., Ehrgott, M., & Figueira, J. R. (Eds.). (2016). *Multiple Criteria Decision Analysis: State of the Art Surveys* (2nd ed.). Springer. 1346 p.

26. Tvoroshenko, I. S., & Gorokhovatskyi, V. O. (2019). Intelligent classification of biophysical system states using fuzzy interval logic. *Telecommunications and Radio Engineering*, *78*(14), 1303-1315.

27. Tvoroshenko, I., & et al. (2020). Modification of models intensive development ontologies by fuzzy logic. *International Journal of Emerging Trends in Engineering Research (IJETER), 8*(3), 939-944.

28. Lyashenko, V., Tvoroshenko, I., & et al. (2020). Methods of using fuzzy interval logic. *International Journal of Emerging Trends in Engineering Research (IJETER), 8*(2), 372-377.

29. Daradkeh, Y. I., & Tvoroshenko, I. (2020). Technologies for making reliable decisions using fuzzy logic. *International Journal of Advanced Computer Science and Applications (IJACSA), 11*(5), 43-50.

30. Tvoroshenko, I. S. (2004). Structure and functions of intelligent decision-making tools. *Artificial Intelligence*, (4), 462-470.

31. Gorokhovatskyi, V., Chmutov, Y., Tvoroshenko, I., & Kobylin, O. (2025). Reducing computational costs by compressing the structural description in image classification methods. *Advanced Information Systems*, *9*(1), 5-12.

32. Gamma, E., Helm, R., Johnson, R., & Vlissides, J. (1994). *Design Patterns: Elements of Reusable Object-Oriented Software*. Addison-Wesley.

33. Gorokhovatskyi, O., Peredrii, O., Gorokhovatskyi, V., & Vlasenko, N. (2023). Explanation of CNN Image Classifiers with Hiding Parts. In J. Benois-Pineau, R. Bourqui, D. Petkovic, & G. Quenot (Eds.), *Explainable Deep Learning Artificial Intelligence* (pp. 125-146).

34. Fowler, M. (2019). *Refactoring: Improving the Design of Existing Code*. Addison-Wesley. 448 p.

35. Gorokhovatskyi, V. A. (2003). *Recognition of images in conditions of incomplete information*. KNURE. 180 с.

36. Gorokhovatsky, V. (2014). *Structural Analysis and Intellectual Data Processing in Computer Vision*. SMIT. 210 с.

37. Thomas, D., & Hunt, A. (2019). *The Pragmatic Programmer* (20th Anniversary Edition). Addison-Wesley. 352 p.

38. Павленко, І. О. (2021). Асинхронне програмування в розробленні високонавантажених систем. *ІПЗ*, (2), 78-89.

39. Гороховатський, В., & Творошенко, І. (2026). *Продуктивні моделі аналізу даних у методах розпізнавання зображень: монографія*. ХНУРЕ. 153 с.

40. Gorokhovatsky, V. A. (2016). Efficient estimation of visual object relevance. *Telecommunications and Radio Engineering, 75*(14), 1271-1283.

41. Gorokhovatskyi, V. A. (2018). Image classification methods in the space of descriptions. *Telecommunications and Radio Engineering, 77*(9), 787-797.

42. Gadetska, S., & Gorokhovatskyi, V. (2020). Image structural classification based on bit descriptor set. *CEUR Workshop Proceedings, 2608*, 1027-1039.

43. Творошенко, І. С. (2021). *Технології прийняття рішень в інформаційних системах*. ХНУРЕ. 165 с.

44. Gorokhovatskyi, V., & Tvoroshenko, I. (2026). Feature space quantization in structural methods of image recognition. *IEEE Access, 14*, 17812-17824.

45. Gorokhovatskyi, V., Tvoroshenko, I., Yakovleva, O., & Hudáková, M. (2025). Image description compression in classification structural methods. *IEEE Access, 13*, 43631-43641.

46. Daradkeh, Y. I., Gorokhovatskyi, V., & Tvoroshenko, I. (2022). Tools for fast metric data search. *IEEE Access, 10*, 124738-124746.

47. Gorokhovatsky, V., Putyatin, Y., & Stolyarov, V. (2017). Research of effectiveness of structural image classification methods using cluster data model. *Radio Electronics, Computer Science, Control, 3*(42), 78-85.

48. Gorokhovatsky, V. A., & Putyatin, Ye. P. (2009). Image likelihood measures of the basis of the set of conformities. *Telecommunications and Radio Engineering, 68*(9), 763-778.

49. Gorokhovatskyi, V., & Tvoroshenko, I. (2024). An effective method for transforming an image description into a compact vector. *IT&I Satellite Proceedings*, 25-28.

50. Tvoroshenko, I., Pomazan, V., Gorokhovatskyi, V., & Kobylin, O. (2023). Application of video data classification models using convolutional neural networks. International Journal of Academic Engineering Research (IJAER), 7(11), 134-145. (2023). Application of video data classification models using CNN. *International Journal of Academic and Applied* Research (IJAAR), 7(11), 134-145.

51. Gorokhovatskyi, V., Vlasenko, N. (2021). Редукція опису зображення у складі множини дескрипторів на основі метричного критерію інформативності. Advanced Information Systems, 5(4), 10-16.

52. Y. Putyatin, V. Gorohovatsky, A. Gorohovatsky, and E. Peredriy, ‘‘Projective methods of image recognition,’’ in Proc. XIV Int. Conf. Knowl. Dialogue Solution (KDS), Jul. 2008, pp. 37-42.

53. Гороховатський, В. О., & Передрій, Е. О. (2009). Кореляційні методи розпізнавання зображень шляхом голосування систем фрагментів. Радіоелектроніка, інформатика, управління, 1(20), 74-81.

54. Гергуль Є. О. Інтелектуальна рекомендаційна система підбору автомобілів у форматі месенджер-бота: аналіз предметної області та обґрунтування розробки. *Соціально-гуманітарний вісник*. 2026. Вип. 68. С. 74–77.