ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА ШТУЧНОГО ІНТЕЛЕКТУ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «Віртуальний помічник з персоналізованими рекомендаціями товарів на основі методів штучного інтелекту»

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Штучний інтелект
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Олександр ГРУНСЬКИЙ
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконала:
здобувачка вищої освіти Олександр ГРУНСЬКИЙ
група ШІД-41

Керівник: Тетяна КИСІЛЬ
науковий ступінь, вчене звання

Рецензент:
науковий ступінь, вчене звання _____
Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Штучного інтелекту
Ступінь вищої освіти Бакалавр
Спеціальність Комп'ютерні науки
Освітньо-професійна програма Штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедрою Штучного інтелекту

_____ Ольга ЗІНЧЕНКО
«_____» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Грунському Олександру Сергійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Віртуальний помічник з персоналізованими рекомендаціями товарів на основі методів штучного інтелектів

керівник кваліфікаційної роботи Тетяна КИСІЛЬ старший викладач,_____
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «24» лютого 2025 р. № 56

2. Строк подання кваліфікаційної роботи «23» травня 2025 р.

3. Вихідні дані до кваліфікаційної роботи: науково-технічна література з веб-технологій (HTML5, CSS3, JavaScript), документація Pexels API, аналіз існуючих інтернет-магазинів та агрегаторів автозапчастин, вимоги до сучасних користувацьких інтерфейсів.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
Дослідження предметної області та аналіз існуючих рішень для пошуку автозапчастин.
Проектування архітектури та користувацького інтерфейсу клієнтського веб-додатку.

Програмна реалізація ключових функцій системи (пошук, багатомовність, робота з історією).
Інтеграція з зовнішніми сервісами та API для розширення функціоналу.
Розробка альтернативного клієнта у вигляді Telegram-бота.

5. Перелік графічного матеріалу: *презентація*
   1. Загальна архітектурна схема веб-додатку.
   2. Дизайн та структура ключових екранів користувацького інтерфейсу.
   3. Схемам алгоритму роботи Telegram-бота.

6. Дата видачі завдання «25» лютого 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної області та існуючих онлайн-сервісів для пошуку автозапчастин	25.02-07.03.25	Виконано
2	Вивчення сучасних frontend-технологій (HTML5, CSS3, JavaScript ES6+)	07.03-13.03.25	Виконано
3	Дослідження та аналіз зовнішніх API (Pexels API) для інтеграції в додаток	14.03-21.03.25	Виконано
4	Проектування архітектури клієнтського додатку та дизайну користувацького інтерфейсу (UI/UX)	21.03-26.03.25	Виконано
5	Розробка логіки інтелектуального пошуку та механізму генерації посилань на магазини	27.03-04.04.25	Виконано
6	Програмна реалізація основного функціоналу веб-додатку та Telegram-бота	05.04-11.04.25	Виконано
7	Оформлення пояснювальної записки до кваліфікаційної роботи (вступ, висновки, реферат)	12.04-20.04.25	Виконано
8	Підготовка демонстраційних матеріалів та презентації для захисту роботи	21.04-23.05.25	Виконано

Здобувачка вищої освіти _____ Олександр ГРУНСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Тетяна КИСІЛЬ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 48 стор., 14 рис., 10 табл., 20 джерел.

Мета роботи – розробка та реалізація клієнтської веб-системи, що підвищує ефективність та зручність процесу пошуку автомобільних запчастин шляхом надання користувачам інтелектуальних інструментів для ідентифікації деталей та агрегації релевантної інформації з різних джерел.

Об'єкт дослідження – процес взаємодії користувачів з веб-інтерфейсами для пошуку та підбору автомобільних компонентів, а також методи покращення цього процесу за допомогою сучасних frontend-технологій.

Предмет дослідження – клієнтська веб-система інтелектуального пошуку автомобільних запчастин, її архітектура, функціональні можливості та технології реалізації (HTML, CSS, JavaScript, зовнішні API).

Короткий зміст роботи: У роботі представлено результати дослідження та розробки веб-додатку "AI Car Parts Finder", спрямованого на спрощення процесу пошуку автомобільних запчастин. Запропоновано підхід, що дозволяє користувачам здійснювати пошук як за ідентифікаційним номером деталі, так і за описовими характеристиками автомобіля (марка, модель, рік) та назвою запчастини. Реалізовано багатомовний інтерфейс, можливість перемикання тем оформлення, збереження історії пошуків та автозаповнення для підвищення зручності користування.

Система інтегрується з зовнішнім API (Pexels) для візуального супроводу результатів пошуку зображенням відповідного автомобіля. Крім того, додаток генерує прямі посилання на сторінки пошуку популярних українських та міжнародних інтернет-магазинів автозапчастин.

Проведено аналіз використаних технологій та описано ключові аспекти програмної реалізації. Розроблена система є клієнтським додатком, що демонструє можливості створення ефективних пошукових інструментів без необхідності

розгортання складної серверної інфраструктури для основного функціоналу. Отримані результати мають практичну цінність для автовласників та осіб, що займаються підбором автозапчастин.

КЛЮЧОВІ СЛОВА: ВЕБ-ДОДАТОК, ПОШУК АВТОЗАПЧАСТИН, JAVASCRIPT, HTML, CSS, API, КОРИСТУВАЦЬКИЙ ІНТЕРФЕЙС, FRONTEND, PEXELS API, ІНТЕЛЕКТУАЛЬНИЙ ПОШУК.

ABSTRACT

Bachelor's qualification thesis text: (Leave space for actual count) 48 pp., (Count) 14 figs., 10 (Count) tables, 20 (Count) refs.

The aim of the work is the development and implementation of a client-side web system that enhances the efficiency and convenience of searching for automotive spare parts by providing users with intelligent tools for identifying components and aggregating relevant information from various sources.

The object of research is the process of user interaction with web interfaces for searching and selecting automotive components, as well as methods for improving this process using modern frontend technologies.

The subject of research is a client-side web system for the intelligent search of automotive spare parts, its architecture, functional capabilities, and implementation technologies (HTML, CSS, JavaScript, external APIs).

Summary of the work: This work presents the results of the research and development of the "AI Car Parts Finder" web application, aimed at simplifying the process of searching for automotive spare parts. An approach is proposed that allows users to search both by part identification number (Part ID) and by descriptive characteristics of the vehicle (make, model, year) and the part name. A multilingual interface, the ability to switch design themes, saving search history, and autocomplete for enhancing user convenience have been implemented.

The system integrates with an external API (Pexels) for visual support of search results with an image of the corresponding vehicle. Additionally, the application generates direct links to the search pages of popular Ukrainian and international online auto parts stores. An analysis of the technologies used is provided, and key aspects of the software implementation are described.

The developed system is a client-side application that demonstrates the potential for creating effective search tools without the need to deploy complex server-side

infrastructure for core functionality. The results obtained have practical value for car owners and individuals involved in the selection of auto parts.

KEYWORDS: WEB APPLICATION, AUTO PARTS SEARCH, JAVASCRIPT, HTML, CSS, API, USER INTERFACE, FRONTEND, PEXELS API, INTELLIGENT SEARCH.

ЗМІСТ

ВСТУП

Актуальність теми. Ринок автомобільних запчастин в Україні та світі демонструє стабільне зростання, що зумовлено постійним збільшенням кількості автомобілів та потребою в їх обслуговуванні та ремонті. Водночас, автовласники та автомайстерні часто стикаються зі значними труднощами при пошуку та підборі необхідних компонентів. До основних проблем можна віднести складність точної ідентифікації деталі за каталожними номерами або характеристиками автомобіля, трудомісткість порівняння цін та наявності у різних постачальників, а також ризик придбання несумісної або неякісної продукції.

Існуючі онлайн-платформи та інтернет-магазини пропонують широкий асортимент, однак їхні пошукові механізми не завжди є інтуїтивно зрозумілими або достатньо гнучкими для задоволення всіх потреб користувачів. Це призводить до значних витрат часу та зусиль на пошук. В таких умовах виникає гостра потреба у розробці інтелектуальних вебсистем, здатних автоматизувати та оптимізувати процес пошуку автомобільних запчастин, надаючи користувачам швидкий доступ до релевантної та достовірної інформації. Використання сучасних вебтехнологій та підходів, що імітують інтелектуальний аналіз, може суттєво покращити користувацький досвід та підвищити ефективність підбору необхідних деталей.

Мета роботи. Метою даної дипломної роботи є розробка вебдодатку системи інтелектуального пошуку автомобільних запчастин, який забезпечує користувачам зручний та ефективний інструмент для знаходження деталей за різними критеріями, такими як ідентифікаційний номер деталі (Part ID) або марка, модель, рік випуску автомобіля та назва запчастини. Система має на меті агрегувати інформацію з відкритих джерел, надавати короткий опис деталі, зображення відповідного автомобіля та актуальні посилання на українські та міжнародні інтернет-магазини для її можливого придбання.

Об'єкт дослідження. Процес пошуку та підбору автомобільних запчастин користувачами через вебінтерфейси. Аналіз методів структурування запитів та

представлення результатів для підвищення ефективності та зручності користувацького досвіду в даній предметній області.

Предметом дослідження є методи та технології розробки клієнтської частини (frontend) вебдодатку для інтелектуального пошуку автомобільних запчастин, включаючи проєктування користувацького інтерфейсу, реалізацію пошукової логіки на стороні клієнта, взаємодію з зовнішніми API для отримання додаткової інформації (зображень) та формування релевантних посилань на основі даних, введених користувачем.

Розробка такої системи дозволить не лише поглибити знання у сфері сучасних вебтехнологій (HTML, CSS, JavaScript), але й зробити внесок у покращення доступності інформації про автомобільні запчастини, спрощуючи процес їх пошуку для широкого кола користувачів. Очікується, що результатом роботи стане функціональний та інтуїтивно зрозумілий вебдодаток.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ПОШУКУ АВТОМОБІЛЬНИХ ЗАПЧАСТИН

1.1 Загальна характеристика ринку автомобільних запчастин

Ринок автомобільних запчастин є важливою складовою глобальної автомобільної індустрії. Він охоплює виробництво, дистрибуцію та продаж широкого спектру компонентів, необхідних для ремонту, обслуговування та модернізації транспортних засобів. З кожним роком кількість автомобілів у світі, зокрема і в Україні, зростає, що прямопропорційно збільшує попит на запчастини. Цей ринок характеризується високою конкуренцією, великою номенклатурою товарів (від дрібних кріплень до складних агрегатів) та наявністю як оригінальних (OEM), так і неоригінальних (aftermarket) деталей.

Для автовласників та сервісних центрів навігація в цьому різноманітті часто стає складним завданням. Необхідність точної ідентифікації деталі, сумісної з конкретною моделлю, роком випуску та модифікацією автомобіля, є ключовою. Помилка у виборі може призвести не тільки до фінансових втрат, але й до проблем з безпекою експлуатації транспортного засобу.

1.2 Проблеми та виклики при пошуку автомобільних запчастин онлайн

З розвитком інтернет-торгівлі значна частина ринку автозапчастин перемістилася в онлайн. Існує безліч інтернет-магазинів, каталогів та агрегаторів, що пропонують свої послуги. Однак, користувачі часто стикаються з низкою проблем:

- *Складність ідентифікації***.** Не завжди легко знайти точний каталожний номер деталі. Пошук за назвою або описовими характеристиками може давати нерелевантні результати через відмінності в термінології або неповні дані.
- *Велика кількість пропозицій***.** навіть якщо деталь ідентифікована правильно, користувач отримує величезну кількість пропозицій від різних виробників та продавців, що ускладнює порівняння та вибір.

- *Відсутність стандартизації описів***.** різні продавці можуть надавати різний рівень деталізації опису товару, що не дозволяє повноцінно оцінити його характеристики та сумісність.
- *Недостовірність інформації***..** існує ризик натрапити на неактуальну інформацію про наявність або ціну товару.
- *Часові витрати***.** процес пошуку, порівняння та перевірки сумісності може займати значну кількість часу.
- *Відсутність візуального підтвердження* часто користувачеві важко уявити, чи дійсно знайдена деталь підходить до його автомобіля, особливо якщо немає якісних зображень самої деталі або автомобіля, для якого вона призначена.

Ці проблеми підкреслюють необхідність створення більш інтелектуальних та користувацько-орієнтованих систем пошуку.

1.3 Огляд існуючих онлайн-сервісів пошуку автозапчастин

На українському та міжнародному ринках існує багато вебсайтів, що спеціалізуються на продажу автозапчастин. Розглянемо декілька типових підходів:

Великі інтернет-магазини. Пропонують широкі каталоги, пошук за VIN-кодом, маркою автомобіля, іноді за номером деталі. Їхні переваги – великий асортимент та розвинена логістика. Недоліки – іноді перевантажені інтерфейси, складність навігації для недосвідчених користувачів, пошук може бути не завжди гнучким. Серед найпопулярніших варіантів, можна виділити 3 інтернет магазини:

На ринку українських онлайн-магазинів автозапчастин вирізняються декілька великих гравців, кожен з яких має свої особливості, переваги та недоліки.

Exist.ua (рис. 1.1) є значним українським онлайн-магазином автозапчастин, який може похвалитися широким асортиментом продукції та розгалуженою мережею офісів по всій країні. Клієнтам пропонуються як оригінальні деталі, так і якісні аналоги, а також доступні консультації фахівців. Ця платформа забезпечує зручний вибір запчастин.

Серед популярних українських ресурсів також виділяється ***Dok.ua*** (рис. 1.2) відомий інтернет-магазин автозапчастин, мастил та шин, що функціонує на власній

онлайн-платформі. Магазин обіцяє швидку доставку та співпрацю лише з перевіреними постачальниками. Хоча Dok.ua є популярним, його цінова політика не завжди може бути найнижчою на ринку, а якість обслуговування клієнтів, як і у багатьох великих рітейлерів, іноді варіюється.

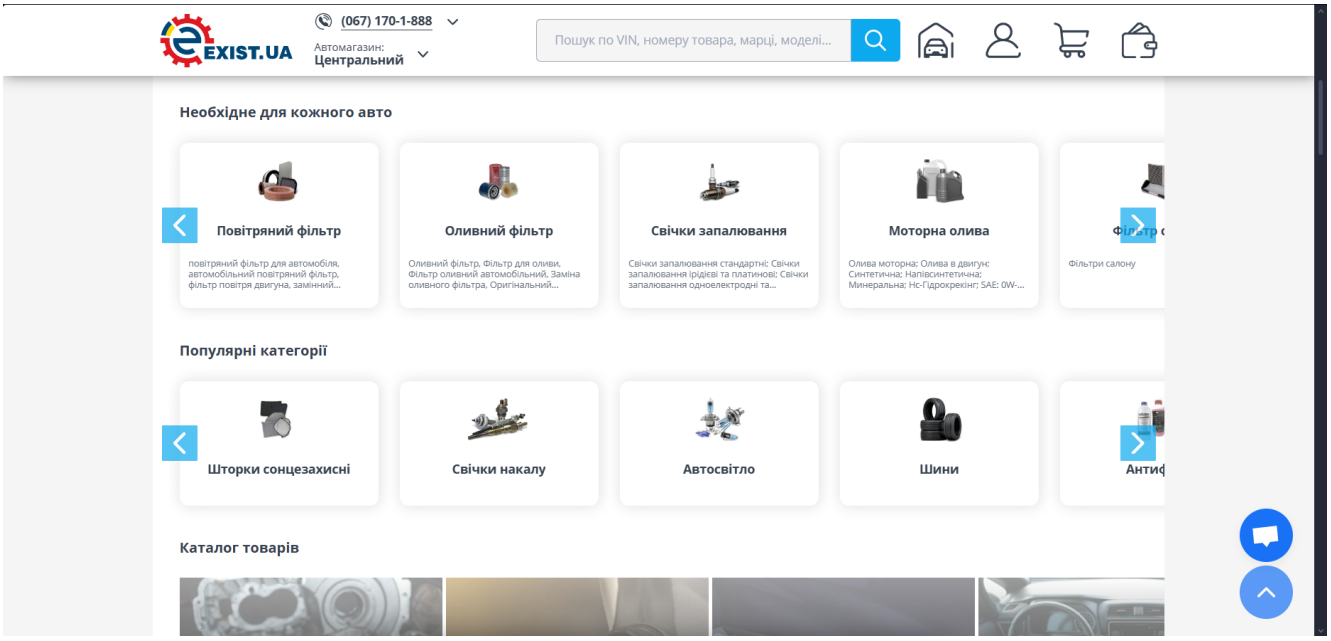


Рисунок 1.1 - Зовнішній вигляд веб-сайту Exist.ua

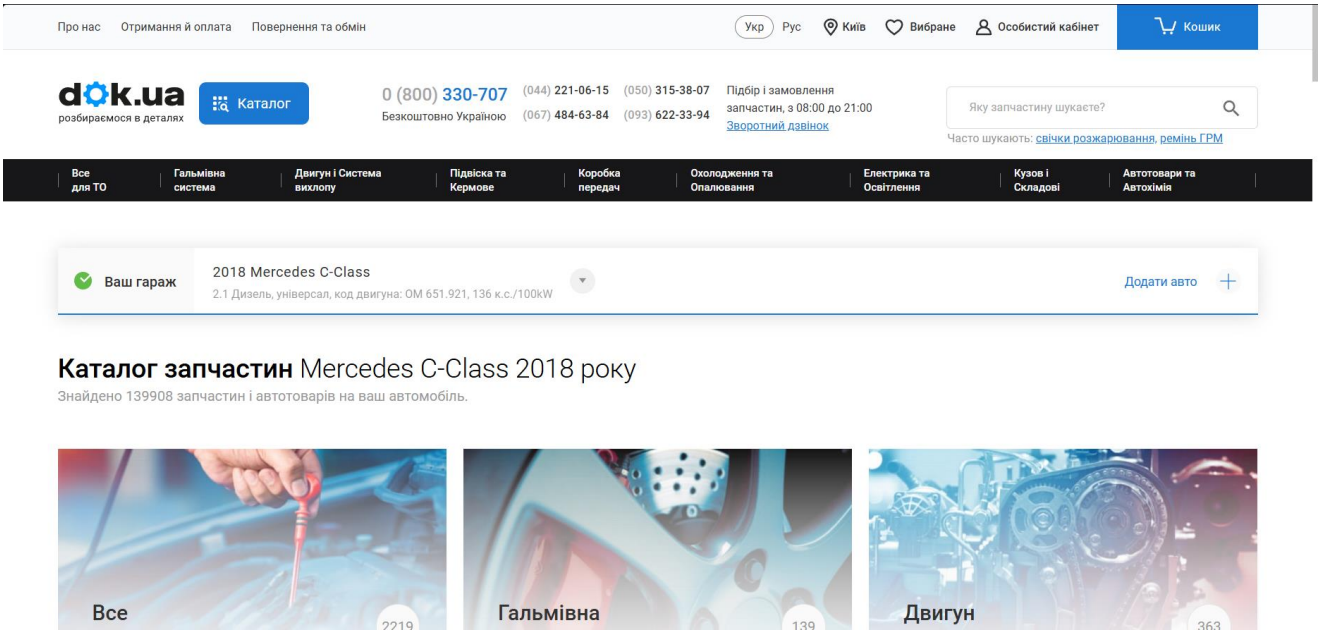


Рисунок 1.2 - Зовнішній вигляд веб-сайту Dok.ua

Ще одним помітним гравцем є ***Autoklad.ua*** (рис. 1.3) український інтернет-магазин запчастин, який пропонує зручний підбір деталей за VIN-кодом автомобіля та має власну бонусну систему для лояльних клієнтів. Крім запчастин, тут можна

знайти автохімію та оливи. Однак, асортимент Autoklad.ua може бути дещо меншим у порівнянні з гігантами ринку, а фізична присутність обмежена лише Києвом, що створює певні незручності для клієнтів з регіонів, які потребують особистого обслуговування.

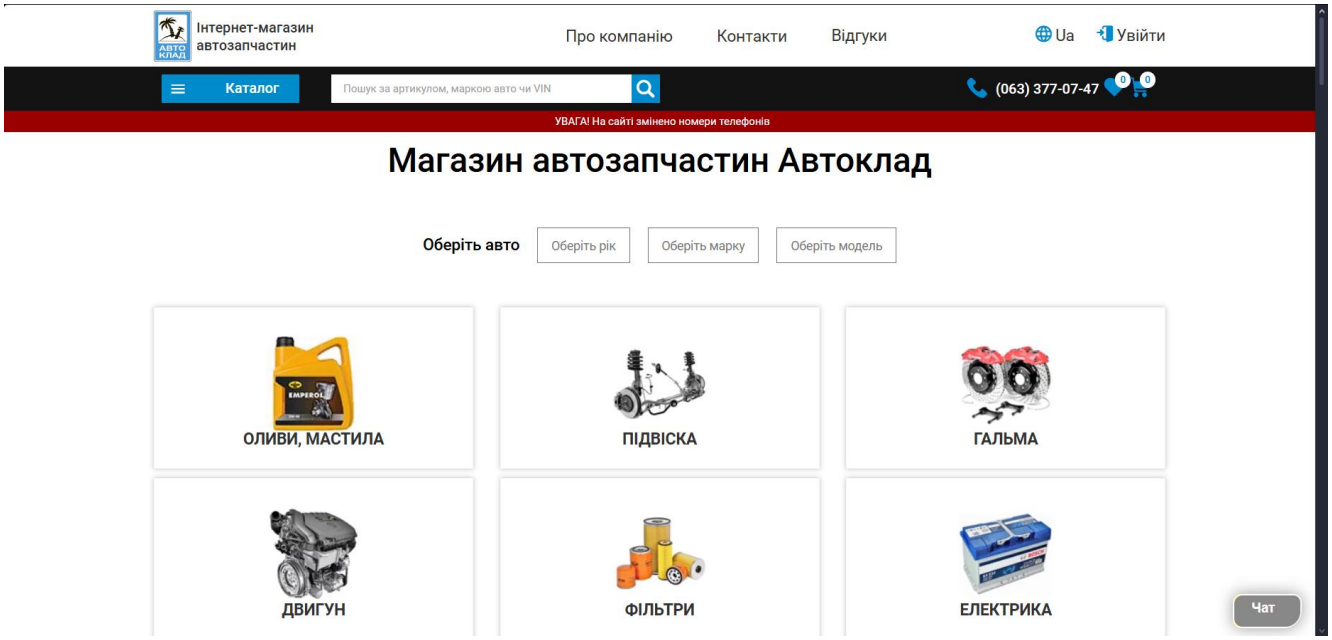


Рисунок 1.3 - Зовнішній вигляд веб-сайту Autoklad.ua

Міжнародні платформи. Мають доступ до глобального ринку запчастин, що особливо корисно для рідкісних або американських/європейських автомобілів. Недоліки – можливі складнощі з доставкою та митним оформленням для українських користувачів, мовні бар'єри.

На глобальному ринку автозапчастин значне місце посідають такі гравці, як ***RockAuto.com*** та ***eBay.com***, кожен з яких має свої унікальні пропозиції, переваги та недоліки, а також ймовірне використання штучного інтелекту у своїй діяльності.

RockAuto.com (рис. 1.4) є великим американським інтернет-магазином, відомим своїм величезним каталогом автозапчастин, особливо для автомобілів американського ринку. Ця платформа часто пропонує конкурентні ціни на самі деталі та здійснює доставку по всьому світу. Хоча RockAuto.com і приваблює покупців асортиментом та цінами, значними недоліками для міжнародних клієнтів є дуже висока вартість та тривалі терміни міжнародної доставки. Крім того, можливі додаткові митні збори, а також складнощі з поверненням товару чи гарантійним обслуговуванням. Ймовірно, RockAuto.com використовує штучний

інтелект для оптимізації свого каталогу, керування запасами та, можливо, для персоналізації пошуку товарів або пропозицій, виходячи з історії переглядів користувача.

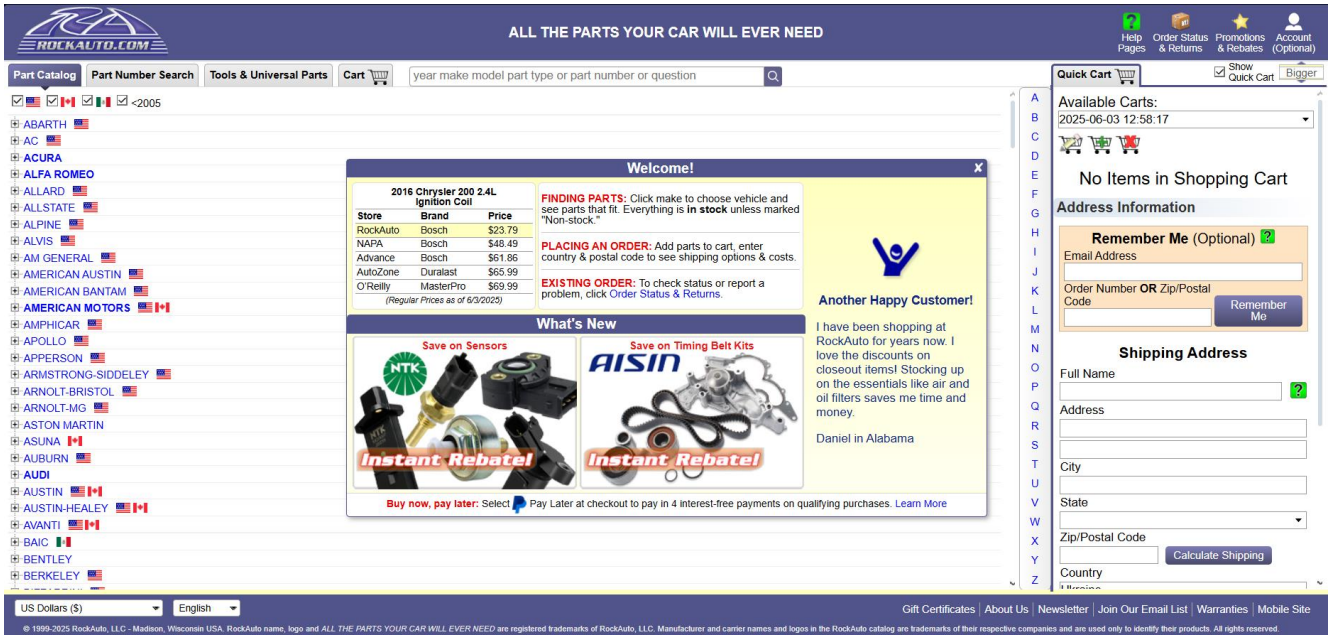


Рисунок 1.4 - Зовнішній вигляд веб-сайту RockAuto.com

eBay.com це глобальний онлайн-маркетплейс (рис. 1.5), де можна знайти практично будь-які автозапчастини, як нові, так і вживані, від продавців з усього світу, включно з рідкісними екземплярами.

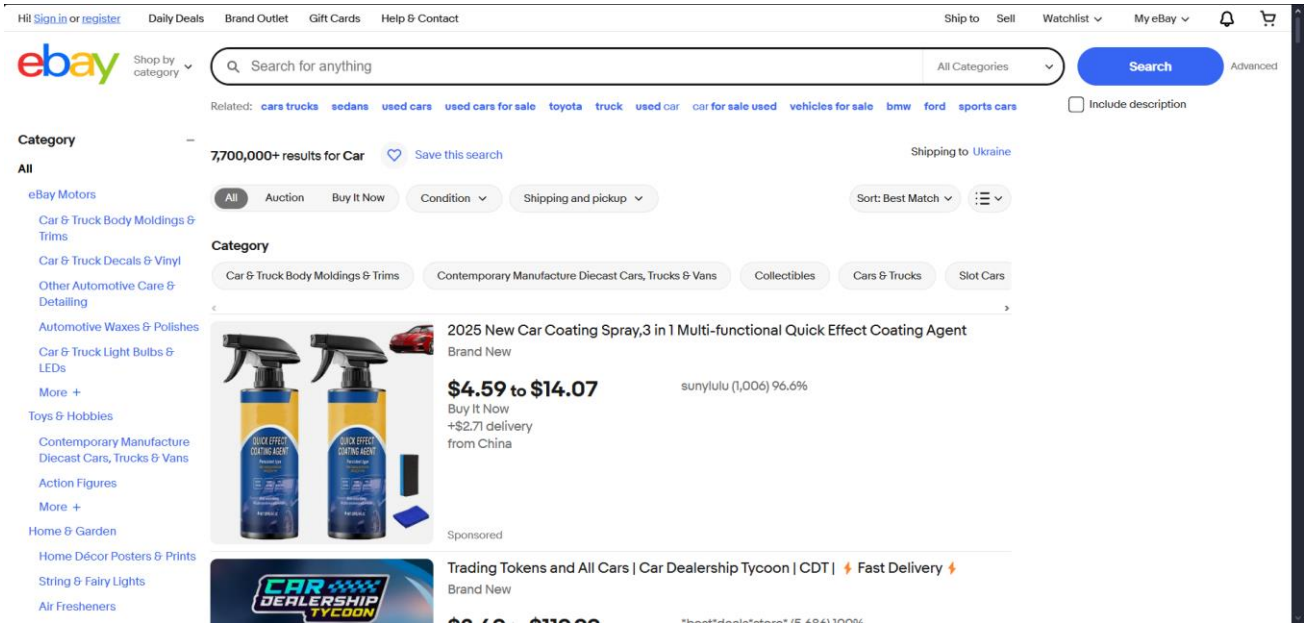


Рисунок 1.5 - Зовнішній вигляд веб-сайт ebay.com

Така різноманітність є значною перевагою. Однак, з цим широким вибором пов'язані і суттєві недоліки: існує високий ризик придбати підробку або неякісний товар, особливо від неперевірених продавців. Також спостерігається велика варіативність у вартості та термінах доставки, а процедури повернення можуть бути складними і залежать від конкретного продавця. Без сумніву, eBay активно застосовує штучний інтелект у своїй роботі. ШІ використовується для багатьох аспектів, включаючи персоналізовані рекомендації товарів (на основі історії пошуку та покупок), оптимізацію пошукових запитів, виявлення шахрайства та підробок, а також для сегментації ринку та аналізу поведінки продавців і покупців для покращення користувацького досвіду та безпеки угод.

Агрегатори та дошки оголошень. Дозволяють знаходити пропозиції від різних продавців, включаючи приватних осіб та невеликі магазини. Переваги – можливість знайти вживані деталі або вигідніші ціни. Недоліки – вищий ризик придбання неякісного товару, відсутність гарантій. Вебсайт OLX зображено на рис. 1.6.

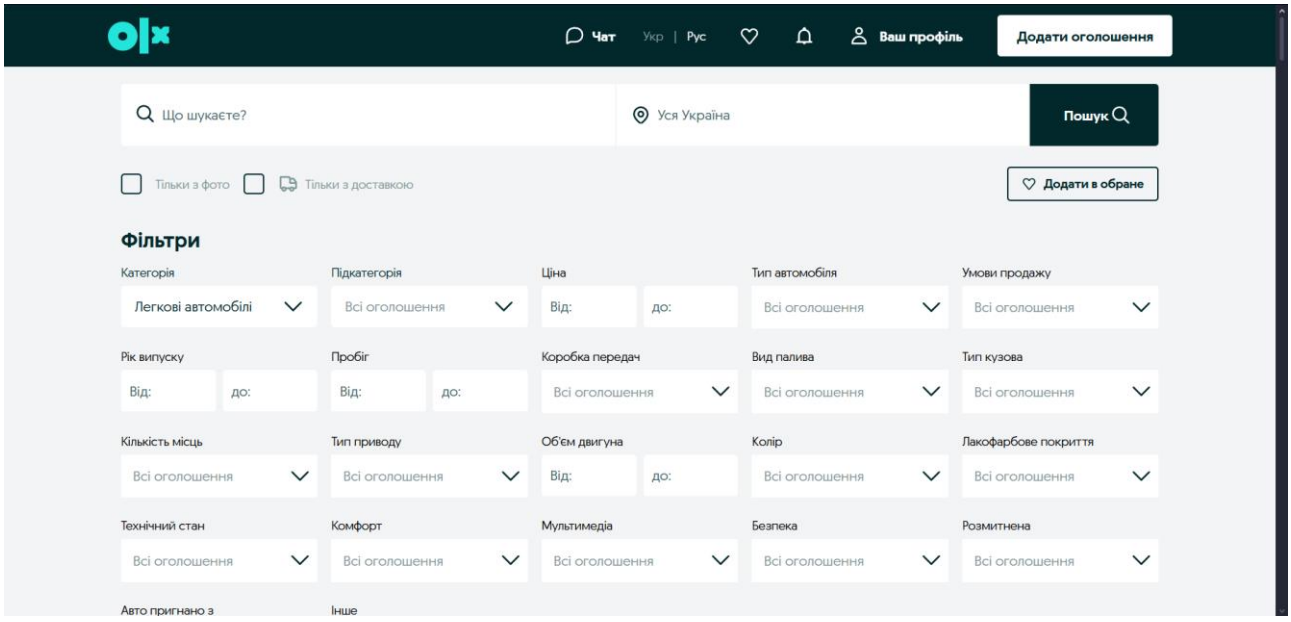


Рисунок 1.6 - Зовнішній вигляд веб-сайт olx.ua

Більшість існуючих систем покладаються на традиційні методи пошуку за ключовими словами або навігацію по структурованих каталогах. Хоча деякі з них мають досить розширені фільтри, справді "інтелектуального" підходу, який би

аналізував нечіткі запити, пропонував альтернативи на основі непрямих ознак або надавав комплексну допоміжну інформацію (як-от зображення автомобіля для візуальної ідентифікації сумісності) в рамках одного простого інтерфейсу, часто бракує.

1.4 Потреба в розробці спеціалізованого веб-додатку

Аналіз предметної області та існуючих рішень показує наявність ніші для вебдодатку, який би спростив та оптимізував процес пошуку автозапчастин, особливо для користувачів, які не є професійними автомеханіками. Ключовими аспектами такої системи мають стати:

- *Гнучкість пошуку* можливість шукати як за точним ID деталі, так і за описовими параметрами автомобіля та назвою деталі.
- *Інформативність результатів* надання не лише списку деталей, а й додаткової інформації, такої як зображення автомобіля, до якого підходить деталь, для кращої візуальної верифікації.
- *Агрегація посилань* збір та надання прямих посилань на сторінки пошуку або товари на популярних українських та міжнародних ресурсах.
- *Зручний та інтуїтивний інтерфейс* мінімалістичний дизайн, підтримка кількох мов, теми оформлення, історія пошуків для покращення користувацького досвіду.
- *Використання "AI" підходів (в рамках клієнтської реалізації)* хоча повноцінний ШІ виходить за рамки клієнтського додатку, "інтелектуальність" може бути реалізована через продуману логіку обробки запитів, фільтрацію результатів (як у випадку з Pexels API для зображень) та надання максимально релевантних посилань.

Розробка такого вебдодатку, як "AI Car Parts Finder", є актуальним завданням, що відповідає потребам сучасних автовласників.

Ринок автомобільних запчастин є обширним і динамічно зростаючим, проте його складність часто створює значні труднощі для кінцевих користувачів у процесі пошуку та підбору необхідних деталей. Попри широкий функціонал, доступний на

існуючих онлайн-платформах, багато з них страждають від перевантажених інтерфейсів або недостатньо гнучких пошукових механізмів, що не завжди дозволяє користувачам швидко й точно знаходити потрібні запчастини. Ця ситуація підкреслює виражену потребу у створенні вебсистеми нового покоління, яка б значно спрощувала процес пошуку, надавала релевантну додаткову інформацію, таку як зображення автомобіля для візуальної верифікації, і агрегувала посилання на різні джерела для придбання. Саме на вирішення цих проблем спрямований проєкт "AI Car Parts Finder", метою якого є розробка інтуїтивно зрозумілого, багатомовного вебдодатку з гнучкими пошуковими можливостями та інтеграцією із зовнішніми сервісами.

Щодо запровадження штучного інтелекту, можна стверджувати, що багато сучасних онлайн-платформ для продажу автозапчастин вже використовують або починають інтегрувати елементи штучного інтелекту у свої робочі процеси. Великі гравці, такі як *Exist.ua, Dok.ua, Autoklad.ua*, ймовірно, застосовують ШІ для оптимізації управління запасами, аналізу попиту, прогнозування продажів і базової персоналізації рекомендацій. Глобальні маркетплейси, як-от *RockAuto.com* та особливо *eBay.com*, безсумнівно, широко застосовують штучний інтелект. Вони використовують його для просунутих рекомендаційних систем, які аналізують поведінку користувачів та властивості товарів для надання високорелевантних пропозицій. Також ШІ допомагає їм у покращенні пошукових алгоритмів, виявленні шахрайства, автоматизації клієнтської підтримки через чат-ботів та оптимізації ціноутворення. Це свідчить про те, що чим більшим є масштаб і складність платформи, тим глибше ШІ інтегрується в її функціонування для підвищення ефективності та покращення користувацького досвіду.

2 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ ДЛЯ ПОШУКУ АВТОМОБІЛЬНИХ ЗАПЧАСТИН

2.1 Формулювання вимог до системи

На основі аналізу предметної області та потреб потенційних користувачів, були сформульовані наступні функціональні та нефункціональні вимоги до вебдодатку "AI Car Parts Finder".

Таблиця 2.1 Функціональні вимоги до проектування веб-ресурсу

Код вимоги	***Назва вимоги***
1	**Багатомовний інтерфейс.** Система повинна підтримувати щонайменше дві мови інтерфейсу – українську та англійську, з можливістю легкого перемикання між ними.
2	**Пошук запчастин:** - Можливість пошуку за ідентифікаційним номером деталі (Part ID). - Можливість пошуку за параметрами автомобіля (марка, модель, рік випуску) та назвою деталі.
3	**Відображення результатів пошуку:** - Коротка назва/опис знайденої деталі (або підтвердження пошуку за ID). - Зображення автомобіля, для якого підходить деталь (отримане через Pexels API). - Мінімум 5 прямих посилань на сторінки пошуку або товари в інтернет-магазинах (3 українських, 2 міжнародних).
4	**Взаємодія з результатами пошуку:** - Можливість "підтвердити пошук", що повертає користувача на головний екран з пропозицією нового пошуку. - Можливість "спробувати інші варіанти" (Deny Search), що генерує посилання на іншу групу українських магазинів для того ж запиту .
5	**Історія пошуків.** Система повинна зберігати останні 5 унікальних пошукових запитів користувача (в localStorage) та відображати їх для швидкого повторного пошуку і можливість очистити історію пошуків.
6	**Автозаповнення.** Надання підказок (автозаповнення) для поля "Марка автомобіля".

Код вимоги	***Назва вимоги***
7	**Персоналізація інтерфейсу.** - Можливість перемикання між світлою та темною темами оформлення. - Збереження обраної теми в localStorage.
8	**Навігація.** Кнопка/логотип для повернення на головний екран.
9	**Інформаційні повідомлення:** - Відображення повідомлень про процес пошуку (наприклад, "AI шукає вашу запчастину..."). - Відображення повідомлень про помилки (наприклад, неповні дані для пошуку, відсутність результатів, помилки API).
10	**Зовнішні посилання** Кнопка для переходу до Telegram-бота.

Проаналізуємо нефункціональні вимоги мого застосунку, представлені у табл. 2.2

Таблиця 2.2 Нефункціональні вимоги

Код вимоги	***Назва вимоги***
1	**Продуктивність:** Система повинна швидко реагувати на дії користувача та надавати результати пошуку протягом прийнятного часу (з урахуванням часу відповіді зовнішніх API).
2	**Надійність:** Система повинна стабільно працювати та коректно обробляти помилки.
3	**Зручність використання (Usability):** Інтерфейс має бути інтуїтивно зрозумілим, простим у використанні та не перевантаженим зайвими елементами.
4	**Адаптивність (Responsiveness):** Вебдодаток повинен коректно відображатися та функціонувати на різних розмірах екранів (десктопи, планшети, мобільні пристрої).
5	**Безпека:** Оскільки система є клієнтською та не зберігає критичних даних користувача на сервері (окрім localStorage), основні аспекти безпеки пов'язані з коректною взаємодією з зовнішніми API (використання API ключів).
6	6. **Масштабованість** (в контексті клієнтської частини): Код має бути структурованим та модульним для полегшення подальших модифікацій та додавання нового функціоналу (наприклад, розширення списку магазинів).

2.2 Проєктування архітектури вебдодатку

Вебдодаток "AI Car Parts Finder" реалізовано як односторінковий додаток (Single Page Application - SPA) на стороні клієнта. Вся логіка відображення, взаємодії з користувачем та обробки даних (включаючи запити до зовнішніх API) виконується у браузері користувача за допомогою HTML, CSS та JavaScript.

Основні компоненти архітектури:

1. *Презентаційний шар (Frontend):*
   - HTML (index.html): Визначає базову структуру вебсторінки, контейнери для різних секцій (початковий екран, форма пошуку, результати), модальні вікна та інші елементи інтерфейсу. Містить плейсхолдери для динамічного контенту.
   - CSS (style.css): Відповідає за візуальне оформлення всіх елементів, адаптивність, реалізацію світлої/темної тем та стилізацію динамічно створених елементів (наприклад, картки результатів).
   - JavaScript (script.js): Містить всю клієнтську логіку:
     - Обробка подій користувача (кліки, відправка форм).
     - Маніпуляція DOM (показ/приховування секцій, динамічне створення HTML для результатів пошуку, оновлення тексту).
     - Валідація введених даних.
     - Управління станом (поточна мова, тема, історія пошуків через localStorage).
     - Взаємодія з зовнішніми API (Pexels API для зображень).
     - Логіка "інтелектуального пошуку" (формування пошукових запитів для магазинів, обробка відповіді Pexels).
     - Відображення повідомлень та помилок.
2. *Зовнішні сервіси (API):*
   - Pexels API: Використовується для отримання зображень автомобілів на основі марки, моделі та року.
   - Інтернет-магазини автозапчастин (непряма взаємодія): Система не взаємодіє з API магазинів напряму (через відсутність безкоштовних публічних API для цього), а генерує прямі URL-посилання на сторінки пошуку цих магазинів,

використовуючи досліджені шаблони URL та пошукові запити, сформовані на основі введених користувачем даних.

Відобразимо спрощену схему взаємодії компонентів системи (рис. 1.1). Отримана схема відображає архітектуру проектованого ресурсу, побудовану за класичним трирівневим принципом, з додаванням окремого модуля для машинного навчання.

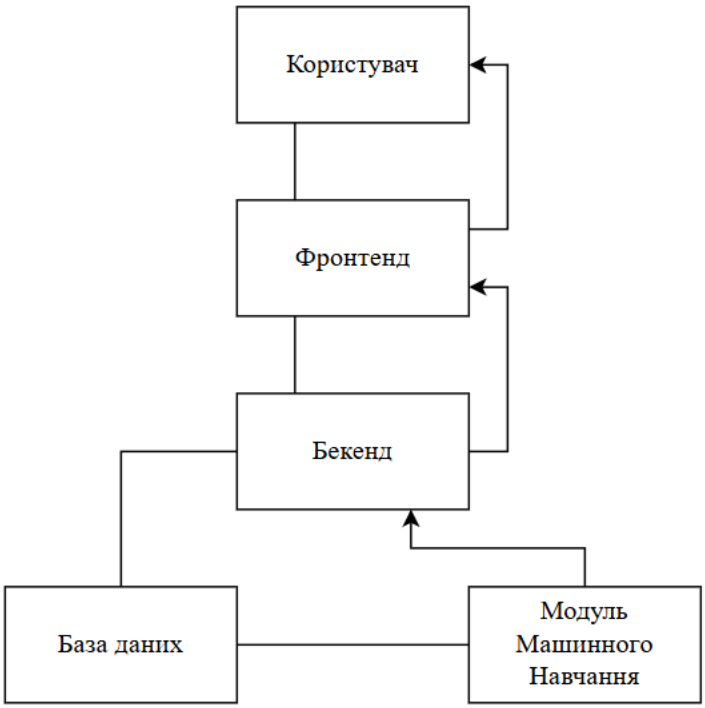


Рисунок 2.1 - Схема взаємодії компонентів проектованої системи

Взаємодія системи починається з *Користувача*, який є ініціатором усіх операцій. Користувач взаємодіє з *Фронтендом* – це є клієнтська частина веб-додатку, тобто те, що користувач бачить і з чим безпосередньо працює (веб-сторінки, інтерфейс пошуку, відображення результатів). Фронтенд відповідає за представлення інформації та збір вхідних даних від користувача.

Фронтенд, у свою чергу, обмінюється даними з *Бекендом* – серверною частиною додатку. Бекенд виконує основну логіку обробки запитів: приймає дані від фронтенду (наприклад, пошукові запити, інформацію про автомобіль), обробляє їх, звертається до інших компонентів системи та готує відповідь для фронтенду.

Бекенд має прямий зв'язок з *Базою даних*. У базі даних зберігається вся необхідна інформація для функціонування ресурсу, зокрема дані про автозапчастини, їхні характеристики, сумісність з моделями авто, а також, ймовірно, інформація про користувачів, історія їхніх запитів чи вподобань. Бекенд отримує звідти дані для формування відповідей на запити користувачів та зберігає нову інформацію.

Особливістю цієї архітектури є наявність окремого *Модуля Машинного Навчання*. Цей модуль підключається до Бекенду, надаючи системі інтелектуальні можливості. Він відповідає за реалізацію алгоритмів штучного інтелекту, зокрема для персоналізованих рекомендацій товарів. Модуль машинного навчання може отримувати дані з Бази даних (наприклад, історію взаємодій користувачів), обробляти їх і надавати Бекенду результати – наприклад, ранжований список рекомендованих запчастин або прогнози щодо сумісності. Бекенд інтегрує ці дані у відповідь для Користувача через Фронтенд.

Таким чином, схема відображає типову клієнт-серверну архітектуру з виділеним шаром для обробки та інтелектуалізації даних за допомогою машинного навчання. Стрілки показують двосторонній зв'язок між Користувачем та Фронтендом, а також між Фронтендом та Бекендом. Бекенд є центральним вузлом, що взаємодіє з Базою даних для зберігання/отримання інформації та з Модулем Машинного Навчання для інтелектуальної обробки та формування персоналізованих пропозицій.

Потік даних при пошуку на сайті запчастин з ШІ. У процесі проектування веб-сайту для пошуку автозапчастин із залученням штучного інтелекту, потік даних під час пошуку розгортається наступним чином. Спочатку користувач ініціює дію, вводячи необхідні дані у форму пошуку та натискаючи кнопку "Search". На цьому етапі JavaScript відіграє ключову роль, отримуючи введені дані та здійснюючи їхню первинну валідацію.

Якщо введені дані відповідають встановленим вимогам валідації, система одночасно виконує кілька операцій. По-перше, надсилається запит до зовнішнього сервісу, такого як Pexels API, для отримання зображення автомобіля. Цей запит

формується на основі введених користувачем даних про марку, модель та рік випуску автомобіля, що дозволяє візуалізувати транспортний засіб для кращого підтвердження правильності вибору. По-друге, паралельно з цим, JavaScript генерує список прямих посилань на відповідні інтернет-магазини автозапчастин, які потенційно можуть мати потрібний товар.

Після успішного отримання відповіді від Pexels API, або у випадку відсутності зображення – використання заздалегідь підготовленого зображення-заглушки, JavaScript динамічно формує HTML-код. Цей код відповідає за візуальне відображення результатів пошуку для користувача. Сформований HTML-код включає отримане зображення автомобіля, відповідний опис, а також згенеровані посилання на інтернет-магазини. Нарешті, оновлена секція результатів відображається користувачеві, надаючи йому всю необхідну інформацію для подальшого вибору та придбання запчастин. Важливо зазначити, що хоча на даному етапі ШІ не вказано як прямий учасник цього фронтендового потоку, його функціонал (наприклад, персоналізовані рекомендації, оптимізація пошукових запитів або фільтрація результатів) буде інтегрований на бекенді, впливаючи на якість і релевантність згенерованих посилань та описів, перш ніж вони будуть відображені користувачеві.

2.3 Проєктування користувацького інтерфейсу

Дизайн інтерфейсу розроблявся з акцентом на простоту, інтуїтивність та зручність використання. При цьому були застосовані основні принципи дизайну: прагнення до мінімалізму, уникаючи зайвих елементів та візуального шуму, аби користувач міг повністю зосередитися на головній меті – пошуку необхідних запчастин. Важливе значення надавалося чіткій структурі, що передбачає логічне групування елементів та послідовне представлення інформації. Для виділення ключових елементів використовувалася візуальна ієрархія, досягнута за допомогою розмірів шрифтів, кольорів та відступів. Також було забезпечено адаптивність для коректного відображення на різноманітних пристроях. Система передбачає надання користувачеві зворотного зв'язку щодо поточного стану, інформуючи про процеси

завантаження або можливі помилки. Естетична складова реалізована через використання сучасної колірної схеми, що поєднує сірі тони з помаранчевими акцентами, а також відповідних шрифтів (рис. 2.2).

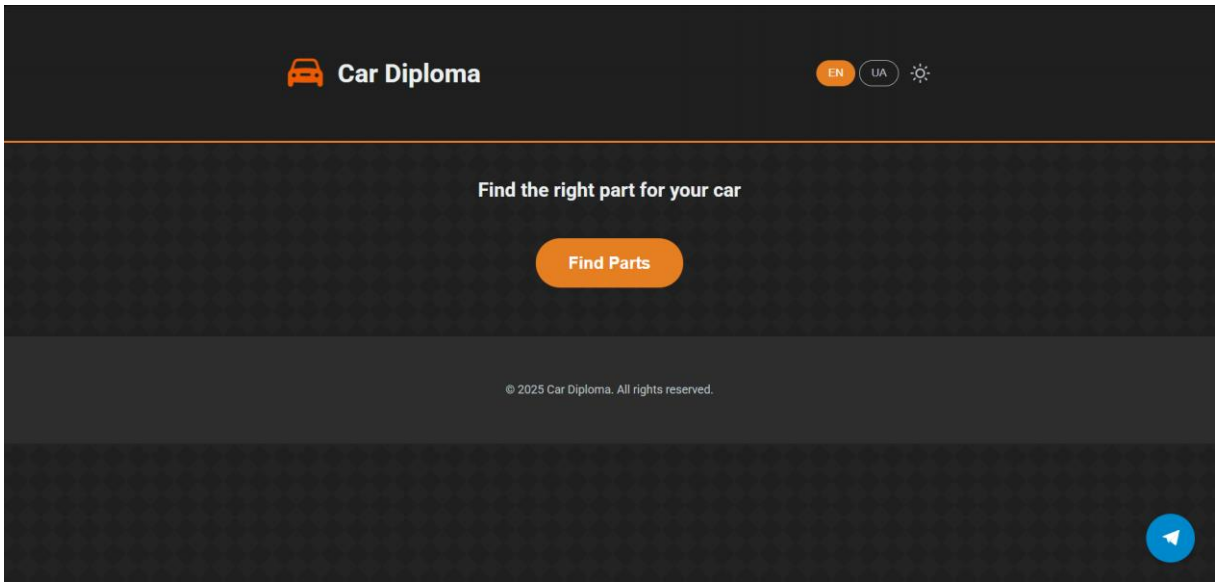


Рисунок 2.2 - Зовнішній вигляд проектованого веб-ресурсу.m

Серед ключових екранів та елементів системи, першим є початковий екран, названий "initial-view". Він містить заголовок, короткий опис функціоналу та головну кнопку "Знайти Запчастини".

Екран пошуку (рис. 2.3), позначений як "search-view", розроблений для інтуїтивного введення даних користувачем. На ньому розміщена форма, яка містить поля для введення ідентифікатора деталі або, як альтернатива, комбінації даних про марку, модель та рік випуску автомобіля, а також назву необхідної деталі. Для покращення зручності та прискорення введення інформації, у полі "Марка автомобіля" реалізовано функцію автозаповнення за допомогою елемента `<datalist>`. Крім того, на цьому екрані відображається історія останніх пошуків, що дозволяє користувачеві швидко повторити попередні запити або ж очистити історію за потреби, що забезпечує гнучкість у навігації.

Екран результатів (рис. 2.4), позначений як "results-view", призначений для візуалізації знайдених даних у зручному для сприйняття форматі. На цьому екрані результати пошуку відображаються у вигляді інтерактивних карток. Кожна така картка містить зображення відповідного автомобіля, назву або детальний опис

знайденої запчастини, актуальну інформацію про ціни, а також кнопки-посилання, що ведуть безпосередньо до відповідних інтернет-магазинів, де можна здійснити покупку.

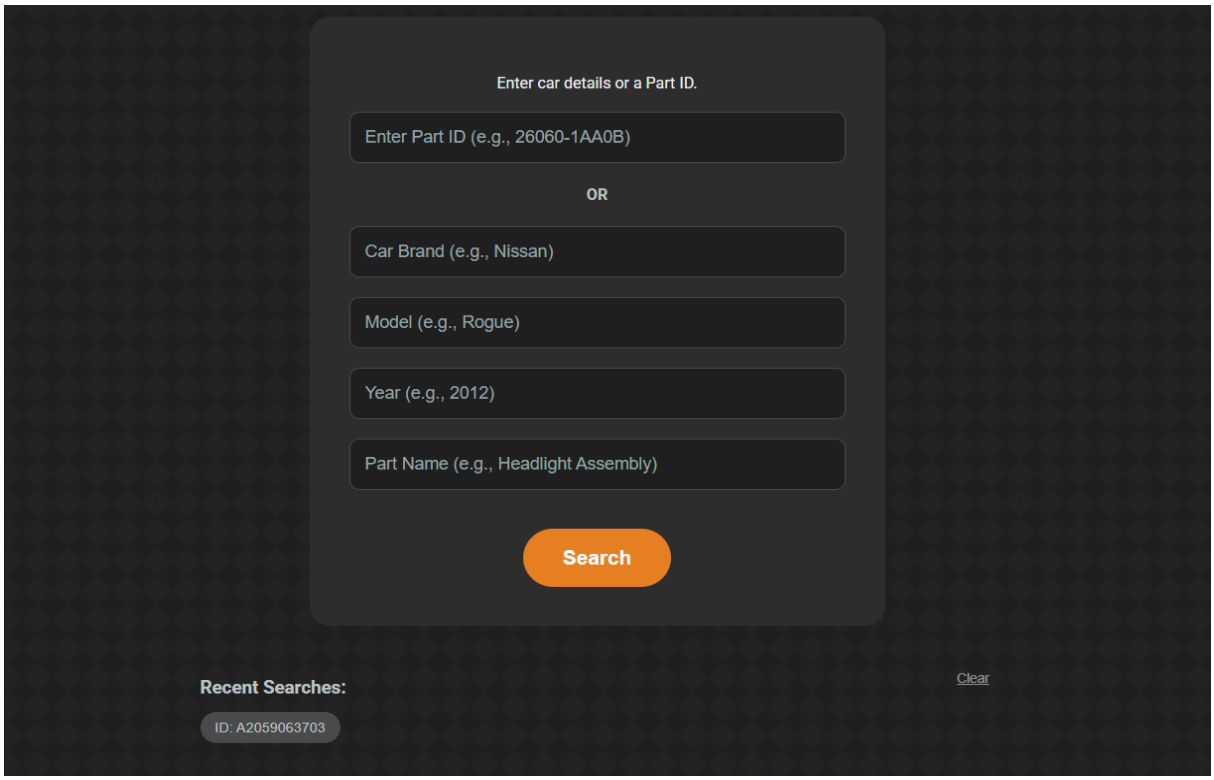


Рисунок 2.3 - Екран пошуку інформації на веб-сайті

Для подальшої взаємодії користувачу доступні дві функціональні кнопки: "Підтвердити пошук", яка повертає на головний екран, та "Шукати інші варіанти", що дозволяє ініціювати новий пошук або змінити параметри поточного.

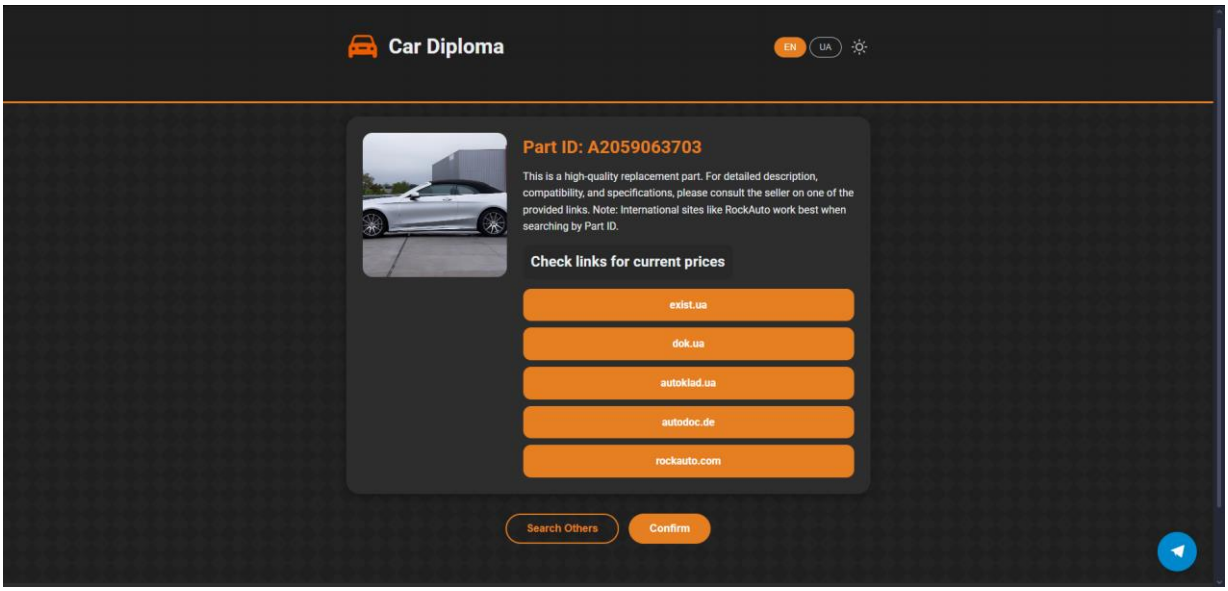


Рисунок 2.4 - Екран результатів пошуку на веб-ресурсі

Інтерфейс сайту також включає стандартні, але функціональні елементи для зручності користувача. У верхній частині розташований **Заголовок (Header)**, що містить логотип та назву сайту. Ці елементи є клікабельними, дозволяючи користувачеві швидко повернутися на головний екран. Крім того, заголовок оснащений перемикачем мов для багатомовної підтримки та перемикачем теми, що дає можливість обрати світлу або темну схему відображення інтерфейсу.

У нижній частині веб-сторінки розміщено **Підвал (Footer)**, який традиційно містить інформацію про копірайт. (рис. 2.5). Для покращення взаємодії з користувачем та надання зворотного зв'язку передбачено *Модальне вікно* (рис. 2.6). Воно використовується для відображення системних повідомлень, таких як індикатори завантаження або сповіщення про помилки, забезпечуючи миттєву інформацію про стан системи.

Рисунок 2.5 - Заголовок та логотип проектованого ресурсу

Додатково, на сторінці інтегрована *плаваюча кнопка Telegram.* Ця кнопка забезпечує швидкий перехід до Telegram-бота, розширюючи можливості взаємодії та підтримки користувачів. Візуальне оформлення сайту включає фонове зображення з текстурою, а також спеціально розроблений логотип, який наочно демонструє автомобільну тематику вебсайту.

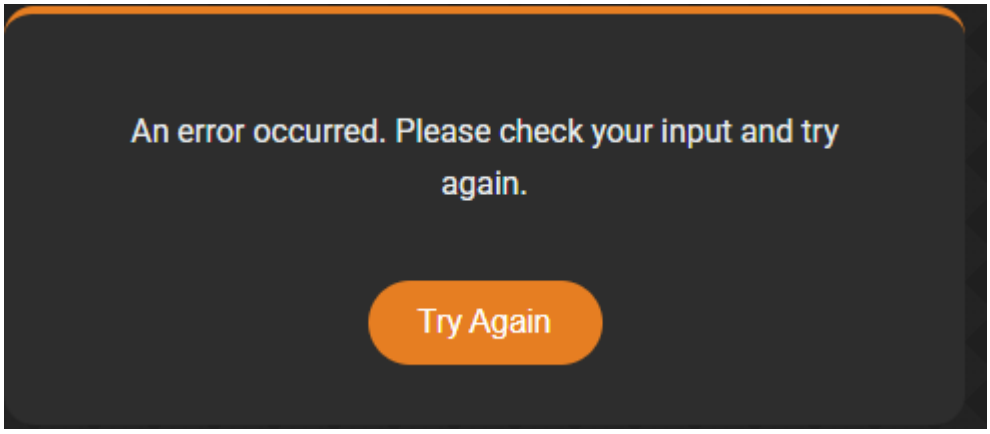


Рисунок 2.6 - Вигляд модального вікна з повідомленняи про помилку

2.4 Вибір технологій для програмної реалізації

Для реалізації клієнтської частини вебдодатку було обрано наступний стек технологій, враховуючи вимоги до інтерактивності, користувацького досвіду та простоти розробки без серверної частини:

Таблиця 2.3 Технології, використані для розробки вебдодатку

Категорія	Технологія / Мова	Призначення
Структура	HTML5	Створення семантичної структури вебсторінок, розмітка контенту та елементів інтерфейсу.
Стилізація	CSS3	Візуальне оформлення, адаптивний дизайн, реалізація світлої/темної тем, стилізація динамічних елементів.
Логіка клієнта	JavaScript (ES6+)	Реалізація всієї інтерактивності, DOM-маніпуляції, валідація форм, обробка подій, робота з localStorage.
API Зображень	Pexels API	Отримання фотографій автомобілів для візуалізації результатів пошуку.
Шрифти	Google Fonts (Roboto)	Забезпечення сучасного та читабельного вигляду тексту.
Контроль версій	Git (концептуально)	Управління версіями коду під час розробки (хоча не було прямої реалізації в нашому діалозі).

Для розробки веб-ресурсу був обраний стек технологій, що включає HTML5, CSS3 та JavaScript. Цей вибір обґрунтований тим, що це є стандартний, широко підтримуваний та гнучкий набір для створення сучасних вебінтерфейсів, який дозволяє реалізувати весь необхідний функціонал безпосередньо на стороні клієнта. Додатково, для отримання високоякісних зображень автомобілів інтегровано Pexels API, що надає безкоштовний доступ до великої бібліотеки фотографій, ідеально відповідних завданню.

Для естетичного оформлення інтерфейсу та зручності читання тексту використовується Google Fonts, який забезпечує простий спосіб підключення

якісних вебшрифтів. З метою зберігання невеликих обсягів даних, таких як налаштування або історія пошуків, безпосередньо у браузері користувача, задіюється LocalStorage, що є зручним механізмом для цієї мети. Сукупність цих технологій дає змогу створити повноцінний клієнтський додаток, який не вимагає розгортання окремої серверної інфраструктури для реалізації основного функціоналу, що повністю відповідає початковим вимогам проєкту.

2.5. Обґрунтування вибору зовнішніх API та підходів до інтеграції

Для реалізації функціоналу, що виходить за межі стандартних можливостей клієнтського додатку, було прийнято рішення про інтеграцію з зовнішніми сервісами через програмні інтерфейси додатків (API). Вибір конкретних API та підходів до їх використання базувався на критеріях доступності, функціональності, простоти інтеграції та відповідності вимогам проекту.

2.5.1. Вибір API для отримання зображень

Для забезпечення візуального супроводу результатів пошуку, що значно покращує користувацький досвід завдяки можливості візуально підтвердити правильність вибору деталі, було поставлено завдання відображати зображення автомобіля, якому відповідає знайдена запчастина. Для вирішення цього завдання було проаналізовано декілька підходів. Першим варіантом було створення власної бази даних зображень, проте цей підхід потребує значних ресурсів для збору, зберігання та обробки великого масиву фотографій, що виходить за рамки даної роботи. Другим підходом розглядалося використання API пошукових систем, таких як Google Images, однак більшість пошукових гігантів не надають безкоштовного та вільного API для пошуку зображень сторонніми розробниками. Зрештою, найбільш оптимальним варіантом для цього проєкту визнано використання API фотостоків.

Після аналізу доступних сервісів, вибір зупинився на **Pexels API**. Цей вибір був обґрунтований кількома ключовими перевагами. Pexels надає щедрий безкоштовний план доступу для розробників, який повністю покриває потреби проєкту. Сервіс містить велику бібліотеку високоякісних фотографій, що дозволяє

знаходити естетичні зображення автомобілів. Крім того, API Pexels вирізняється простотою та зрозумілістю, базуючись на стандартних REST-принципах та маючи чітку документацію, що значно спростило його інтеграцію в JavaScript-додаток за допомогою стандартного методу fetch(). Важливою перевагою є також гнучкість пошуку, оскільки API дозволяє використовувати текстові запити та фільтрувати результати, що стало основою для "інтелектуального" відбору релевантних зображень. Як альтернатива, розглядався Unsplash API зі схожим функціоналом, проте Pexels було обрано через більш просту процедуру отримання ключа та зрозумілі ліміти використання на момент розробки.

2.5.2. Підхід до інтеграції з інтернет-магазинами

Однією з ключових вимог до системи було забезпечення користувача можливістю не лише знайти необхідну деталь, а й одразу перейти до її придбання. Ідеальний сценарій передбачав би пряме отримання актуальних цін та інформації про наявність товару безпосередньо з веб-сайтів продавців. Проте, аналіз виявив значні технічні перешкоди на цьому шляху. По-перше, переважна більшість як українських, так і міжнародних інтернет-магазинів автозапчастин не надають відкритих та безкоштовних API для отримання даних про свої товари сторонніми сервісами. По-друге, прямі запити до сторінок інтернет-магазинів з клієнтського JavaScript-коду блокуються політикою безпеки сучасних браузерів, відомою як CORS (Cross-Origin Resource Sharing). Це робить неможливим автоматичний збір даних (скрапінг) безпосередньо у браузері користувача.

З огляду на ці обмеження, було обрано прагматичне та ефективне рішення: генерація прямих посилань на сторінки пошуку кожного конкретного інтернет-магазину. Цей підхід має низку переваг. Він забезпечує надійність, оскільки, на відміну від скрапінгу, який може перестати працювати при найменшій зміні дизайну сайту-джерела, структура пошукових URL-адрес є значно стабільнішою. Також гарантується актуальність інформації: користувач, переходячи за посиланням, потрапляє безпосередньо на сторінку магазину з найактуальнішими даними про ціни та наявність на момент запиту. Крім того, це рішення відзначається простотою реалізації, адже воно не потребує серверної інфраструктури та

реалізується на стороні клієнта шляхом дослідження та використання стандартизованих шаблонів пошукових запитів для кожного сайту. Таким чином, обраний підхід дозволяє надати користувачеві максимальну цінність в рамках існуючих технічних обмежень, ефективно спрямовуючи його до першоджерела необхідної інформації.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБ-ДОДАТКУ "AI CAR PARTS FINDER"

У цьому розділі докладно описано програмну реалізацію ключових функцій та модулів веб-додатку "AI Car Parts Finder", яка ґрунтується на обраних технологіях HTML, CSS та JavaScript. Тут розглядаються всі аспекти створення інтерактивного та функціонального інтерфейсу, що забезпечує користувачеві ефективний пошук та отримання рекомендацій щодо автозапчастин. Це включає розробку візуальних елементів, механізмів введення даних та відображення результатів. Особлива увага приділяється детальному розгляду структури інтерфейсу, починаючи від загального дизайну, що базується на принципах мінімалізму, чіткої структури та адаптивності, і до конкретних елементів, таких як заголовок, підвал та модальні вікна. Крім того, висвітлюється обґрунтування вибору технологій для клієнтської частини, зокрема інтеграції з Pexels API для отримання зображень автомобілів та методу генерації прямих посилань на інтернет-магазини автозапчастин, зважаючи на технічні обмеження щодо отримання актуальних цін та наявності. Таким чином, розділ надає повне уявлення про архітектурні та технічні рішення, що лежать в основі функціонування веб-додатку.

3.1. Структура проєкту та файлова організація веб-ресурсу

Однією з ключових вимог до системи було забезпечення користувача можливістю не лише знайти необхідну деталь, а й одразу перейти до її придбання. Ідеальний сценарій передбачав би пряме отримання актуальних цін та інформації про наявність товару безпосередньо з веб-сайтів продавців. Проте, аналіз виявив значні технічні перешкоди на цьому шляху. По-перше, переважна більшість як українських, так і міжнародних інтернет-магазинів автозапчастин не надають відкритих та безкоштовних API для отримання даних про свої товари сторонніми сервісами. По-друге, прямі запити до сторінок інтернет-магазинів з клієнтського JavaScript-коду блокуються політикою безпеки сучасних браузерів, відомою як

CORS (Cross-Origin Resource Sharing). Це робить неможливим автоматичний збір даних (скрапінг) безпосередньо у браузері користувача.

З огляду на ці обмеження, було обрано прагматичне та ефективне рішення: генерація прямих посилань на сторінки пошуку кожного конкретного інтернет-магазину. Цей підхід має низку переваг. Він забезпечує надійність, оскільки, на відміну від скрапінгу, який може перестати працювати при найменшій зміні дизайну сайту-джерела, структура пошукових URL-адрес є значно стабільнішою. Також гарантується актуальність інформації: користувач, переходячи за посиланням, потрапляє безпосередньо на сторінку магазину з найактуальнішими даними про ціни та наявність на момент запиту. Крім того, це рішення відзначається простотою реалізації, адже воно не потребує серверної інфраструктури та реалізується на стороні клієнта шляхом дослідження та використання стандартизованих шаблонів пошукових запитів для кожного сайту. Таким чином, обраний підхід дозволяє надати користувачеві максимальну цінність в рамках існуючих технічних обмежень, ефективно спрямовуючи його до першоджерела необхідної інформації.

3.2 Реалізація основних функціональних можливостей

При завантаженні сторінки відбувається спрацьовування події DOMContentLoaded, яка активує функцію init(). Ця функція виконує кілька ключових завдань, що забезпечують початкове налаштування веб-додатку. Вона відповідає за отримання посилань на всі важливі DOM-елементи, ідентифікуючи їх за їхніми id. Крім того, функція init() заповнює список <datalist>, який використовуватиметься для автозаповнення марок автомобілів, що значно покращує зручність введення даних для користувача. Вона також встановлює початкові обробники подій для основних кнопок інтерфейсу, забезпечуючи функціональність для переходу на сторінку пошуку, перемикання мов, зміни теми оформлення, очищення історії пошуків та повернення на головну сторінку. Функція init() також застосовує збережені раніше налаштування теми та мови, а також налаштовує початкову видимість секцій сторінки, роблячи видимим початковий

екран і приховуючи інші секції до моменту їхньої активації. Для виконання цих операцій з DOM-елементами використовуються стандартні команди маніпуляції DOM. Для виконання DOM елементів використовуються такі команди:

```
const homeLink = document.getElementById('home-link');
const findPartsBtn = document.getElementById('find-parts-btn');
const initialView = document.getElementById('initial-view');
```

Детально опишемо програмну реалізацію функціональних можливостей створеного вебдодатку. Тут розглядається ініціалізація головного додатку, яка відбувається при завантаженні сторінки через спрацьовування події DOMContentLoaded, що запускає функцію init(). Ця функція виконує ключові налаштування: вона отримує посилання на основні DOM-елементи за їхніми ідентифікаторами, заповнює список <datalist> для автозаповнення марок автомобілів, а також встановлює початкові обробники подій для кнопок, які відповідають за навігацію, зміну мови та теми, очищення історії та повернення на головну сторінку. Крім того, init() застосовує збережені налаштування теми та мови, а також налаштовує початкову видимість секцій сторінки, роблячи видимим лише початковий екран.

Далі детально розглянуто функції, що керують інтерфейсом. Функція setLanguage(lang) відповідає за зміну мови інтерфейсу, оновлюючи глобальну змінну currentLang, встановлюючи атрибут lang для тегу <html>, оновлюючи текстовий вміст елементів з атрибутом data-lang-key з об'єкта translations та стилізуючи активну кнопку мови. Функція toggleTheme() перемикає клас light-mode на елементі <body>, зберігаючи вибір у localStorage та оновлюючи іконку перемикача теми. Функція applyInitialTheme() завантажує збережену тему при запуску додатку.

Обробка пошукового запиту відбувається через обробник події submit для форми searchForm. Він запобігає стандартній відправці форми, зчитує та очищує значення з полів вводу (ID деталі, марка, модель, рік, назва деталі). Система перевіряє валідність даних, переконавшись, що введено або ID деталі, або повний набір параметрів автомобіля з назвою деталі. Якщо дані валідні, запит додається до

історії останніх пошуків, відображається модальне вікно завантаження, після чого викликається асинхронна функція performSearch(query). У разі невалідних даних користувачеві відображається модальне вікно з повідомленням про помилку.

Виконання "інтелектуального" пошуку здійснюється асинхронною функцією performSearch(query, deny = false). Ця функція викликає findCarImage() для отримання URL зображення автомобіля з Pexels API та generateSearchLinks() для формування списку прямих посилань на інтернет-магазини. Параметр deny використовується для ротації українських магазинів при натисканні кнопки "Шукати інші варіанти". Якщо посилання не були згенеровані, система показує повідомлення "No results found" і завершує роботу. Після цього ховається модальне вікно завантаження, відображення перемикається з форми пошуку на секцію результатів, і викликається функція displayResults() для візуалізації знайденої інформації. Можливі помилки на кожному етапі обробляються у блоці catch.

Отримання зображення автомобіля реалізовано асинхронною функцією findCarImage(brand, model, year). Вона перевіряє наявність усіх необхідних параметрів та API ключа, повертаючи шлях до зображення-заглушки у випадку їх відсутності. Функція формує пошуковий запит для Pexels API (наприклад, "2022 Ford Mustang car"), здійснює fetch запит з заголовком авторизації. У випадку неуспішного запиту або відсутності фото, повертається шлях до заглушки. Якщо фотографії знайдені, функція аналізує alt текст перших п'яти фотографій: якщо в описі є згадка марки або моделі, повертається URL цього зображення. Якщо точного збігу в описі не знайдено, функція повертає URL першого зображення з результатів Pexels або заглушку.

Генерація посилань на магазини виконується функцією generateSearchLinks(query, deny = false). Вона визначає основний пошуковий термін – ID деталі, якщо він вказаний, або комбінацію назви деталі та параметрів автомобіля. Пошуковий термін кодується для використання в URL. Функція здійснює ротацію списку українських магазинів, якщо deny встановлено в true, та формує три посилання на українські магазини, підставляючи закодований пошуковий термін у відповідні шаблони URL. Аналогічно формуються два

посилання на іноземні магазини, з урахуванням специфіки RockAuto, де при пошуку за ID деталі цей ID використовується для параметра partnum. Функція повертає масив об'єктів, кожен з яких містить назву магазину та сформоване URL. На рис. 3.1 показано приклад картки товару, що генерується цією функцією.

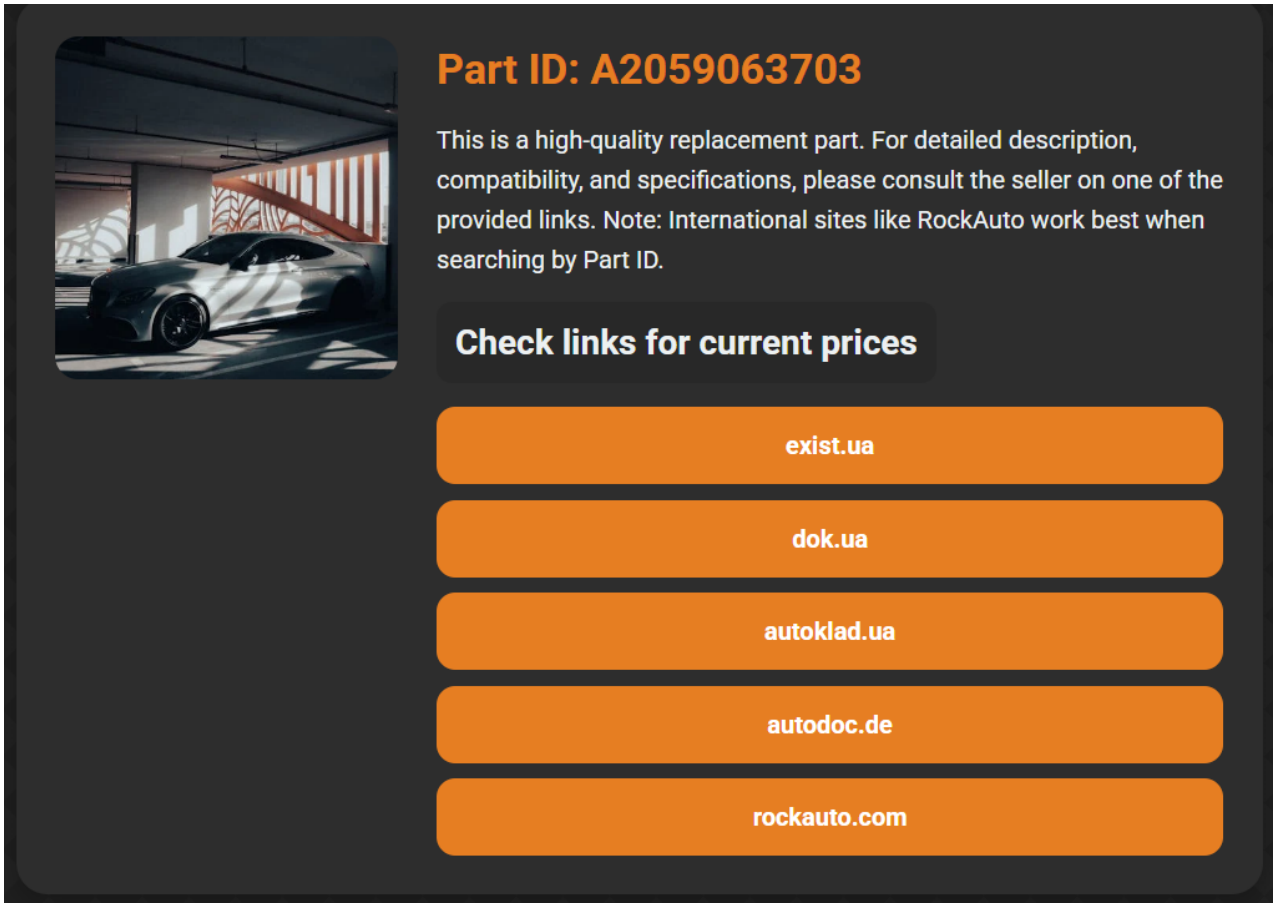


Рисунок 3.1 - Приклад відображення результату пошуку в веб-додатку

Відображення результатів здійснюється функцією displayResults(query, imageUrl, links). Вона формує заголовок для знайденої деталі на основі вхідних даних query, готує статичний опис, який є локалізованим. Далі функція динамічно генерує HTML-код для картки результату за допомогою шаблонних рядків. Цей HTML включає тег <img> з src, встановленим на imageUrl, заголовок деталі, опис, заглушку для ціни, а також контейнер з кнопками-посиланнями, згенерованими на основі масиву links. Згенерований HTML вставляється у div#results-view, після чого додаються обробники подій до нових кнопок "Підтвердити пошук", що викликає goToHome(), та "Шукати інші варіанти", що викликає performSearch(query, true).

3.3. Тестування проектованого веб-ресурсу

Тестування програмного продукту є невід'ємним етапом його життєвого циклу, спрямованим на перевірку відповідності реалізованого функціоналу визначеним вимогам, а також на виявлення та усунення помилок. Для веб-додатку "AI Car Parts Finder" було проведено комплексне ручне тестування.

3.3.1. Методологія тестування

З огляду на клієнтську архітектуру додатку, основним методом тестування було обрано **ручне тестування за принципом "чорної скриньки" (black-box testing)**. Цей підхід дозволяє оцінити роботу системи з точки зору кінцевого користувача, не заглиблюючись у внутрішню реалізацію коду. Взаємодія з інтерфейсом та перевірка отриманих результатів здійснювалися відповідно до розроблених тест-кейсів.

В якості основного інструменту для налагодження та аналізу використовувалися **інструменти розробника веб-браузера (Chrome DevTools)**, зокрема вкладки "Console" для відстеження помилок JavaScript та "Network" для моніторингу запитів до зовнішніх API.

3.3.2. Тестування функціональних вимог (тест-кейси)

Для перевірки основного функціоналу було розроблено ряд тест-кейсів, що охоплюють ключові сценарії використання системи. Приклади таких тест-кейсів наведено в таблицях 3.1 – 3.7.

Таблиця 3.1 ТС-01: Пошук за ID деталі

Ідентифікатор	ТС-01: Пошук за ID деталі
Актори	Користувач сайту
Початкові умови	Користувач повинен увійти на сайт
Очікуваний результат	Відображається сторінка результатів з заголовком, що містить введений ID, зображенням автомобіля та 5 прямими посиланнями на магазини.
Порядок дій	1. Натиснути кнопку “Find parts”. 2. Ввести існуючий ID деталі у відповідне поле (напр., "26060-1AA0B"). 3. Натиснути кнопку "Search".
Альтернативні шляхи	Відсутні

Таблиця 3.2 ТС-02: Пошук за параметрами автомобіля

Ідентифікатор	ТС-02: Пошук за параметрами автомобіля
Актори	Користувач сайту
Початкові умови	Користувач повинен увійти на сайт
Очікуваний результат	Відображається сторінка результатів з коректним заголовком, зображенням Nissan Rogue та 5 посиланнями.
Порядок дій	1. Відкрити сторінку пошуку. 2. Заповнити поля: Марка (напр., "Nissan"), Модель ("Rogue"), Рік ("2012"), Назва деталі ("Headlight"). 3. Натиснути "Search".
Альтернативні шляхи	Відсутні

Таблиця 3.3 ТС-03: Неповні дані для пошуку

Ідентифікатор	ТС-03: Неповні дані для пошуку
Актори	Користувач сайту
Початкові умови	Дані для пошуку введені неправильно, або не повністю
Очікуваний результат	З'являється модальне вікно з повідомленням про помилку та проханням перевірити введені дані. Перехід на сторінку результатів не відбувається.
Порядок дій	1. Відкрити сторінку пошуку. 2. Не правильні дані для пошуку. 2a. Заповнити лише поле "Марка". 2b. Заповнити лише поле "Модель".
Порядок дій	2c. Заповнити лише поле "Рік випуску". 2d. Заповнити лише поле "Деталь". 3. Відповідь системи
Альтернативні шляхи	*2a. Заповнити лише поле "Марка".* 2a-1. Система відповіла “Введено лише марку”
	2b. Заповнити лише поле "Модель". 2-1. Система відповіла “Введено лише модель”
	2c. Заповнити лише поле "Рік випуску". 2c-1. Система відповіла “Введено лише рік випуску”
	2d. Заповнити лише поле "Деталь". 2d-1. Система відповіла “Введено лише деталь”

Таблиця 3.4 ТС-04: Перемикання мови

Ідентифікатор	ТС-04: Перемикання мови
Актори	Користувач сайту
Початкові умови	Користувач повинен увійти на сайт
Очікуваний результат	Весь текст інтерфейсу змінюється на український.
Порядок дій	1. Користувач вирішив змінити мову 1a. Користувач натиснув на кнопку "UA" в заголовку сайту. 1b. Користувач натиснув на кнопку "UA" в заголовку сайту. 2. Користувач продовжує пошук.
Альтернативні шляхи	*1a. Користувач натиснув на кнопку "UA" в заголовку сайту.* 1a-1. Мову змінено на українську.
	1b. Користувач натиснув на кнопку "EN" в заголовку сайту. 1b-1. Мову змінено на англійську.

Таблиця 3.5 ТС-05: Збереження та використання історії пошуків

Ідентифікатор	ТС-05: Збереження та використання історії пошуків
Актори	Користувач сайту
Початкові умови	Виконати пошук (ТС-02)
Очікуваний результат	Форма пошуку автоматично заповнюється даними з обраного запиту. Повторне натискання "Search" виконує той самий пошук.
Порядок дій	1. Повернутися на сторінку пошуку. 2. Натиснути на кнопку з відповідним запитом в секції "Recent Searches".
Альтернативні шляхи	Відсутні

Тестування зручності використання (Usability Testing) було ретельно проведено з метою оцінки загального досвіду взаємодії користувача з додатком. Цей процес охоплював кілька ключових аспектів. Насамперед, перевірялася **адаптивність** інтерфейсу: за допомогою Chrome DevTools імітувалися мобільні пристрої, планшети та десктопи для підтвердження коректності відображення та функціонування всіх елементів на різних ширинах екрану. Було підтверджено, що основні елементи інтерфейсу масштабуються та перебудовуються, забезпечуючи

при цьому збереження читабельності та доступності. Далі, перевірялася **навігація**, щоб упевнитися, що всі навігаційні елементи, такі як логотип, що слугує посиланням на головну сторінку, кнопка "Find Parts" та кнопки на сторінці результатів, коректно направляють користувача до відповідних секцій додатку.

Була оцінена читабельність тексту: аналізувалася контрастність та розміри шрифтів в обох темах – світлій та темній – для забезпечення максимального комфорту читання для користувача.

Таблиця 3.6 ТС-06: Перевірка функції "Шукати інші варіанти"

Ідентифікатор	ТС-06: Перевірка функції "Шукати інші варіанти"
Актори	Користувач сайту
Початкові умови	Виконати пошук (ТС-02).
Очікуваний результат	Сторінка оновлюється. Список посилань на українські магазини змінюється. Зображення та назва деталі залишаються незмінними.
Порядок дій	1. На сторінці результатів натиснути кнопку "Search Others".
Альтернативні шляхи	Відустні

Таблиця 3.7 ТС-07: Помилка API зображень

Ідентифікатор	
Актори	Розробник сайту
Початкові умови	1. Введено невірний Pexels API ключ у script.js.
Очікуваний результат	Відображається сторінка результатів з зображенням-заглушкою (car-placeholder.png) замість фото автомобіля.
Порядок дій	1. Виконати пошук (ТС-02).
Альтернативні шляхи	Відсутні

За результатами проведеного ручного тестування можна зробити висновок, що розроблений веб-додаток відповідає усім сформульованим функціональним та нефункціональним вимогам. Основний функціонал пошуку, персоналізації та навігації працює стабільно. Система коректно обробляє як успішні сценарії, так і

випадки з некоректними даними або помилками API. Виявлені незначні недоліки у верстці на проміжних етапах розробки були успішно усунені.

3.4. Опис взаємодії з зовнішніми сервісами та API

Ефективність та функціональність розробленого веб-додатку значною мірою залежить від його здатності взаємодіяти з зовнішніми сервісами для отримання та агрегації даних. В рамках даного проекту було реалізовано два ключових типи такої взаємодії: прямий запит до програмного інтерфейсу додатків (API) для отримання зображень та розробка методології формування прямих пошукових посилань на сторонні веб-ресурси.

3.5.1. Інтеграція з Pexels API для отримання зображень

Мета інтеграції. Основною метою інтеграції з сервісом Pexels було збагачення результатів пошуку візуальним контентом. Відображення зображення автомобіля, для якого шукається деталь, значно покращує користувацький досвід (UX), оскільки надає користувачеві візуальне підтвердження та підвищує впевненість у правильності запиту.

Процес авторизації. Pexels API використовує стандартний механізм авторизації за допомогою API-ключа. Для отримання доступу необхідно було зареєструвати акаунт на порталі для розробників Pexels та згенерувати унікальний ключ. Згідно з документацією API, цей ключ повинен передаватися в заголовку кожного HTTP-запиту до сервісу. В реалізації на JavaScript це виглядає наступним чином:

JavaScript

```
// Приклад заголовка для запиту
const headers = {
Authorization: 'ВАШ_PEXELS_API_КЛЮЧ'
};
```

Формування запиту. Для пошуку зображень використовується кінцева точка (endpoint) https://api.pexels.com/v1/search. Запит до неї здійснюється за допомогою асинхронного методу fetch() в JavaScript. Ключовим аспектом є динамічне формування пошукового рядка (параметр query), який складається з даних,

введених користувачем. Для підвищення релевантності результатів до запиту додається ключове слово "car".

Приклад формування URL для запиту:

JavaScript

```
// Дані, введені користувачем
const brand = 'Nissan';
const model = 'Rogue';
const year = '2012';

// Формування пошукового запиту
const searchQuery = `${year} ${brand} ${model} car`;

// Формування повного URL з кодуванням спеціальних символів
const apiUrl = `https://api.pexels.com/v1/search?query=$
{encodeURIComponent(searchQuery)}&per_page=5`;
```

Використання `encodeURIComponent()` є обов'язковим для коректної передачі рядка з пробілами та іншими символами в URL. Параметр per_page=5 вказує API повернути до 5 найбільш релевантних зображень, що є основою для подальшої інтелектуальної фільтрації на стороні клієнта.

Обробка відповіді та інтелектуальна фільтрація. Після успішного виконання запиту, Pexels API повертає відповідь у форматі JSON, що містить масив об'єктів photos. Кожен об'єкт представляє одне зображення та містить різноманітну метаінформацію, включаючи URL-адреси зображень у різних розмірах та текстовий опис (alt).

Щоб уникнути показу випадкових або нерелевантних зображень, які API може повернути за відсутності точного збігу, було реалізовано механізм клієнтської фільтрації:

1. Скрипт перебирає отриманий масив з 5 фотографій.
2. Для кожного фото отримується його опис з поля alt.
3. Опис фото та введені користувачем назви марки (brand) і моделі (model) переводяться у нижній регістр для порівняння без урахування регістру.

4. Здійснюється перевірка, чи містить опис зображення назву марки або моделі автомобіля.

5. Якщо збіг знайдено, URL цього зображення (photo.src.large) негайно повертається для відображення. Це вважається успішним результатом.

6. Якщо після перевірки всіх 5 фотографій релевантного збігу не знайдено, функція повертає URL першого зображення зі списку як найбільш імовірний варіант, або ж, для більш суворого підходу, повертає шлях до локального зображення-заглушки (images/car-placeholder.png). Це дозволяє уникнути ситуацій, коли на запит "1995 Opel Vectra" показується фото сучасного міста.

Цей підхід є прикладом "інтелектуальної" обробки даних на стороні клієнта, що підвищує якість та точність результатів, які бачить кінцевий користувач.

3.5.2. Методологія формування прямих посилань на інтернет-магазини

Обґрунтування підходу. Ключовим завданням проекту є надання користувачеві шляхів для придбання знайденої деталі. Пряма інтеграція з API інтернет-магазинів для отримання цін та наявності виявилася неможливою через дві фундаментальні причини:

1. **Відсутність публічних API:** Більшість інтернет-магазинів не надають безкоштовних публічних API для сторонніх розробників.
2. **Політика безпеки браузерів (CORS):** Прямі запити з JavaScript-коду нашого сайту до сторінок іншого домену (наприклад, exist.ua) блокуються браузером.

Тому було розроблено методологію **генерації прямих пошукових посилань**. Цей підхід є надійним та ефективним інженерним рішенням в умовах зазначених обмежень.

Процес дослідження шаблонів URL. Для реалізації цього методу було проведено ручний аналіз кожного цільового інтернет-магазину. Для цього виконувалися тестові пошукові запити безпосередньо на сайтах магазинів, після чого аналізувалася структура URL-адреси у рядку браузера. Було виявлено, що кожен сайт використовує стабільний шаблон для передачі пошукових запитів через параметри URL.

Структура даних. На основі дослідження було створено масиви об'єктів у JavaScript, які зберігають інформацію про кожен магазин:

- UKRAINIAN_STORES
- FOREIGN_STORES

Кожен об'єкт у цих масивах має наступну структуру:

```
JavaScript
{
    name: 'dok.ua', // Назва магазину для відображення на кнопці
    urlTemplate:
'https://dok.ua/ua/search/all?keyword={searchTerm}' // Шаблон URL з
плейсхолдером
}
```

Така структура дозволяє легко додавати нові магазини або змінювати існуючі в майбутньому, не торкаючись основної логіки.

Алгоритм генерації посилань. Функція generateSearchLinks(query, deny) реалізує наступний алгоритм:

1. Визначається основний пошуковий термін (searchTerm): якщо користувач ввів ID деталі, використовується він; в іншому випадку, термін формується з марки, моделі, року та назви деталі.

2. Пошуковий термін кодується за допомогою функції encodeURIComponent() для безпечної вставки в URL (наприклад, пробіли замінюються на %20).

3. Якщо параметр deny має значення true (що відповідає натисканню кнопки "Шукати інші варіанти"), індекс стартового українського магазину зсувається для забезпечення ротації.

4. Скрипт ітерує по масивах UKRAINIAN_STORES та FOREIGN_STORES.

5. Для кожного магазину береться його urlTemplate та за допомогою методу .replace('{searchTerm}', encodedSearchTerm) створюється фінальне посилання.

6. Особлива логіка застосовується для rockauto.com: якщо пошук іде за ID деталі, він підставляється у параметр partnum, оскільки цей сайт має спеціалізований пошук за номерами, що значно підвищує точність.

7. Функція повертає масив об'єктів, готових для відображення у вигляді кнопок-посилань.

При розгляді обраної методології виділяються її значні переваги та певні недоліки. Головними перевагами є надійність, оскільки структура URL-адрес змінюється значно рідше, ніж візуальний дизайн веб-сторінок, що забезпечує стабільність посилань. Крім того, гарантується актуальність інформації: користувач завжди бачить найсвіжіші дані про ціни та наявність безпосередньо на сайті продавця. Метод також вирізняється простотою та швидкістю реалізації, оскільки не потребує серверної частини і виконується миттєво у браузері користувача. Важливою перевагою є повне дотримання політики безпеки, що дозволяє уникнути проблем, пов'язаних із обмеженнями CORS.

Водночас, існують і певні недоліки. До них належить неможливість безпосереднього відображення та порівняння цін на нашому сайті, що вимагає від користувача переходу на сторонні ресурси. Крім того, процес додавання нових магазинів до системи потребує ручного аналізу їхніх пошукових шаблонів. Однак, зважаючи на всі ці фактори, обрана методологія є оптимальним компромісом, що забезпечує необхідну функціональність та надійність у рамках існуючих технічних обмежень проєкту.

В результаті, було детально висвітлено структуру проєкту, надаючи опис основних файлів, таких як index.html, style.css та script.js, а також їхнє призначення в загальній архітектурі вебдодатку. Розглянуто програмну реалізацію ключових функціональних модулів, що становлять основу вебдодатку "AI Car Parts Finder". Це включає процеси ініціалізації системи, механізми перемикання мов та тем, логіку обробки пошукових запитів, а також виконання "інтелектуального" пошуку, що реалізується через взаємодію з Pexels API для отримання зображень та генерацію прямих посилань на інтернет-магазини автозапчастин. Описано логіку відображення результатів пошуку та механізми роботи з історією пошуків користувача. Для кращого розуміння реалізації окремих функцій були наведені концептуальні приклади коду. Завдяки проведеному тестуванню було підтверджено повну працездатність основного функціоналу розробленого вебдодатку.

4. РОЗРОБКА АЛЬТЕРНАТИВНОГО КЛІЄНТА НА ОСНОВІ TELEGRAM BOT API

4.1. Обґрунтування вибору платформи Telegram

Для розширення доступності системи та надання користувачам альтернативного способу взаємодії, було прийнято рішення про розробку чат-бота. В якості платформи було обрано **Telegram** з огляду на наступні переваги:

- *Висока популярність.* Telegram є одним з найпопулярніших месенджерів в Україні та світі, що забезпечує доступ до широкої аудиторії потенційних користувачів без необхідності встановлення додаткових програм.
- *Bot API.* Telegram Bot API надає розробникам потужний набір інструментів для створення інтерактивних ботів, включаючи підтримку кнопок під повідомленнями (inline keyboards), форматування тексту (Markdown), відправку медіафайлів (зображень) та управління діалогом.
- *Кросплатформеність.* Бот, створений для Telegram, автоматично доступний на всіх платформах, де є клієнт месенджера (iOS, Android, Windows, macOS, Web).
- *Безпека та надійність.* Telegram відомий своїм фокусом на безпеці та стабільній роботі інфраструктури.

В порівнянні з альтернативами, такими як Viber або Facebook Messenger, Telegram Bot API на момент розробки надавав більш гнучкі та добре документовані можливості для створення складних діалогових сценаріїв, що було вирішальним фактором.

4.2. Архітектура та технології розробки чат-бота

На відміну від веб-додатку, який є повністю клієнтським, Telegram-бот за своєю природою потребує серверної частини (backend), яка б виконувала логіку та взаємодіяла з серверами Telegram.

Архітектура чат-бота. Архітектура системи є клієнт-серверною, де сервери Telegram виступають посередником між користувачем та нашим серверним скриптом. На рис. 4.1. Представлена схема алгоритму роботи Telegram-бота.

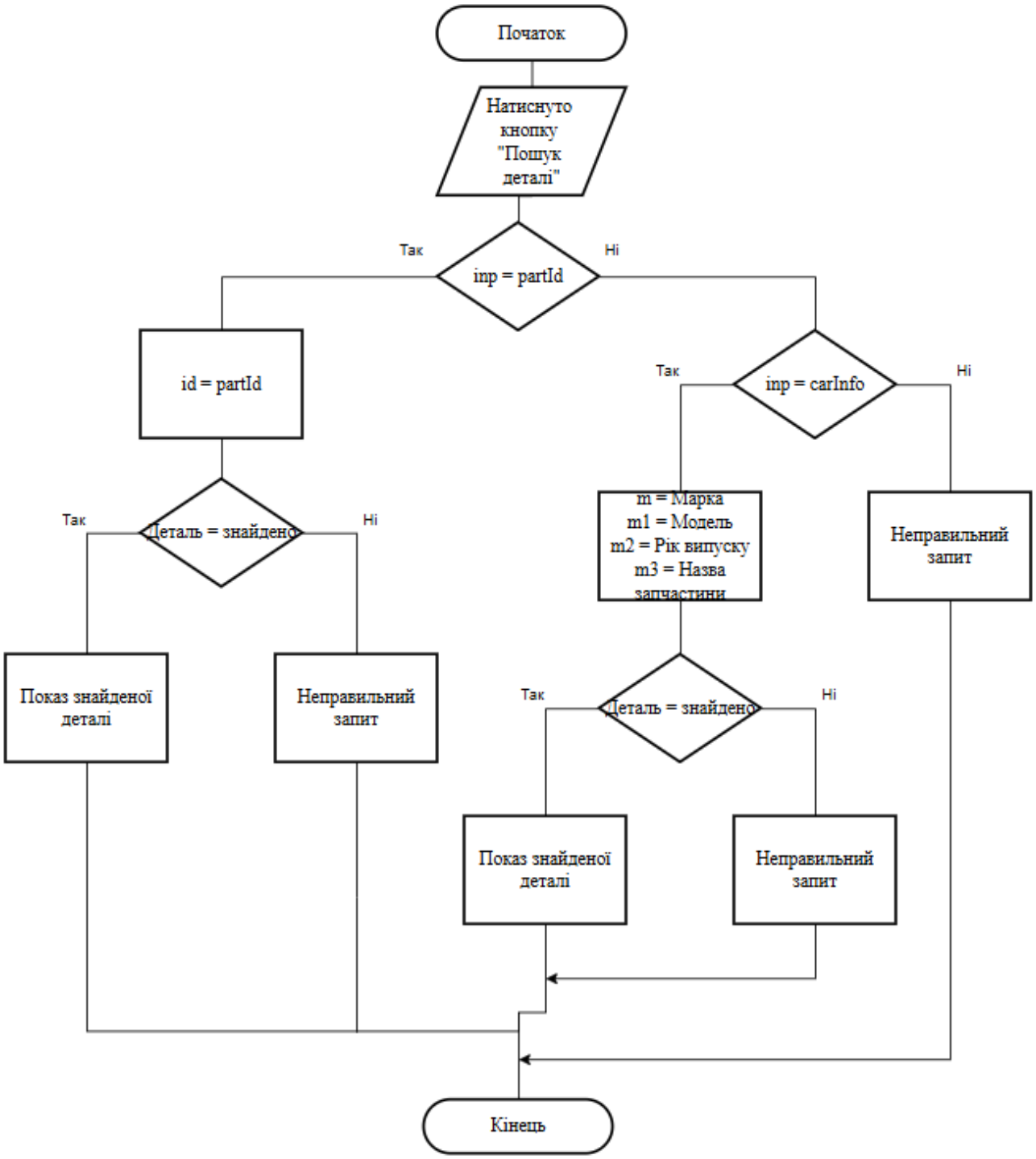


Рисунок 4.1 - Схема алгоритму роботи Telegram-бота

Схема роботи створеного Telegram-бота для пошуку автомобільних деталей докладно ілюструє послідовність дій та логіку прийняття рішень від початкової взаємодії з користувачем до відображення результатів. Процес функціонування бота розпочинається, коли користувач натискає кнопку "Пошук деталі" в його

інтерфейсі, і система отримує цей запит. Після цього бот переходить до аналізу введених даних, визначаючи тип запиту.

Якщо введений запит відповідає ідентифікатору деталі (inp = partId), система присвоює це значення як id і перевіряє наявність відповідної деталі. У разі успішного знаходження, бот відображає знайдену деталь; якщо ж деталь не знайдено, користувач отримує повідомлення про "Неправильний запит".

У випадку, коли запит не є ідентифікатором деталі, система перевіряє, чи містить він інформацію про автомобіль (inp = carInfo). Якщо так, бот очікує надання таких параметрів, як марка, модель, рік випуску та назва запчастини. Після отримання цих даних відбувається спроба знайти деталь. При успішному знаходженні деталь відображається; в іншому випадку користувач отримує повідомлення "Неправильний запит".

Якщо ж введений запит не відповідає ані ідентифікатору деталі, ані інформації про автомобіль, система одразу розпізнає його як "Неправильний запит" і повідомляє про це користувача. Після завершення одного з цих сценаріїв, тобто після показу знайденої деталі або інформування про неправильний запит, процес роботи бота завершується. Таким чином, схема чітко демонструє розгалужену логіку бота, що дозволяє ефективно обробляти різні типи вхідних даних для пошуку деталей та забезпечувати релевантний зворотний зв'язок користувачеві.

Для розробки створеного Telegram-бота, призначеного для пошуку автомобільних деталей, вибір мови програмування зупинився на Python. Це рішення обумовлене простотою синтаксису цієї мови, широкою доступністю бібліотек, необхідних для веб-розробки та ефективної взаємодії з API, а також значною і активною спільнотою розробників. В якості основної бібліотеки для роботи з Telegram ботом була обрана python-telegram-bot. Вона є однією з найбільш популярних і функціональних бібліотек для взаємодії з Telegram Bot API, забезпечуючи підтримку асинхронності та пропонуючи зручні абстракції для обробки різноманітних елементів, таких як команди, повідомлення та кнопки. Для виконання HTTP-запитів, зокрема для взаємодії з Pexels API, використовується бібліотека requests.

4.3. Реалізація основного функціоналу бота

Функціонал бота був розроблений таким чином, щоб максимально відтворювати можливості веб-сайту у форматі діалогу.

4.3.1. Керування діалогом та станами

Для реалізації послідовного збору даних від користувача (наприклад, при пошуку за параметрами авто) було використано механізм ConversationHandler з бібліотеки python-telegram-bot. Цей інструмент дозволяє визначати різні "стани" розмови (наприклад, очікування марки авто, очікування моделі тощо) та призначати відповідні обробники для кожного стану, що робить діалог логічним та керованим. Було визначено наступні стани: CHOOSE_LANG, CHOOSE_SEARCH_METHOD, GET_PART_ID, GET_BRAND, GET_MODEL, GET_YEAR, GET_PART_NAME. Команда /cancel дозволяє вийти з будь-якого стану та повернутися до головного меню.

4.3.2. Головне меню та навігація

Для забезпечення зручної навігації було реалізовано концепцію "Головного меню". Після кожної завершеної дії (пошуку, зміни мови, скасування) або при помилці, бот надсилає користувачеві повідомлення з головним меню, яке містить наступні кнопки:

- "🔍 Знайти Запчастини"
- "🌐 Змінити Мову"
- "❓ Допомога"

Це створює для користувача постійну та зрозумілу точку повернення, з якої можна ініціювати будь-яку дію.

4.3.3. Реалізація пошуку та відображення результатів

Пошуковий функціонал бота відтворює логіку веб-сайту:

1. **Ініціація:** Користувач натискає кнопку "Знайти Запчастини" в головному меню або вводить команду /findparts.

2. **Вибір методу:** Бот пропонує обрати пошук за ID або за параметрами авто за допомогою кнопок.

3. **Збір даних:** Бот послідовно запитує необхідну інформацію.

4. **Обробка запиту:** Після отримання всіх даних, серверний скрипт викликає ті ж самі функції find_car_image_bot() та generate_search_links_bot(), які є Python-аналогами функцій з веб-сайту.

5. **Відображення результатів:** Результат надсилається єдиним повідомленням. Якщо знайдено зображення автомобіля, воно надсилається за допомогою send_photo, а весь текстовий опис та посилання додаються у caption. Якщо зображення не знайдено, вся інформація надсилається текстовим повідомленням. Посилання на магазини представлені у вигляді кнопок під повідомленням (inline keyboard), що робить їх зручними для натискання.

На рис. 4.2 показано приклад відповіді бота за відповідним результатами пошуку.

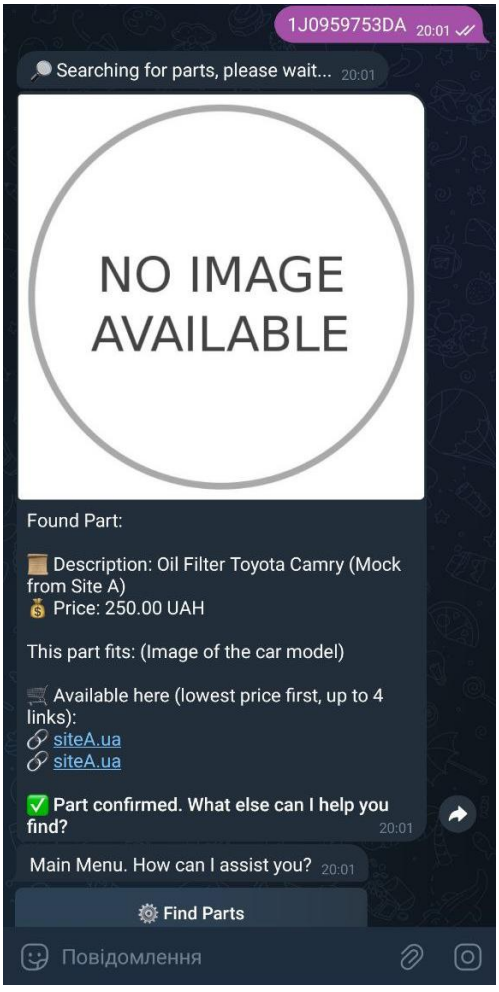


Рисунок 4.2 - Приклад відображення результатів пошуку у Telegram-боті

В результаті, було успішно розроблено альтернативний клієнт для доступу до функціоналу системи у вигляді Telegram-бота, що значно розширює доступність сервісу. Також обґрунтовано вибір платформи Telegram та відповідного технологічного стеку, заснованого на Python, що забезпечило швидку та ефективну розробку. У цьому розділі описано клієнт-серверну архітектуру бота та реалізовано його основні функції, які дублюють можливості веб-сайту, включаючи пошук, багатомовність та відображення результатів із застосуванням інтерактивних елементів Telegram. Додатково, впроваджено концепцію "Головного меню", яка значно покращує навігацію та загальний користувацький досвід у діалоговому інтерфейсі. Таким чином, створений Telegram-бот є успішним доповненням до основного веб-додатку, демонструючи гнучкість розробленої логіки пошуку, що може бути адаптована для різних платформ.

ВИСНОВКИ

У рамках даної дипломної роботи було успішно розроблено клієнтський вебдодаток під назвою "AI Car Parts Finder", призначений для інтелектуального пошуку автомобільних запчастин.

Проведено глибокий аналіз предметної області, що дозволило виявити ключові проблеми, з якими стикаються користувачі при онлайн-пошуку автозапчастин, а також обґрунтувати актуальність розробки запропонованої системи. Цей аналіз також включав огляд існуючих рішень та визначення основних вимог до нового додатку.

Була спроектована архітектура односторінкового вебдодатку (SPA), що функціонує повністю на стороні клієнта. У процесі проектування були визначені основні компоненти системи, обрано технологічний стек, що включає HTML, CSS, JavaScript, Pexels API та LocalStorage, а також розроблені принципи користувацького інтерфейсу, орієнтовані на зручність та простоту використання.

Реалізовано основний функціонал системи, який охоплює можливість пошуку запчастин як за ID деталі, так і за комбінацією параметрів, таких як марка, модель, рік випуску та назва деталі. Результати пошуку відображаються у вигляді карток, що містять зображення відповідного автомобіля, отримане через Pexels API з елементами інтелектуальної фільтрації, а також прямі посилання на три українських та два міжнародних інтернет-магазини. Додаток надає функціонал "пошуку інших варіантів" з ротацією українських магазинів. Інтерфейс підтримує багатомовність (українська, англійська) та дозволяє перемикатися між світлою та темною темами оформлення. Система зберігає та відображає історію останніх п'яти пошукових запитів з можливістю їх очищення. Зручність користування підвищується завдяки автозаповненню для поля "Марка автомобіля", інтуїтивній навігації, що включає кнопку "Додому" та посилання на Telegram-бота, а також інформативним модальним вікнам для сповіщень про завантаження та помилки.

Проведене тестування ключових функцій підтвердило повну працездатність системи та її відповідність основним вимогам, демонструючи коректну обробку запитів, відображення інформації та реакцію на дії користувача.

Розроблений вебдодаток "AI Car Parts Finder" є функціональним прототипом, який демонструє можливість створення корисного інструменту для автовласників з використанням доступних вебтехнологій та безкоштовних API. Він може слугувати основою для подальшого розвитку та розширення функціоналу.

У майбутньому планується розширення функціоналу системи за декількома напрямками. Передбачається збільшення кількості підтримуваних українських та міжнародних інтернет-магазинів. Також планується додати можливість фільтрації результатів пошуку за певними критеріями, такими як виробник деталі або її стан (нова чи вживана), за умови наявності відповідних API. Удосконалення включатиме глибшу інтеграцію з Pexels API для точнішого підбору зображень або використання альтернативних джерел зображень. Для розширення можливостей системи передбачається розробка серверної частини, що дозволить зберігати ширшу базу даних запчастин, користувацькі акаунти та відгуки. В планах також повна реалізація функціоналу Telegram-бота, який би дублював усі можливості вебсайту. Розглядається впровадження прогресивних вебдодатків (PWA) для можливості встановлення на мобільні пристрої та часткової роботи офлайн. Нарешті, планується додати функцію "Поділитися результатом" через Web Share API, що спростить поширення знайденої інформації.

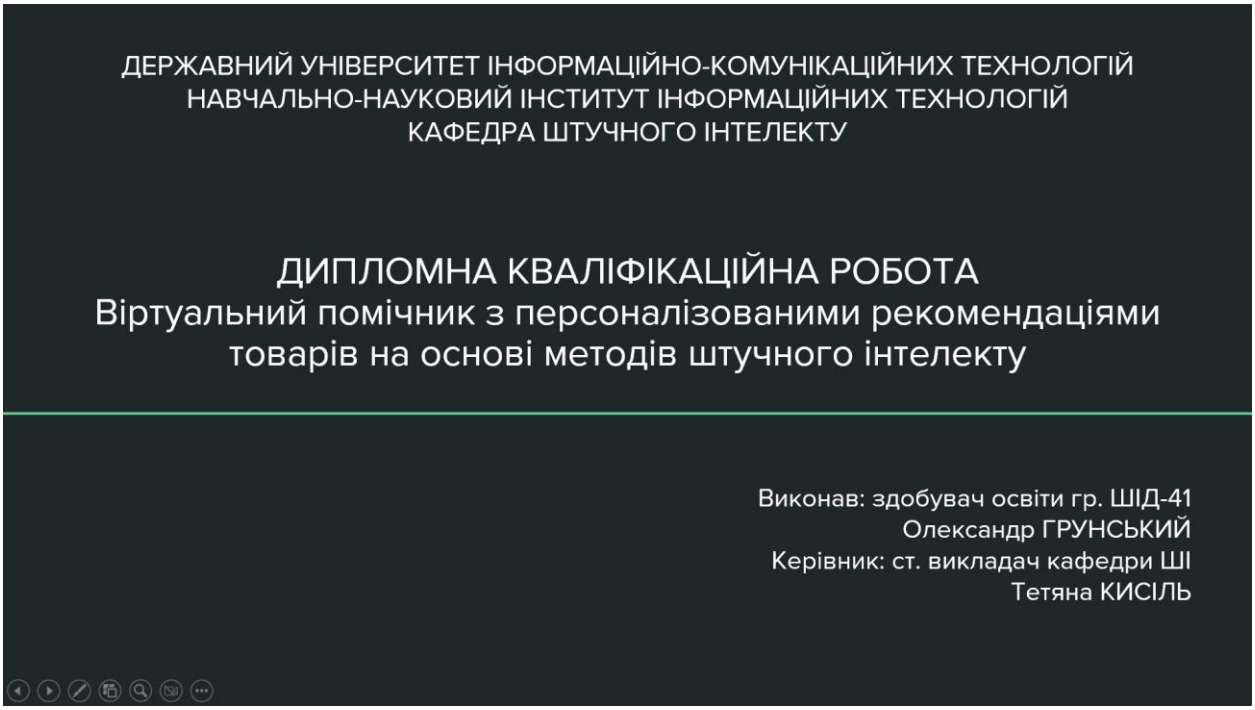
ПЕРЕЛІК ПОСИЛАНЬ

1. Al-Otaibi, Z. S., & Al-Marri, A. M. (2023). A Survey on Chatbot-Based Recommender Systems for E-commerce. *Journal of King Saud University – Computer and Information Sciences*, *35*(2), 173-186.
2. Alpaydin, E. (2020). *Machine Learning: The New AI*. Cambridge, MA: MIT Press.
3. Bengio, Y., & Glorot, X. (2010). *Understanding the difficulty of training deep feedforward neural networks*. In Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics (AISTATS).
4. Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (2013). Recommender systems survey. *Knowledge-Based Systems*, *46*, 109–132.
5. Chollet, F. (2018). *Deep Learning with Python*. Shelter Island, NY: Manning Publications.
6. Conversational AI Summit. (2023). *Latest Trends in Conversational AI and Virtual Assistants*. Proceedings of the Conversational AI Summit. (Приклад узагальненої конференції, якщо немає конкретної статті)
7. Géron, A. (2019). *Hands-On Machine Learning with Scikit-Learn, Keras, and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems* (2nd ed.). Sebastopol, CA: O'Reilly Media.
8. Gu, Y., Zhang, W., & Xu, J. (2020). Deep Learning for Recommender Systems: A Survey. *ACM Computing Surveys*, *52*(1), 1–35.
9. Jannach, D., & Adomavicius, G. (2016). Towards a Framework for Evaluating the Economic Value of Recommender Systems. In *Proceedings of the 10th ACM Conference on Recommender Systems (RecSys '16)* (pp. 165–172). ACM.
10. Jurema, T. (2021). *Recommendation Systems with Python*. Birmingham, UK: Packt Publishing.
11. Kang, W., & Lee, S. (2022). A Deep Reinforcement Learning Approach for Personalized Product Recommendation in E-commerce. *Expert Systems with Applications*, *205*, 117765.
12. Liu, H., Fang, H., Song, X., Zhao, X., Li, X., & Lv, J. (2021). Reinforcement Learning for Personalized Recommendation: A Survey. *Journal of Big Data*, *8*(1), 1–28.

13. Russell, S., & Norvig, P. (2020). *Artificial Intelligence: A Modern Approach* (4th ed.). Pearson Education.
14. Shani, G., & Gunawardana, A. (2011). Evaluating Recommender Systems. In F. Ricci, L. Rokach, B. Shapira, & P. B. Kantor (Eds.), *Recommender Systems Handbook* (pp. 257–297). Springer.
15. Sharda, R., Delen, D., Turban, E. (2020). *Business Intelligence, Analytics, and Data Science: A Managerial Perspective* (5th ed.). Boston: Pearson.
16. Wang, Y., Zhu, J., & Li, T. (2021). Building Intelligent Conversational Agents: A Survey of Dialogue Systems with Deep Learning. *Artificial Intelligence Review*, *54*(3), 1633–1675.
17. Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). Deep Learning Based Recommender System: A Survey and New Perspectives. *ACM Computing Surveys*, *52*(1), 1–38.
18. Гудфеллоу, І., Бенджіо, Й., Курвіль, А. (2017). *Глибоке навчання*. Бостон: MIT Press.
19. Кисіль Т.М., Грунський О.С., Ломака В.І. МОДЕРНІЗАЦІЯ ПЛАТІЖНОЇ СИСТЕМИ VISA ЗАСОБАМИ ШТУЧНОГО ІНТЕЛЕКТУ /3 Всеукраїнська науково-практична конференція «Сучасні інтелектуальні інформаційні технології в науці та освіті». Збірник тез. – К.: ДУІКТ, 2023, с. 109, 110
20. Шен, Н., & Стівенсон, С. (2020). *Природна мова для розробників: від основи до глибокого навчання*. Київ: Видавництво "Основи".

ДОДАТКИ

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ



2

Мета роботи: Розробка веб-помічника, що забезпечує ефективний інтелектуальний пошук автомобільних запчастин за різними критеріями, надаючи користувачеві релевантну інформацію та посилання для придбання.

Об'єкт дослідження: процес розробки веб-помічника для пошуку товарів за запитом користувача.

Предмет дослідження: система інтелектуального пошуку з використанням ШІ

Основні завдання кваліфікаційної роботи:

1. Визначення вимоги до системи.
2. Проектування архітектури веб-додатку.
3. Розробка користувацького інтерфейсу з підтримкою двох мов (українська, англійська).
4. Реалізація функціонал пошуку запчастин за ID деталі або за параметрами автомобіля.
5. Інтеграція API для отримання даних (зображення автомобілів, посилання на магазини).
6. Проведення тестування розробленої системи.

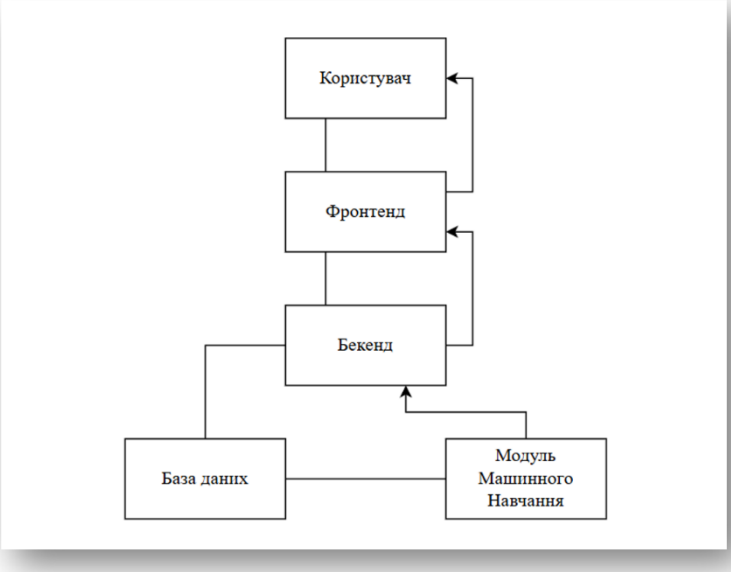
Огляд використаних технологій - API та інше

- **Pexels API: Для отримання зображень, що відповідають запиту користувача.**

- **Принципи "AI Search"**
  Пошук реалізовано через агрегацію та інтелектуальну обробку даних з відкритих джерел та формування прямих посилань на сторінки пошуку українських та міжнародних онлайн-магазинів автозапчастин.

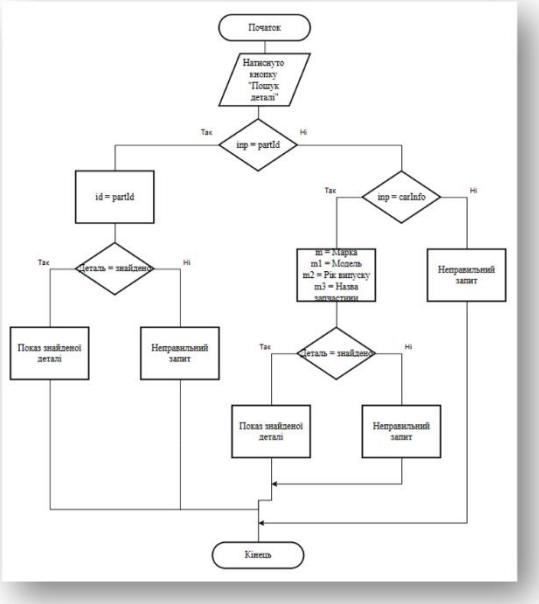
- **LocalStorage браузера: Для збереження налаштувань користувача (тема, мова, історія пошуків).**

Схема функціоналу проекту



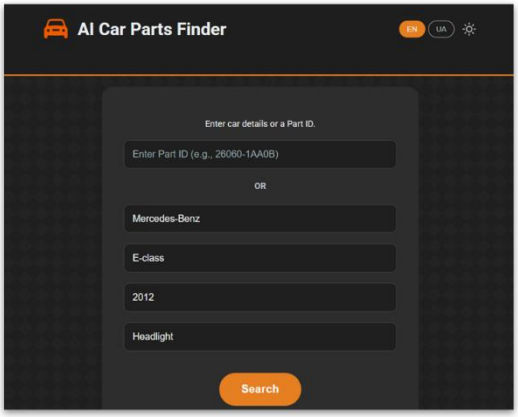
Алгоритм роботи проекту

5

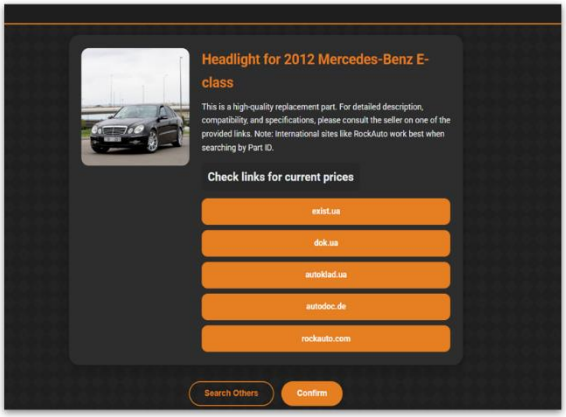


Демонстрація роботи створеного проекту

6



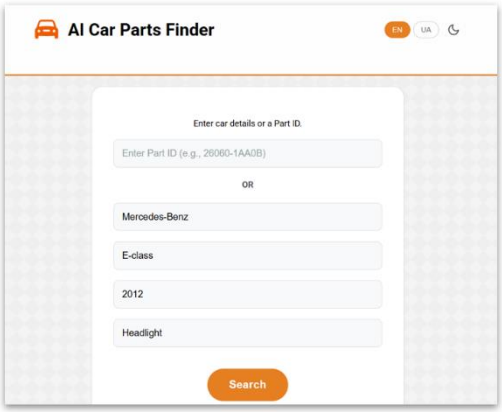
Головна сторінка з панеллю для введення інформації
(темний варіант)



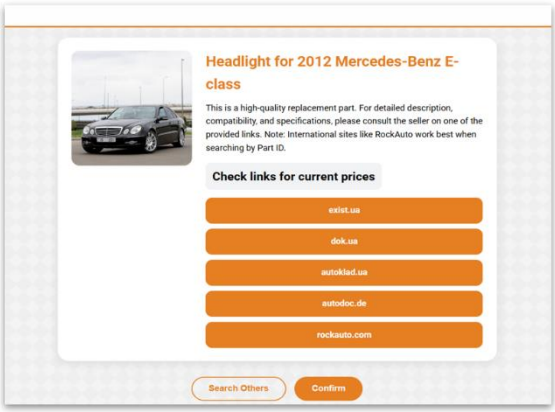
Сторінка з результатом пошуку
(темний варіант)

Демонстрація роботи створеного проекту

7



Головна сторінка з панеллю для введення інформації
(світлий варіант)



Сторінка з результатом пошуку
(світлий варіант)

8

Результати роботи та тестування

Досягнуті результати:

- Розроблено функціональний веб-сайт, що відповідає поставленим завданням.
- Реалізовано зручний користувацький інтерфейс.
- Забезпечено пошук автозапчастин за ключовими параметрами.
- Інтегровано зовнішні сервіси для збагачення результатів пошуку.

Тестування:

- Проведено ручне тестування основного функціоналу (пошук, перемикання мови, теми, відображення результатів).

Висновки

- Розроблена система з корисним інструментом для автовласників.
- Розроблений користувацький інтерфейс з підтримкою двох мов (українська, англійська).
- Реалізовано функціонал пошуку запчастин за ID деталі або за параметрами автомобіля.
- Інтегровано API для отримання даних (зображення автомобілів, посилання на магазини).
- Проект демонструє можливість створення корисних веб-додатків з використанням сучасних frontend-технологій та безкоштовних API.