Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування факультету)

Обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи

Магістр
(освітньо–кваліфікаційний рівень)

на тему: "Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля"

08-23.МКР.007.00.000 ПЗ

Виконав: студент 2 курсу, групи КІ–19м,
спеціальності:
123 – комп'ютерна інженерія
(шифр і назва напряму підготовки)

Маренич Богдан Сергійович
(прізвище та ініціали)

Керівник: Савицька Людмила Анатоліївна
(прізвище та ініціали)

м. Вінниця – 2020 рік

АНОТАЦІЯ

Дана кваліфікаційна робота присвячена розробці Web–орієнтованого програмного засобу для організації та супроводження процесу підбору і замовлення автомобіля.

Досконало розглянуто питання необхідності та доцільності створення даної системи перед аналогами, розглянуто основні методи та середовища розробки онлайн ресурсу, проаналізовано та зроблено висновки на основі результатів дослідження можливостей застосування сучасних тенденцій розробки. Покращено процес пошуку та підбору автомобілів за рахунок використання сучасних інформаційних технологій веб–розробки, фреймворків та високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій.

Перевірка працездатності здійснена шляхом практичних випробувань інформаційної системи в реальних умовах.

ANNOTATION

This qualification work is devoted to the development of Web−oriented software for organizing and maintaining the process of selection and ordering a car.

The questions of the necessity and expediency of creating this system before analogues are considered thoroughly, the main methods and environment of development of an online resource are considered, the conclusions are analyzed and made based on the results of research on the possibilities of applying modern development tendencies. Improved search and selection of cars through the use of modern information technology web development, frameworks and high-performance non-relativistic database, where the processing of information is faster than the previous generation of technologies.

Performance testing is carried out through practical tests of the information system in real conditions.

ЗМІСТ

					08-23.МКР.007.00.000 ПЗ			
					Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розробив		Маренич Б.С.				Арк. 1	Аркушів 2	
Керівник		Савицька Л. А.						
Рецензент		Яремчук Ю. Є.						
Н. Контроль		Швець С. І.				ВНТУ, гр. КІ-19м		
Затверджую		Азаров О. Д.						

ВСТУП

Веб–технологія повністю перевернула наші уявлення про роботу з інформацією та і з комп'ютером взагалі. Виявилось, що традиційні параметри розвитку обчислювальної техніки – продуктивність, пропускна спроможність, ємкість пристроїв, що запам'ятовують, не враховували головного «вузького місця» системи – інтерфейсу з людиною. Застарілий механізм взаємодії людини з інформаційною системою стримував впровадження нових технологій і зменшував вигоду від їх застосування. І лише коли інтерфейс між людиною і комп'ютером був спрощений до легкого сприйняття звичайною людиною, послідкував безпрецедентний вибух інтересу до можливостей обчислювальної техніки [1].

З розвитком технологій гіпертекстової розмітки в Інтернеті почало з'являтися все більше сайтів, тематика яких була абсолютно різною – від сайтів великих компаній, що оповідають про успіхи компанії та її провали, до сайтів маленьких фірм, що пропонують відвідати їх офіси в межах одного міста [2].

Наразі існує чимало аналогів даної інформаційної системи, якими користується значна маса людей та обробляються тисячі запитів одночасно, що призводить до низької швидкодії та систематичних збоїв у роботі. Розробка даного Web-орієнтованого програмного засобу для організації та супроводження процесу підбору і замовлення автомобіля спрямована на пришвидшення процесу обробки та оптимізації даних за рахунок використання сучасних інформаційних технологій веб–розробки.

Даний веб–сайт створений для компанії, що спрямовує свою діяльність на пригін автомобілів з Америки в Україну. Користувачеві представлена детальна інформація про компанію, її контактні дані, різноманітні варіанти автівок, які він може переглянути та обрати для себе одне або ж декілька, та перейти до узгодження й замовлення. Також, є можливість власноруч

задавати необхідні параметри бажаного авто та отримувати усі можливі варіанти, що будуть відповідати вказаним характеристикам.

В реаліях сьогодення для більшості громадян України власне авто – предмет розкоші, а не засіб пересування, як це у більшості європейських країн. Дана розробка створюється з тією метою, щоб змінити дану ситуацію та зробити автомобілі доступними для кожного, відкрити легкий доступ до автомобільного ринку Америки, на якому ціни є значно меншими. Щоб краще розуміти можливу економію, яку надасть розробка даного проекту, необхідно проаналізувати авторинок України та зробити висновки, що економія на експортуванні автомобіля з США буде складати близько 25–40 % від вартості аналогічного йому в Україні. Даний сайт підходить для людей будь–якої вікової групи та соціального статусу, які прагнуть нарешті отримати омріяне авто дешевше та зекономити власні кошти, що робить дану розробку **актуальною**.

Метою роботи є покращення процесу пошуку та підбору автомобілів за рахунок використання сучасних інформаційних технологій веб–розробки, фреймворків та високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій за рахунок процесу оптимізації пошуку даних.

Об’єкт дослідження: процес оптимізації пошуку актуальних даних при підборі автомобілів.

Предмет дослідження: система оптимального підбору авто за заданими параметрами та пошуку на ринку автомобілів.

Для виконання поставленої у магістерській дипломній роботі мети необхідно виконати таке завдання:

1) виконати аналіз сучасного стану розвитку інформаційних систем пошуку та підбору даних;
2) створити технологічний ланцюжок роботи інформаційної системи підбору та замовлення авто;
3) розробити алгоритм роботи даної інформаційної системи;

4) вдосконалити процес оптимізації пошуку даних;
5) провести детальне тестування розробленого програмного забезпечення.

Результати, представлені у даній роботі, опубліковано та випробувано на XLVIII Науково–технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2019).

Інформаційна система підбору та замовлення авто / Коробейнікова Т. І., Маренич Б. С. // Збірник Матеріалів XLVIII Науково–технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2019). Режим доступу: https://conferences.vntu.edu.ua/index.php/all–fitki/all–fitki–2019/paper/view/6877/5639.

ВСТУП

Веб–технологія повністю змінила наші уявлення про роботу з інформацією та з комп'ютером в цілому. Як з’ясувалось, звичні нам параметри розвитку обчислювальної техніки, а саме: продуктивність, пропускна спроможність, ємкість пристроїв, які запам'ятовують, не врахували одного головного «вузького місця» системи ― інтерфейсу з кінцевим користувачем. Дещо застарілий варіант взаємодії людини з інформаційною системою обмежував впровадження сучасних технологій і зменшував користь від їх застосування. І лише тоді, коли інтерфейс між користувачем і комп'ютером був спрощений до легкого використання звичайною людиною, за цим слідував безпрецедентний всплеск інтересу до перспектив розвитку обчислювальної техніки [1].

З інтенсивним розвитком технологій гіпертекстової розмітки на просторах Інтернету почало з'являтися все більше web–сторінок, тематика яких була дуже різноманітною ― від сайтів маштабних компаній, що розповідають про успіхи компанії та її невдачі, до сайтів дрібних фірм, що пропонують завітати до їх офісів в межах одного міста [2].

Наразі існує чимало аналогів даної інформаційної системи, якими користується значна маса людей та обробляються тисячі запитів одночасно, що призводить до низької швидкодії та систематичних збоїв у роботі. Розробка даного Web–орієнтованого програмного засобу для організації та супроводження процесу підбору і замовлення автомобіля спрямована на пришвидшення процесу обробки та оптимізації даних за рахунок використання сучасних інформаційних технологій веб–розробки.

Даний веб–сайт створений для компанії, що спрямовує свою діяльність на пригін автомобілів з Америки в Україну. Користувачеві представлена детальна інформація про компанію, її контактні дані, різноманітні варіанти автівок, які він може переглянути та обрати для себе одне або ж декілька, та

перейти до узгодження й замовлення. Також, є можливість власноруч задавати необхідні параметри бажаного авто та отримувати усі можливі варіанти, що будуть відповідати вказаним характеристикам.

В реаліях сучасності для переважної частини громадян України власне авто ― справді предмет розкоші, а не засіб для пересування, як це у більшості сусчасних європейських країн. Дана розробка розроблена з метою, щоб змінити поточну ситуацію та зробити автомобілі доступними кожному, надати легкий доступ до автомобільного ринку Америки, де ціни є значно меншими. Для кращого розуміння можливої економії, яку надасть розробка даної роботи, необхідно вивчити авторинок України та зробити висновки, що економія на експортуванні автомобіля під ключ з США буде складати близько 25–40 % від вартості аналогічного йому авто в Україні. Розроблений програмний засіб адаптований для людей будь–якої вікової категорії та соціальної групи, що прагнуть отримати якісне авто дешевше та зекономити кошти, що робить дану розробку актуальною.

Метою роботи є покращення процесу пошуку та підбору автомобілів за рахунок використання сучасних інформаційних технологій веб–розробки, фреймворків та високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій за рахунок процесу оптимізації пошуку даних.

Об’єкт дослідження: процес оптимізації пошуку актуальних даних при підборі автомобілів.

Предмет дослідження: система оптимального підбору авто за заданими параметрами та пошуку на ринку автомобілів.

Для виконання поставленої у магістерській кваліфікаційній роботі мети необхідно виконати таке завдання:

— виконати аналіз сучасного стану розвитку інформаційних систем пошуку та підбору даних;

— створити технологічний ланцюжок роботи інформаційної системи підбору та замовлення авто;

— розробити алгоритм роботи даної інформаційної системи;

— вдосконалити процес оптимізації пошуку даних;

— провести детальне тестування розробленого програмного забезпечення.

Результати, представлені у даній роботі, опубліковано та випробувано на XLVIII Науково–технічної конференції від факультету інформаційних технологій та комп'ютерної інженерії (2019).

Інформаційна система підбору та замовлення авто / Коробейнікова Т. І., Маренич Б. С. // Збірник Матеріалів XLVIII Науково–технічної конференції від факультету інформаційних технологій та комп'ютерної інженерії (2019). Режим доступу: https://conferences.vntu.edu.ua/index.php/all–fitki/all–fitki–2019/paper/view/6877/5639.

1 СУЧАСНИЙ СТАН РОЗВИТКУ ІНФОРМАЦІЙНИХ СИСТЕМ ПОШУКУ ТА ПІДБОРУ ДАНИХ

1.1 Поняття інформаційної системи

Доволі часто можна почути словосполучення “інформаційні системи” у зв'язку з різними сферами людської діяльності. Так що ж входить у поняття “інформаційної системи” в сучасному світі, які їхні функції та завдання, як їх можна класифікувати? Інформаційна система (ІС) являє собою поєднання технічного, організаційного та програмного забезпечення, що призначене для своєчасного забезпечення потрібною інформацією.

Інформаційна система ― система обробки даних, що працює разом із організаційними ресурсами, які забезпечують та розподіляють інформацію.

Основне завдання інформаційних систем ― це забезпечення певних інформаційних потреб у межах певної конкретної області. Інформаційні системи, в яких немає сучасні технічних засобів обробки інформації, та виконання всіх операцій здійснюється людиною, називаються ручними. Зараз їх практично немає. Системи, що приймають участь в процесі обробки даних користувачів, технічних засобів та програмного забезпечення ― автоматизовані ІС. Системи, які виконують всі операції по обробці даних самостійно ― це автоматичні ІС.

Інформаційна система, в ролі системи керування, значно пов’язується як з системами зберігання та видачі інформації, так і з різноманітними іншими системами, які надають обмін інформацією в процесах керування. Вона включає в себе сукупність комплексу засобів та методів, які дозволяють кінцевому користувачу зберігати, поширювати і корегувати відібрану інформацію. Інформаційні системи розпочали своє існування з моменту появи суспільства, так як на кожному етапі його розвитку була потреба в керуванні. Основною задачею інформаційної системи є створення необхідно для підприємства інформації, потрібної для успішного управління всіма її ресурсами, створення інформаційно–технічного середовища для керування її

діяльністю. Інформаційна система може функціонувати і без використання комп'ютерної техніки ― це питання економічної доцільності. В будь–якій ІС управління вирішуються задачі трьох типів:

— задачі розпізнавання образів;

— задачі моделювання;

— задачі прийняття рішень та оптимізації.

Автоматизована інформаційна система (АІС) ― це взаємопов'язаний об'єм даних, програмних засобів, персоналізації, стандартизованих процедур, що призначені для збирання, орброблення, розподілення, зберігання та представлення інформації згідно вимог, які взято з цілей організації. Сьогодні, у вік стрімного розвитку інформації, практично кожна ІС використовує комп'ютеризацію, і тому, надалі, під інформаційними системами розглядатимемо саме автоматизовані інформаційні системи [3].

ІС включають в свою складову: технічні засоби для оброблення даних, програмне середовище і відповідний обслуговуючий персонал. Інформаційну основу утворюють чотири складові частини:

— засоби для фіксування та збору інформації;

— засоби для передачі даних та повідомлень;

— засоби для зберігання ресурсів;

— засоби для аналізування, оброблення та відтворення інформації.

Різноманітність ІС з кожним роком стає все більшою й більшою. В залежності від призначення інформаційної системи, можна виділити основні системи: керуючі (АСУТП, АСУВ), проектні (САПР), наукового пошуку даних (АСНД), діагностичні, системи моделювання, підготовки та прийняття рішень (СППР), а в залежності від сфер застосування ― на економічні, адміністративні, медичні, виробничі, екологічні, навчальні, військові, військові та інші.

Найбільш важливими факторами, що впливають на впровадження ІС, є необхідність організацій та користувачів, а також присутність відповідних

інструментів для їх створення. Найбільший суттєвий внесок до розвитку ІС надали досягнення в галузі комп'ютерної техніки та телекомунікації, мережевих ресурсів.

До відомих напрямів автоматизування інформаційно–управлінської сфери діяльності до організаційних структур відносять:

— автоматизацію оброблення документів шляхом імпортування систем для редагування тексту, автоматизацію обміну відомостями через різні види комунікації;

— автоматизування праці менеджерів на основі комп'ютерних систем комплексних ІС, що надають підтримку в здійнені рішень, та електронних помічників, що робить можливим підвищити рівень організації показників успішності менеджерів.

Перехід до інформаційних систем надає змогу менеджеру отримувати швидкий доступ до усієї зібраної інформації з тим, щоб в перспективі ефективно її використовувати для вирішення заданиї завдань.

Для повсякдневних умов характерно використання високопродуктивних локальних систем інформації, що базуються на використанні сучасних інформаційних технологій, особливо єдиної локальної комп'ютерної мережі. Внутрішня інформаційна система управління являє собою набір інформаційних процесів для здійснення потреб в ресурсах на різних рівнях аналізу й прийняття рішень. ІС включає в себе компоненти редактування інформації, внутрішні та зовнішні канали для передачі даних.

Інформація, в часності її автоматизована обробка, й наразі залишається суттєвим фактором підняття ефективності праці будь–якого підприємства. Значну роль у використанні ресурсів відіграють варіанти їх внесення до бази даних, редагування, накопитчення і передачі, організоване збереження даних і їх видача в коректній формі, нагромадження нової числової та графічної інформації.

В умовах сьогодення у великих підприємствах розроблені та ефективно працюють інформаційні системи, що контролюють процес підготовки і

прийняття рішень управління, та приймають наступні завдання: оброблення даних та інформації, реалізацію інтелектуальних ресурсів з метою створення даних. Інформаційні системи керування одне за одним реалізовують принципи цілісності трудового процесу та інформаційного супроводу з допомогою застосування технічних засобів нагромадження, редагування, збору, обробки і передачі ресурсів в поєднанні з використанням математичної статистики і моделей прогнозування аналітичних прорахунків та інших доцільних прикладних засобів. У виробничо–господарській системі фірми забезпечується структура інформації “знизу – вверх”, конкретизація ресурсів “зверху – вниз”, а також уніфікується інформаційний процес, що орієнтований на отримання планової, науково–технічної, облікової та аналітичної інформації.

Підвищення результативності використання ІС набувається шляхом наскрізної структури та компонування інформаційних систем, що дають можливість змінити дублювання і гарантують багаторазове застосування інформації, створюють інтеграційні зв’язки, зменшують кількість показників, скорочують об’єми інформаційних потоків, покращують рівень використання ресурсів. Інформаційна система повинна включати в себе такі функції, як надання та створення комфортних умов для поширення інформації.

Новітня ІС в заданій сфері діяльності підприємства надає можливість забезпечити рішення таких задач:

— прямий доступ до інформаційного ресурсу;

— координацію діяльності та швидше розповсюдження повідомлень;

— взаємодію із спільними по високотехнологічних маршрутах за рахунок користування більш сучасних та наукових методів відтворення та прийому–передачі повідомлень;

— виділення часу для менеджерів усіх категорій на аналіз та прийняття рішень за рахунок зниження показників затраченого часу на виконання малопродуктивної праці;

— використання системного аналізу та проектування оперативного керування на нижній та середніх критеріях керування виробництвом.

Відсутність взаємодії окремих інструментів автоматизації та окремих технологій може стати обмежуючим аспектом, що робить взаємодію ІС нераціональним.

Розміри статутного капіталу можуть бути як підтримуючою прогрес одиницею, так і гальмом для створення інформаційних систем. Малий відсоток керівників будуть наголошувати, що суттєві інвестиції в автоматизацію підприємства, налаштовані на довгострокові перспективи, є фундаментальними в питаннях існування фірми. Для тимчасової перспективи більшість ставить під роздуми реальність окупності інвестицій, оскільки не має явного представлення про місце високоінтелектуальних систем в керуванні підприємством. Проте, досвідчені керівники великих фірм вважають, що в умовах теперішнього ринку застосування ІС надає суттєву гнучкість і значно нижчі попустні витрати їх організацій.

1.2 Дослідження сучасних інформаційних систем для реалізації процесу пошуку та підбору

Пошукова система (ПС) — це спосіб структурування і класифікації інформації на просторах Інтернету. Для користувачів дані системи виглядають як звичайний сайт, якому він задає свій запит. Сайт відповідає на запит майже миттєвою відповіддю, сторінкою сайту, який вміщає в себе відповіді послідовно розташованих одне за одним посилань та різноманітні елементи. Користувачу ж видно — “обличчя” ПС — сайт, є лише інтелектуальною обгорткою величезного програмного комплексу. Беручи до уваги той факт, що програми мають багато спільних характкерних особливостей, їх індивідуальність не підлягає розповсюдженню та є конфіденційною інформацією. Цим пояснено розходження ПС між собою. Пошукова система постійно динамічно оновлюється, розширює свої можливості та змінює алгоритми роботи.

Пошукова система працює на базі пошукових роботів, які безперервно підбирають нову інформацію на просторах Інтернету, щоб не пропустити нові

сторінки або ж їх оновлення. На даний час кількість роботів досить значна, кожен з яких має завдання з пошуку та збору інформації відмінне від інших. Уся знайдені ресурси реєструється та фіксуються індексатором, сукупність знайденого являє собою базу даних ПС, яка є у свій час не сталою величиною. Цілком природньо, адже безперервно додаються нові веб–сторінки, динамічно оновлюється їх контент, а застарілі зазвичай затираються з загальної бази даних. Для отримання доступу до визначеної інформації, користувач подає запит до веб–сервера пошуковика. Який в свою чергу володіє системами видачі, тобто має можливість відразу відсортувати уже зібрані раніше дані та передати релевантні відповіді.

Покращення пошуку — одна з пріоритетних задач сучасного Інтернету. Загальними положеннями якісних характеристик роботи пошукових систем є наповненість бази, релевантність, морфологія мови та її врахування. Для роботи пошукових систем використовуються різні програмні продукти, часто спеціалізовані для конкретних видів пошуковиків. У середовищі розробників для такого програмного забезпечення використовується термін «рушій пошуковика».

Ефективність пошуку інформації тим вище, чим більші коефіцієнти повноти та точності пошуку і менші час та інші ресурси, необхідні на його проведення. Багато пошукових вузлів з метою підвищення ефективності пошуку дозволяють замість простого пошуку проводити так званий розширений пошук. Для цього вони пропонують користувачу заповнити форму, завдяки якій може бути звуженою область пошуку — за тематикою, за типом сайтів, за датою тощо.

Інформаційно–пошукова система (ІПС) (пошуковий сервер) — комплекс положень, що призначені для збереження та підбору шукаємих файлів, відомостей, зафіксованих положень. Всі ІПС умовно розподіляються на два типи — пошукові каталоги та пошукові машини. Майже кожний пошуковий каталог оснащується ще й пошуковою машиною, а пошукова машина не завжди пропонує користувачу достатньо детальний каталог.

Пошукова машина (Search engine) — засоби, які здійснюють пошук інформації у мережі Інтернет за запитом користувача.

Пошуковий каталог нагадує систематичний каталог звичайної бібліотеки та надає рубрикатор тем.

Пошуковий сервер (або інформаційно–пошукова система) — це Web–вузол, який спеціалізується на пошуку інформації [4].

Існує два типи пошукових систем. Перші відносяться до розряду каталогів — дані про сторінки Інтернет розсортовані у них за тематикою. З каталогами зручно працювати, коли Ви точно знаєте та обираєте загальну тематику, а не окрему сторінку. Наприклад, коли Ви досліджуєте історію культури Давньої Греції, то можете обрати пункти "Історія" та "Культура". А, якщо Ви фахівець у галузі державного управління, то необхідними для Вас рубриками у каталозі є "Право", "Управління" тощо. Розділи (або рубрики, або категорії) каталогів різних пошукувачів частково співпадають, частково — ні. Якщо Ви не знайшли рубрику, яка б відповідала Вашому запиту, на одній ІПС, спробуйте знайти її на іншій. Всі каталоги побудовані за принципом "від загального — до одиничного" та мають зручну деревовидну структуру. Другий тип — це пошукові машини. Користувач, заходячи на сторінку, вводить слово, або словосполучення і проводить пошук. Багато пошукувачів мають як пошукові машини, так і каталоги.

Алгоритм пошуку — метод, користуючись яким пошукова машина приймає рішення про включення чи не включення посилання на сторінку або документ у результаті пошуку.

Коефіцієнт повноти пошуку — це відношення кількості одержаних релевантних результатів до загальної кількості існуючих у пошуковому масиві документів, релевантних даному пошуковому запиту.

Коефіцієнт точності пошуку — це пропорційне співвідношення кількості отриманих релевантних результатів до усієї кількості документів, посилання на які представлені у наданій відповіді ІПС.

Вибір пошукової системи, що найкраще підходить для розв'язання ваших завдань, залежить від того, що ви хочете знайти. Основний об'єкт індексації пошукової системи — тексти. Однак існують пошуковики, що дозволяють робити пошук за картинками, mp3–файлами, архівами програм, новинами тощо. Варто враховувати також область дії пошукової системи. Серед них розрізняють локальні (обмежені національним доменом, певною мовою) і глобальні пошукові системи. Зазвичай глобальні системи добре покривають американський Інтернет і трохи гірше «знають» іншу частину. Тому, якщо ваш пошук свідомо обмежений країною або мовою, краще користуватися локальним пошуковиком (в Україні — це пошукова система META). Крім того, більшість «російськомовних» (локальних) пошукових систем шукають тексти багатьма мовами — українською, білоруською, англійською й ін. Відрізняються ж вони від глобальних систем тим, що в основному обмежуються російськомовними сайтами.

Підбір необхідної інформації відбувається у декілька етапів. Перший етап пошуку характерний підбором інформації в Інтернеті, для цього необхідно проаналізувати тему запиту та результати, що необхідно отримати в після пошуку. Коректно обрана тема пошуку надасть допомогу при підборі ключових слів. Перед вибором певної пошукової системи, необхідно детально дослідити можливості кожної із них. Другий етап полягає у виборі пошукової системи та залежить від власних переконань кінцевого користувача. Третій етап характерний визначенням інструментів, які саме слова є специфічними (ключовими) для шуканого інформаційного ресурсу. Коректно підібрані ключові слова та грамотно сформульований запит кардинально розгортають сектор пошуку й підвищують продуктивність пошуку. Визначивши кінцеву тему пошуку, обравши конкретну систему пошуку, проаналізувавши й обравши ключові слова та виконано опрацювання поданого запиту, переходимо до наступнго етапу — аналізу отриманого списку посилань. На випадко, якщо надані результати не задовольняють пошуковий запит, необхідно зменшити

область пошуку. Це можна зробити завдяки внесення додаткових ключових слів, термінів, що будуть враховані до наступного пошукового запиту.

1.3 Інформаційні технології для реалізації системи підбору та замовлення авто

Дана система створюється з метою створення максимально комфортних умов та поліпшенню доступу до Американського ринку автомобілів. Дана система буде допомагати в підборі автомобіля після вказання основних ключових критеріїв по ньому. Такими критеріями, для прикладу, є тип автомобіля, його двигун, марка, виробник, ціна авто під ключ в Україні та багато інших. Після вибору бажаної моделі йде процес замовлення авто та супровід покупця від моменту покупки на аукціонах Америки і до прибуття на територію України.

Для реалізації даної інформаційної системи створено та оптимізовано веб–сайт за допомогою таких технологій як HTML5, CSS3, JS, SQL.

Для доступу до адміністрування web–сайту необхідно пройти авторизацію, що розуміється як введення логіну та паролю (виконано за допомогою JS) . Після введення даних , ми отримуємо доступ до керування сайтом та його контентом, при умові, що введений логін і пароль є вірним. Якщо ж введено невірні дані ― на сторінці з’явиться повідомлення про помилку при авторизації та проханням спробувати ввести їх знову.

Після авторизації ми отримуємо доступ до керування сайтом та його контентом, маємо можливість загружати нові матеріали в базу даних SQL за допомогою технології JS. Дизайн сайту створено за допомогою HTML5 та CSS3.

HTML (Hyper Text Markup Language) — стандартизована гіпертекстова мова розмітки сторінок в Інтернеті. Переважна більшість веб–сторінок написані за допомогою HTML. Документ HTML

інтерпритується браузером та відображає його склад на екрані у звичному для людського зору вигляді.

HTML є модернізованою гіпертекстовоюб мовою від SGML, що перейняла собі від неї визначення виду файлу та принципи структуризації та розбиття тексту. HTML прийнято вважати штучною мовою, так як вона не відноситься дря жодного з підрозділів мов програмування.

HTML у поєднанні з стилями каскадних таблиць та скриптами — утворюють три загальні технології побудови веб–ресурсів.

Документи HTML мають розширення файлу html, рідше htm. Перевагою даних сторіною та цієї мови гіпертекстової розмітки є те, що їх можна створювати у будь–якому текстовому редакторі.

CSS (Cascading Style Sheets) — спеціалізована мова, що створена для опису візуального зовнішнього вигляду інтернет сторінок, написаних мовами гіпертекстової розмітки даних.

Найбільшого використання CSS має при візуальній презентації сторінок, написаних HTML та XHTML, але даний формат також може застосовуватись до інших видів XML–документів.

CSS — це мова, що побудована на сукупності різноманіть стилів, які визначають відтворення HTML–файлів. Найбільш суттєвою перевагою застосування CSS — абсолютна можливість позиціонування та розділення контенту сторінки та її візуального оформлення. Розділення такого типу дозволило покращити розуміння та простоту змісту, надати покращену гнучкість та управління за відтворенням змісту в різних позиціях, розробити зміст більш структуризованим та доступним. Саме дана логіка була покладена у основу створення цієї технології [5].

JavaScript — це особлива мова програмування, що має за основу об’єктне представлення будь–якого браузера. Його необхідність полягає у тому, щоб дозволити сайту отримати елементи інтерактивності, яких немає звичайний статичним HTML–файл. Особливість JS полягає в тому, що текст програми вбудовується в документ HTML і аналізується самим браузером.

Основним інструментом для створення сценаріїв, який реалізовує можливість на стороні клієнта працювати та здійснювати абсолютну взаємодію з користувачем, займатись управлінням браузера, здійснювати асинхронний обмін даними з серверами, редагувати структуризацію зовнішнього вигляду сторінок.

JavaScript класифіковано як об'єктно–орієнтовану мову з динамічною типізацією даних. Також великою мірою JavaScript підтримує ряд зовсім інших парадигм у галузі програмування, а саме імперативну та функціональну,а також певні архітектурні моєливості, а саме: слабка динамічна типізація даних, автокерування ресурсами пам’яті, наслідування прототипів файлу.

Мова JavaScript має пряме застосування з метою:

— написання інтерактивних сценаріїв;

— створення сторінок–візитівок (Vue.js, ReactJS, AngularJS);

— програмування серверної частини (Node.js);

— стандартизовані сторінки та застосунки (NW.js, Electron);

— мобільних додатків (Cordova, React Native);

— Прикладнепрограмування та їх сценарії (Apache JMeter, Adobe CS);

Node.js ― веб–орієнтована платформа з доступним кодом для створення високопродуктивних додатків у мережі, створених за допомогою мови JavaScript. Основоположником даної платформи є Раян Дал (Ryan Dahl). До недавнього часу Javascript застосовувався для редагування даних в браузері виключно на сторонні користувача, то node.js розширив його можливості та зробив можливим виконувати JavaScript–скрипти на стороні серверу, а також повноцінно відправляти кінцевому користувачеві результат їх роботи. Платформа Node.js дала мові JavaScript суттєвий розвиток та почала базуватисьяк мова загального використання серед розробників [6].

SQL ― мова програмування декларативного типу, створена для взаємодії баз даних, ресурсів та кінцевих користувачів, основним застосуванням є формування запитів, внесення змін та управління керування

реляційною БД, редагування, створення та модифікації існуючої бази даних, здійснення контролю за доступом в систему БД. У своїй основі SQL не має ні системи управління базами даних, ні відокремлених програмним ресурсів. Проте, SQL може створювати повноцінні інтерактивні запити та виступати в ролі інструкцій для управлення даними.

MongoDB — система керування базами даних, яка має документо–орієнтований підхід (СКБД), перевагою якої є відкритий вихідним код, що не залежна від розпису та створення схеми таблиць. MongoDB позиціонується як швидка та масштабована система, дані у якій знаходяться у форматі ключ/значення, а також реляційними СКБД, які є більш зручними та зрозумілими для програміста у формуванні запитів. Код даної нереляційної бази даних MongoDB створений на мові C++ та розповсюджюється в рамках ліцензії AGPLv3.

MongoDB за основу має зберігання документів в JSON — подібному форматі, де мова для формуванян запитів є доволі гнучкою, а також створює індекси для збереження атрибутів, без будь–яких затруднень надає можливість збергіання великих бінарних об’єктів, проводить статистику та веде облік внесени даних до БД, здійснює свою роботу відповідно до парадигми Map/Reduce, має реплікацію та створення можливих відмовостійких конфігурацій. У MongoDB вбудовані засоби набору ресурсів по серверах, за основу даної процедури взяти прототип певного ключа [8].

1.3.1 Засоби реалізації серверної частини

Веб–сервер ― це сервіс, що отримує HTTP–запити від кінцевих користувачів, надає їм HTTP–відповіді на запит, реалізується дана процедура зазвичай із HTML–сторінкою та усіма попутними ресурсами на ній.

Веб–сервером прийнято називати не лише програмне забезпечення, що виконує функціональні обов’язки веб–сервера, а також і сам ПК, на якому дане програмне забезпечення функціонує. Клієнти отримують веб–сервера за

допомогою URL–адреси потрібного їм веб–ресурсу.

Основною метою WEB–серверів є повноцінний доступ до даних, що він у себе включає за допомогою протоколів HTTPS та HTTP. Наразі у світовій мережі вснують мільйони WEB–серверів та кожен із них може розміщувати один та більше різноманітних сайтів. Реалізація даної можливості стала доступною завдяки використання новітніх технологій віртуальних доменів. Переважна кількість WEB–серверів у світі ведуть свою роботу під керуванням операційних систем UNIX та LINUX. ННайбільш розповсюдженим WEB–сервером у світі наразі є APCHE [9].

Сервер по своїй структурі являє собою набір функціональних програм, які виконують управління різноманітними внутрішніми процесами. Для їх функціонування, відповідно, набір даних програм має бути також встановлений на якомусь комп’ютері. Переважно комп'ютер, на якому фізично розміщується уся серверна частина, і називають сервером. Головним функціональним завданням комп'ютера–сервера — відповідати на зщапити кінцевих користувачів та видавати який–небудь відповідний процес, надсилати результати виконання роботи.

1.3.2 Засоби реалізації клієнтської частини

Важливе значення мати правильне розуміння терміну «клієнт». Мова може йти про про клієнтський комп'ютер, через який кінцевий користувач має зв’язок з іншими комп’ютерами у мережі, має серверне та клієнтське програмне забезпечення водночас.

За основу прийнято розуміння, що клієнти та сервери — це програмні одиниці, модулі. Зазвичай вони розташовані на різних комп'ютерах, проте бувають випадки, коли кожна із програм водночас і клієнтська, і серверна, але фізично можуть бути розташовані на одній робочій машині, дана ситуація дає назву серверу — локальний [10].

Модель клієнт–серверної взаємодії прийнято визначати у першу чергу за розподілом обов’язків між сервером та клієнтом. Прийнято відокремлювати 3 види операцій:

— представлення кінцевих даних користувачеві;

— прикладний рівень;

— рівень керування даними.

Дворівнева клієнт–серверна архітектура являє собою програмну взаємодію двох модулів — серверного і клієнтського відповідно. В залежності від типу розподілення між ними функціональних обов’язків їх розділяють на такі види:

— модель тонкого клієнта;

— модель товстого клієнта.

1.4 Підсистеми інформаційної системи підбору та замовлення авто

Для створення й розробки даної інформаційної системи необхідно детально проробити й зкомпонувати шість основних підсистем. Для виконання цих задач необхідно виділити головні складові, з яких і буде складатись база даних, інформацією якої будуть оперувати підсистеми.

Для того, щоб будь–який користувач даної ІС між легко та швидко знайти необхідне йому авто, вирішено сформувати список основних параметрів, по яким в подальшому і буде відбуватись пошук та підбір.

Перший і один з найголовніших параметрів це тип кузову, це може бути: седан, універсал, хетчбек, ліфтбек, мінівен, купе, пікап, кросовер, позашляховик, кабріолет та інші (рис 1.1).

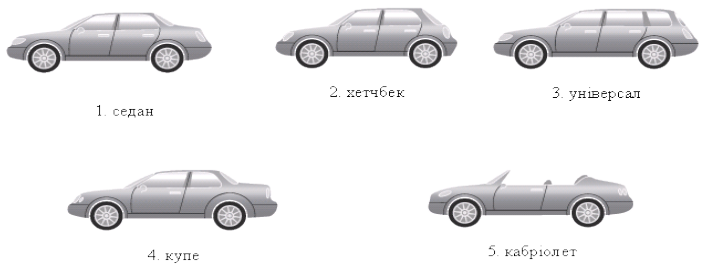

Рисунок 1.1 — Основні типи кузову автомобіля

Двигун — енергосилова машина, що перетворює який–небудь вид енергії на механічну роботу. Двигун внутрішнього згорання – тип двигуна, теплова машина, в якій хімічна енергія палива, що згорає в робочій зоні, перетворюється на механічну роботу. Найчастіше двигун внутрішнього згорання палива використовується у транспортних засобах: автомобілях, мотоциклах, поїздах тощо. Є декілька різновидів типу палива автомобіля, а саме: бензин, дизель, газ, газ/бензин, гібрид, електро (рис. 1.2).

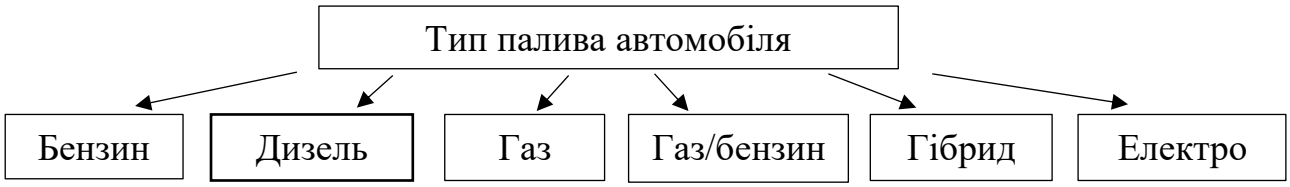

Рисунок 1.2 — Типи палива автомобіля

Коробка передач (КП) транспортних засобів призначена для зміни частоти обертів й обертового моменту в ширших межах, ніж забезпечує ДВЗ трансортного засобу. Як правило, це відноситься до двигунів внутрішнього згоряння (ДВЗ), що мають недостатню здатність до адаптації. Транспортні засоби з паровими або електричними двигунами зазвичай не комплектуються КП, оскільки останні мають високу адаптивність. Є 4 основні типи коробки передач, а саме: механічна коробка (МКПП), автоматична коробка (АКПП), роботизована коробка (РКПП) та варіативна коробка (Варіатор) (рис. 1.3).

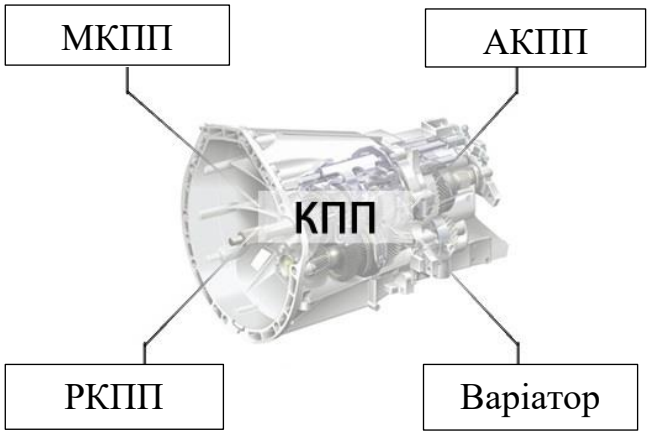

Рисунок 1.3 — Типи коробки переключення передач

Важливу роль для кожного потенційного клієнта при виборі бажаного транспортного засобу є привід автомобіля, він є одним і суттєвих критеріїв при виборі авто (рис. 1.4).

Передній привод (FWD) — конструктивна особливість автомобіля, при якій увесь крутний момент, що переданий з ДВЗ, передано на передню вісь авто.

Задній привід (RWD) — конструктивна особливість автомобіля, при якій увесь крутний момент, що переданий з ДВЗ, передано на задню вісь авто. Реалізація даного розташування може залежати від двигуна і агрегатів трансмісії певного транспортного засобу.

Повний привід (4x4, 4WD, AWD) — конструктивна особливість автомобіля, при якій увесь крутний момент, що переданий з ДВЗ, передано на передню та задню вісь. Окрім вказаних вище технічних характеристик, також необхідно зауважити при виборі автомобіля марку, модель, рік виробництва, об’єм та потужність його двигуна, витрати палива в літрах на 100 кілометрів ходу, актуальних пробіг на момент придбання та інше.

Рисунок 1.4 — Типи приводу автомобіля

Для створення даного інформаційного ресурсу з використанням усіх вищезазначених параметрів для підбору та пошуку авто використано 6 підсистем, а саме: логін/пароль, первинного огляду даних, пошуку авто, підбору авто, виведення даних, виходу (рисунок 1.5).

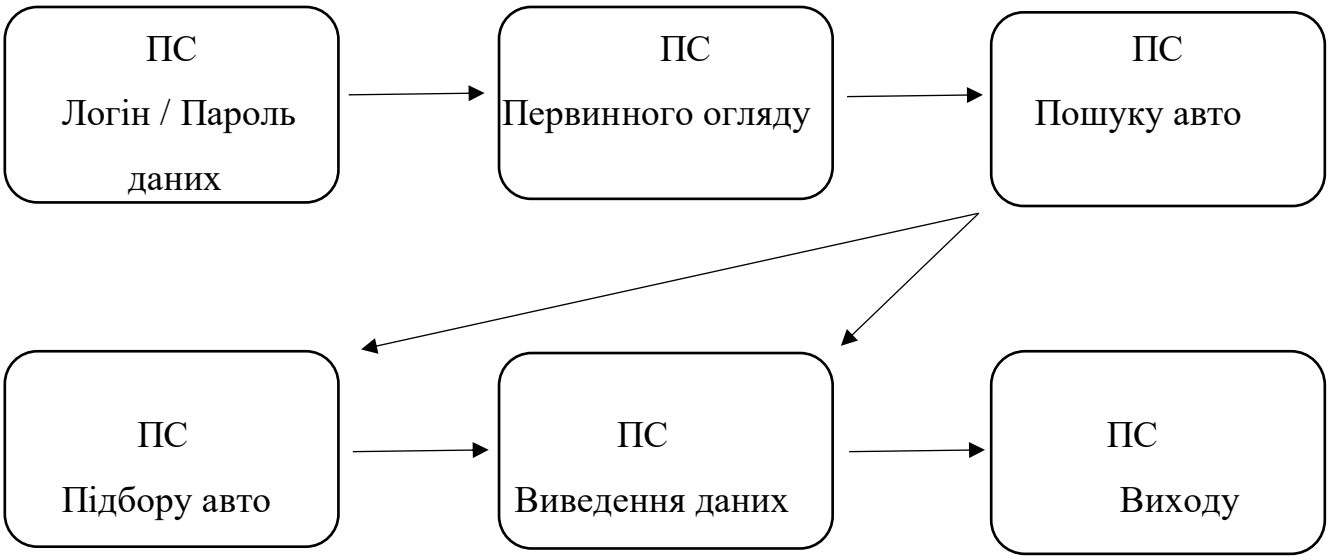

Рисунок 1.5 — Підсистеми інформаційної системи

Кожна підсистема розроблена для виконання відповідних функціональних вимог та у своїй сукупності повноцінно реалізовує інформаційну систему підбору та пошуку авто.

1.4.1 Підсистема логін/пароль

Дана підсистема створена для авторизації користувачів та входження їх до персонального кабінету. При введенні не коректних даних або ж спробі ідентифікації без введеного одного з обов’язкових полів, відбудеться виведення інформуючого повідомлення з відповідною причиною відмови в авторизації користувача. Дана підсистема має два основних типи доступу: адміністратор (admin) та користувач (user). Перший тип має права для редагування даних інформаційної системи, доступ до створення нових та видалення старих елементів системи. Другий тип має можливість користуватись пошуком та підбором автомобілів за усіма існуючими параметрами, зберігати обрані позиції

в особистому кабінеті та отримувати повідомлення про усі їх зміни (рисунок 1.6).

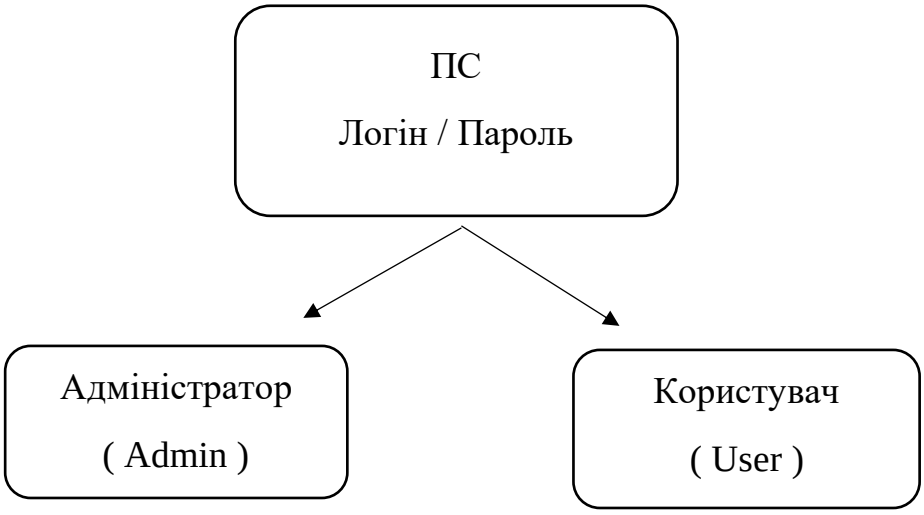

Рисунок 1.6 — Реалізація підсистеми логін/пароль

1.4.2 Підсистема первинного огляду даних

Дана підсистема має ознайомчий характер з структурою інформаційного ресурсу, перевагами перед конкурентами, його можливостями та даними, якими він оперує. Користувачеві надається можливість ознайомитись з усіма технічними параметрами, що застосовуються з метою підбору та пошуку авто та зворотнім зв’язком з адміністрацією даної інформаційної системи (рисунок 1.7).

1.4.3 Підсистема пошуку авто

В даній підсистемі реалізований широкий вибір різних варіантів автомобілів, які запропоновані для вибору потенційним клієнтом. Процес пошуку може відбуватись вручну.

Користувач отримує можливість переглядати представлені авто зі списку та переходити до їх детального технічного опису, або ж автоматично, за допомогою введення ключових символів ідентифікаторів.

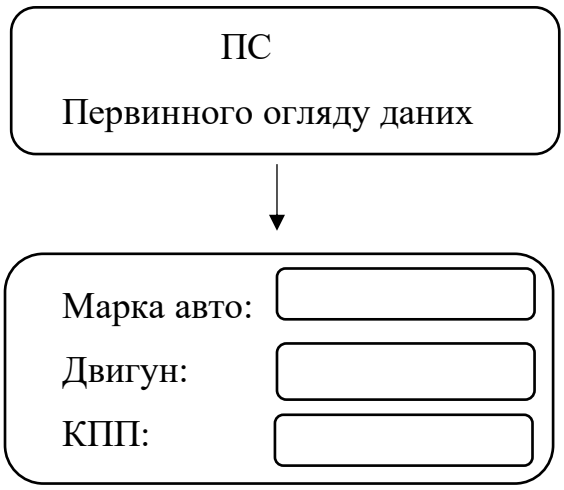

Рисунок 1.7 — Реалізація підсистеми первинного огляду даних

До кожного з представлених користувачеві автомобілів у списку буде прикріплене його фото, пошкодження, детально описані всі його технічні характеристики, такі як: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто з врахуванням усіх послуг компанії з доставкою в Україну (рисунок 1.8).

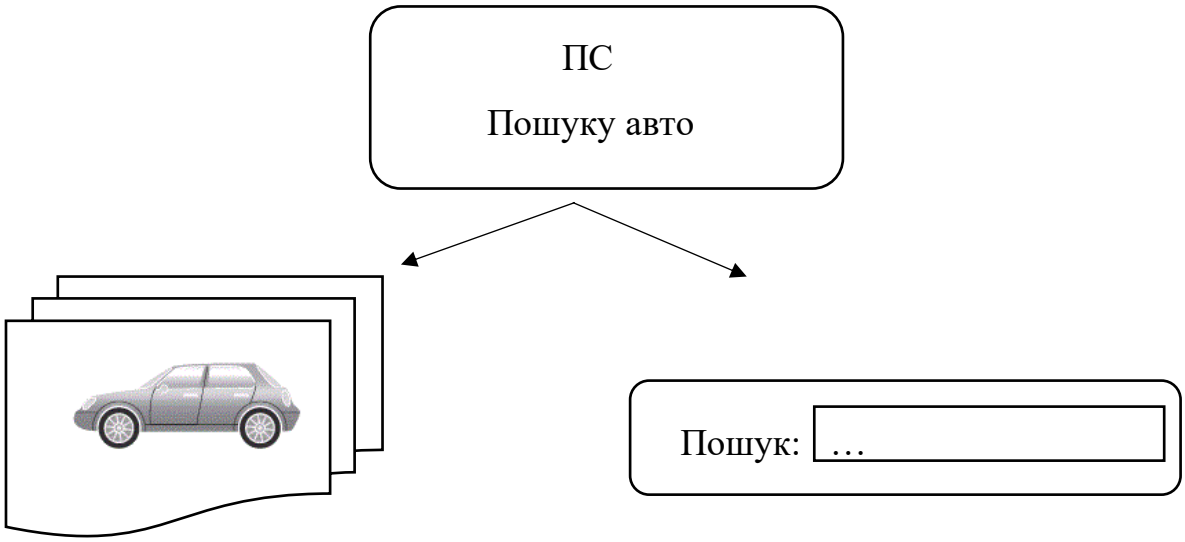

Рисунок 1.8 — Реалізація підсистеми пошуку авто

1.4.4 Підсистема підбору авто

Дана підсистема створена для продвинутого користувача, який розуміється на конструкції та технічних характеристиках автомобілів, та взмозі сам задати усі необідні параметри, що його цікавлять, та отримує на виході декілька варіантів автомобілів, які підходять під задані умови. Процес задання параметрів відбувається вручну, користувач обирає один із можливих варіантів або ж вводить відповідні дані в строки. Список технічних характеристик включає в себе такі пункти: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто (рисунок 1.9).

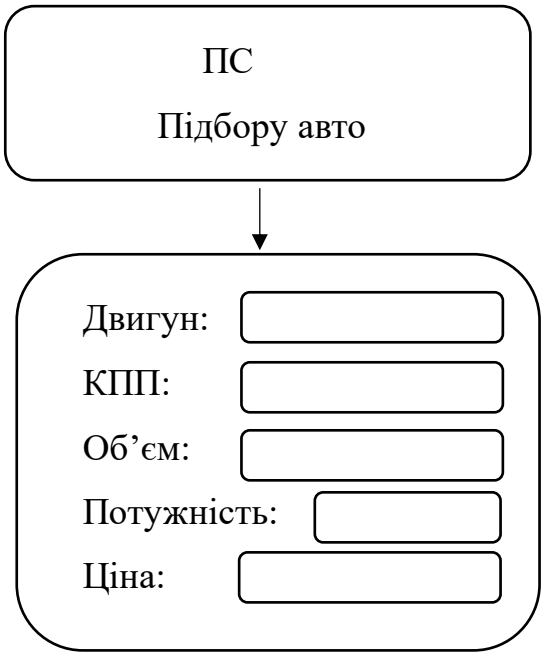

Рисунок 1.9 — Реалізація підсистеми підбору авто

1.4.5 Підсистема виведення даних

Дана підсистема створена для аналізу обраних з запропонованого списку або ж підібраних вручну за технічними характеристиками автомобілів, та

створення форми виведення їх користувачеві. Залежно від заданих параметрів, на виході буде отримано одне або ж декілька авто, що задовольняють умови пошуку. Після цього наступає фінальний етап — перехід до замовлення обраного автомобіля шляхом введення особистих даних, які будуть збережені на сервер. Для цього користувач залишає свої особисті дані, а саме: ПІБ, контактний номер телефону, електронну пошту, серію та номер паспорту, ідентифікаційний код платника податків (рисунок 1.10)

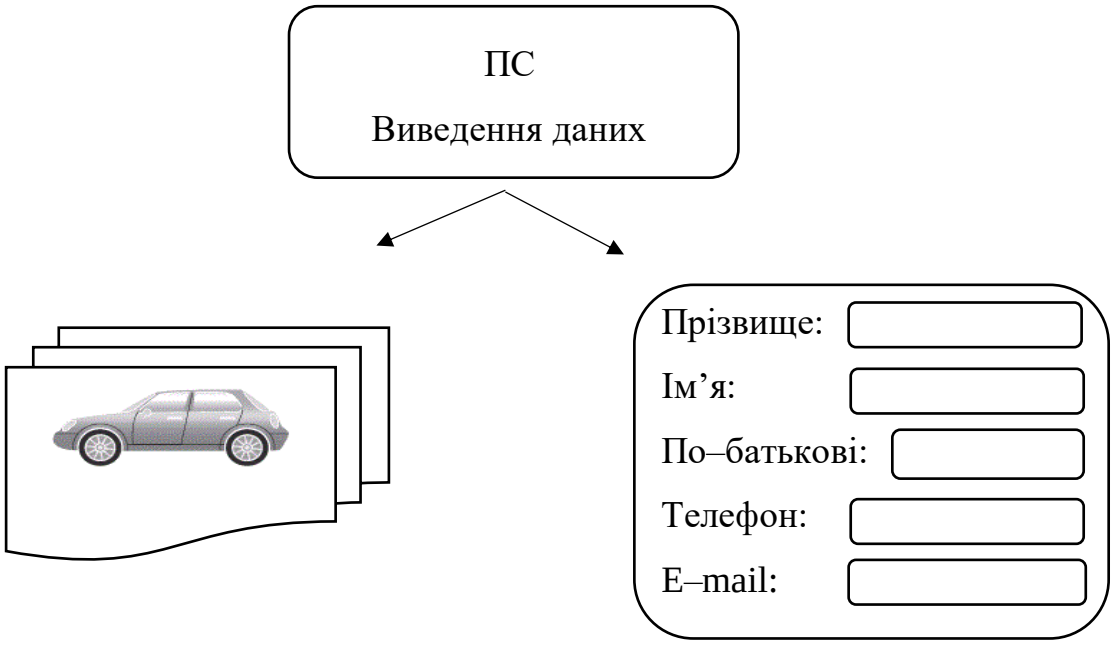

Рисунок 1.10 — Реалізація підсистеми виведення даних

1.4.6 Підсистема виходу

Дана підсистема створена для коректного завершення сеансу роботи з інформаційною системою підбору та замовлення авто. Для адміністратора (Admin) цей процес включає збереження усіх внесених змін та опрацьованих даних, перевірка на їх цілісність, для користувача (User) даний процес включає збереження усіх підібраних авто, їх параметрів та вихід з особистого кабінету (рисунок 1.11).

Рисунок 1.11 — Реалізація підсистеми виходу

1.5 Висновки за розділом

На сьогоднішній день web–сайт ― доволі необхідна річ, що має велику кількість плюсів. Основний з них ― економія часу на пошуках необхідної інформації, оскільки відпадає необхідність у тривалих і втомливих походах до офісів різноманітних компаній, що надають певні послуги, чи ж пошуку необхідного матеріалу на різноманітних інформаційних сайтах.

В теоретичній частині описано стандартні приклади того як потрібно використовувати сучасні веб–технології для реалізації того чи іншого програмного веб–продукту. У даному випадку це web–сайт компанії “USAuto”, який за своєю будовою складається з таких веб–технологій, як HTML5, CSS3, JavaScript, Node.js, SQL, MongoDB. Було проведено аналіз та вивчено технології створення такого електронного ресурсу як web–сайт компанії по наданню послуг. Для того, щоб створити даний продукт, необхідно володіти знаннями, описаними вище.

2 WEB – ОРІЄНТОВАНИЙ ПРОГРАМНИЙ ЗАСІБ ДЛЯ ОРГАНІЗАЦІЇ ТА СУПРОВОДЖЕННЯ ПРОЦЕСУ ПІДБОРУ І ЗАМОВЛЕННЯ АВТОМОБІЛІВ

2.1 Узагальнений технологічний ланцюжок роботи інформаційної системи підбору та замовлення авто

Технологічний ланцюжок роботи інформаційної системи являє собою впорядковану послідовність взаємопов'язаних дій та операцій, що виконуються над початковими даними до отримання необхідного результату. Інформаційна система підбору та замовлення авто складається з ряду основних компонентів, кожен з яких відповідає за певний процес її роботи.

Авторизація в системі можлива двох типів: адміністратор (admin) та користувач (user). Перший тип має права для редагування даних інформаційної системи, доступ до створення нових та видалення старих елементів системи, доступ до БД. Другий тип має можливість користуватись пошуком та підбором автомобілів за усіма існуючими параметрами, зберігати обрані позиції в особистому кабінеті та отримувати повідомлення про усі їх зміни [11].

Процес пошуку авто може відбуватись вручну, а саме переглядом представлених авто зі списку та переходом до їх детального технічного опису, або ж автоматично, за допомогою введення ключових символів ідентифікаторів. До кожного з представлених користувачеві автомобілів у списку буде прикріплене його фото, пошкодження, детально описані всі його технічні характеристики. Процес підбору авто передбачає задання параметрів відбувається вручну, користувач обирає один із можливих варіантів або ж вводить відповідні дані в строки. Список технічних характеристик включає в себе такі пункти: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та

його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто.

Процес виведення інформації та збереження даних напряму залежить від заданих користувачем параметрів, на виході буде отримано одне або ж декілька авто, що задовольняють умови пошуку. Після цього наступає фінальний етап – перехід до замовлення обраного автомобіля шляхом введення особистих даних, які будуть збережені на сервер. Для цього користувач залишає свої особисті дані, а саме: ПІБ, контактний номер телефону, електронну пошту, серію та номер паспорту, ідентифікаційний код платника податків. Для адміністратора (Admin) процес виходу з системи включає в себе збереження усіх внесених змін та опрацьованих даних, перевірка на їх цілісність, для користувача (User) даний процес включає збереження усіх підібраних авто, їх параметрів та вихід з особистого кабінету (рисунок 2.1) .

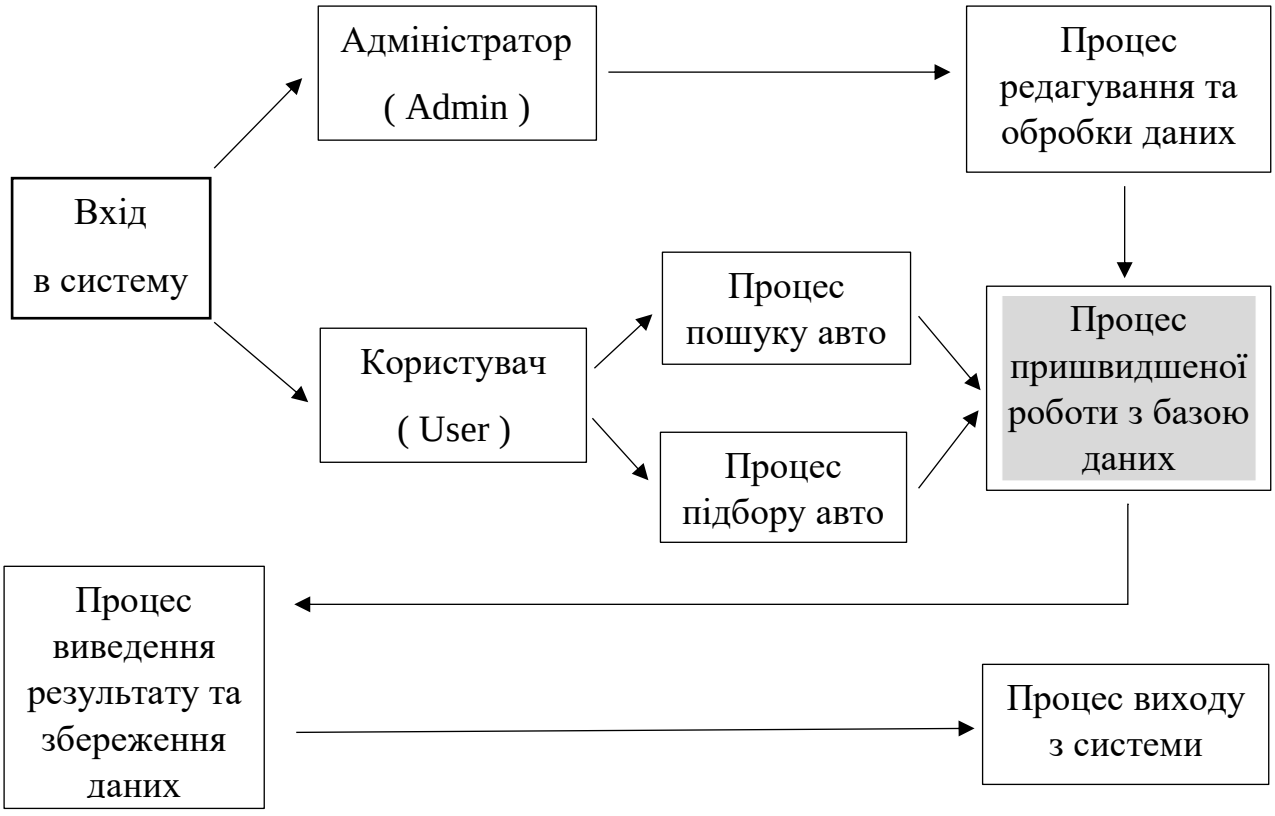

Рисунок 2.1 — Узагальнений технологічний ланцюжок інформаційної системи

Підсистема логін/пароль створена для авторизації користувачів та входження їх до персонального кабінету. При введенні не коректних даних або ж спробі ідентифікації без введеного одного з обов'язкових полів, відбудеться виведення інформуючого повідомлення з відповідною причиною відмови в авторизації користувача. Дана підсистема має два основних типи доступу: адміністратор (admin) та користувач (user). Перший тип має права для редагування даних інформаційної системи, доступ до створення нових та видалення старих елементів системи, доступ до бази даних, в якій зберігається вся інформація про користувачів. Другий тип має можливість користуватись пошуком та підбором автомобілів за усіма існуючими параметрами, зберігати обрані позиції в особистому кабінеті та отримувати повідомлення про усі їх зміни (рисунок 2.2).

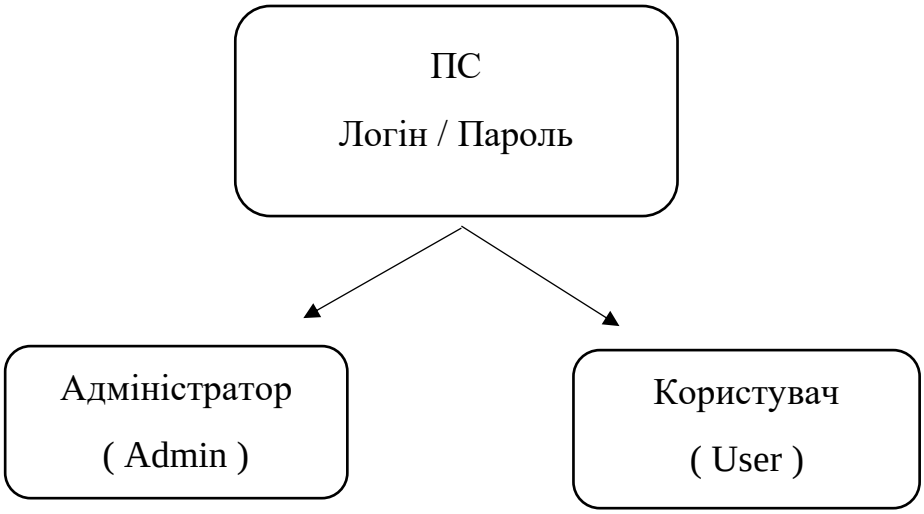

Рисунок 2.2 — Реалізація підсистеми логін/пароль

Підсистема первинного огляду даних має ознайомчий характер з структурою інформаційного ресурсу, перевагами перед конкурентами, його можливостями та даними, якими він оперує. Користувачеві надається можливість ознайомитись з усіма технічними параметрами, що застосовуються з метою підбору та пошуку авто та зворотнім зв'язком з адміністрацією даної інформаційної системи (рисунок 2.3).

В підсистемі пошуку авто реалізований широкий вибір різних варіантів автомобілів, які запропоновані для вибору потенційним клієнтом. Процес

пошуку може відбуватись вручну, а саме переглядом представлених авто зі списку та переходом до їх детального технічного опису, або ж автоматично, за допомогою введення ключових символів ідентифікаторів.

Рисунок 2.3 — Реалізація підсистеми первинного огляду даних.

До кожного з представлених користувачеві автомобілів у списку буде прикріплене його фото, пошкодження, детально описані всі його технічні характеристики (рисунок 2.4).

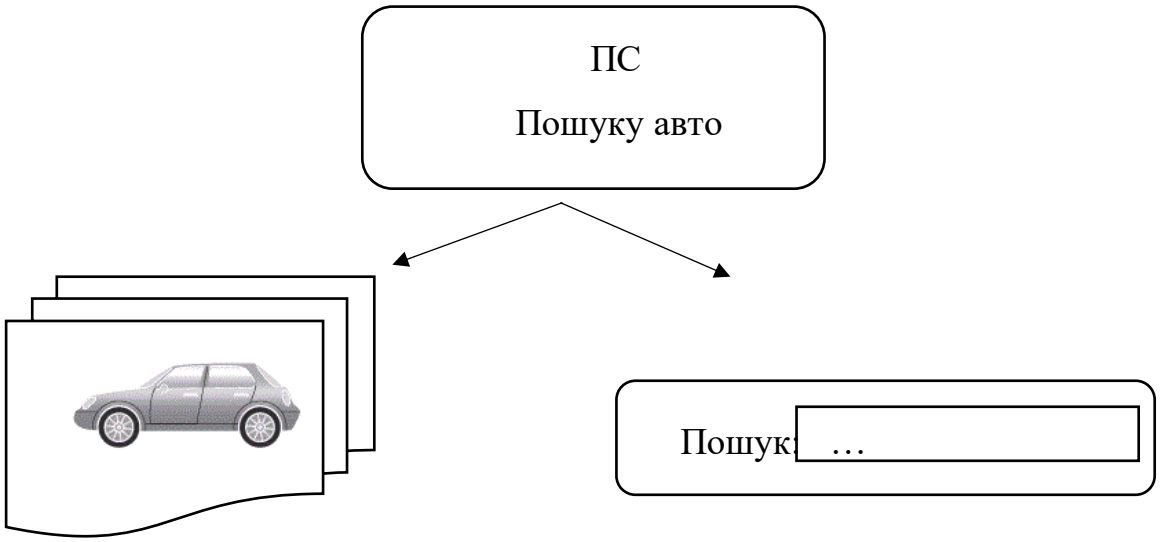

Рисунок 2.4 — Реалізація підсистеми пошуку авто

Підсистема підбору авто створена для продвинутого користувача, який розуміється на конструкції та технічних характеристиках автомобілів, та взмозі сам задати усі необідні параметри, що його цікавлять, та отримує на виході декілька варіантів автомобілів, які підходять під задані умови. Процес задання

параметрів відбувається вручну, користувач обирає один із можливих варіантів або ж вводить відповідні дані в строки. Список технічних характеристик включає в себе такі пункти: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто (рисунок 2.5).

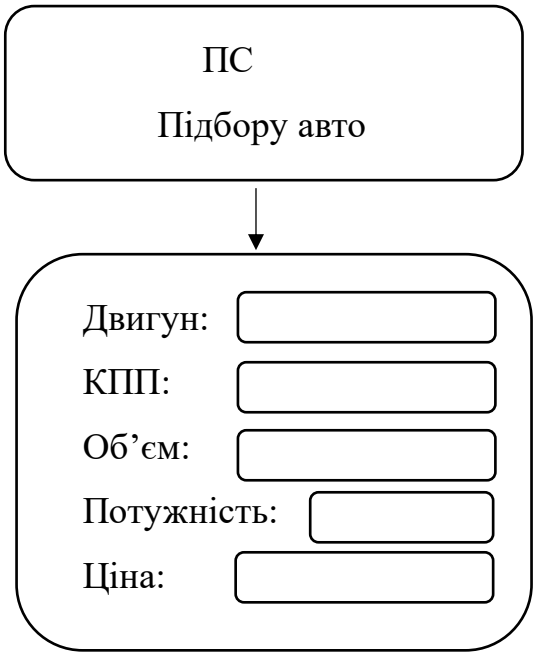

Рисунок 2.5 — Реалізація підсистеми підбору авто

Підсистема виведення даних створена для аналізу обраних з запропонованого списку або ж підібраних вручну за технічними характеристиками автомобілів, та створення форми виведення їх користувачеві. Залежно від заданих параметрів, на виході буде отримано одне або ж декілька авто, що задовольняють умови пошуку. Після цього здійснюється перехід на фінальний етап — перехід до замовлення обраного автомобіля шляхом введення особистих даних, які будуть збережені на сервер. Для цього користувач залишає свої особисті дані, а саме: ПІБ, контактний номер телефону, електронну пошту, серію та номер паспорту, ідентифікаційний код платника податків (рисунок 2.6)

Підсистема виходу створена для коректного завершення сеансу роботи з інформаційною системою підбору та замовлення авто. Для адміністратора (Admin) цей процес включає збереження усіх внесених змін та опрацьованих даних, перевірка на їх цілісність, для користувача (User) даний процес включає збереження усіх підібраних авто, їх параметрів та вихід з особистого кабінету (рисунок 2.7).

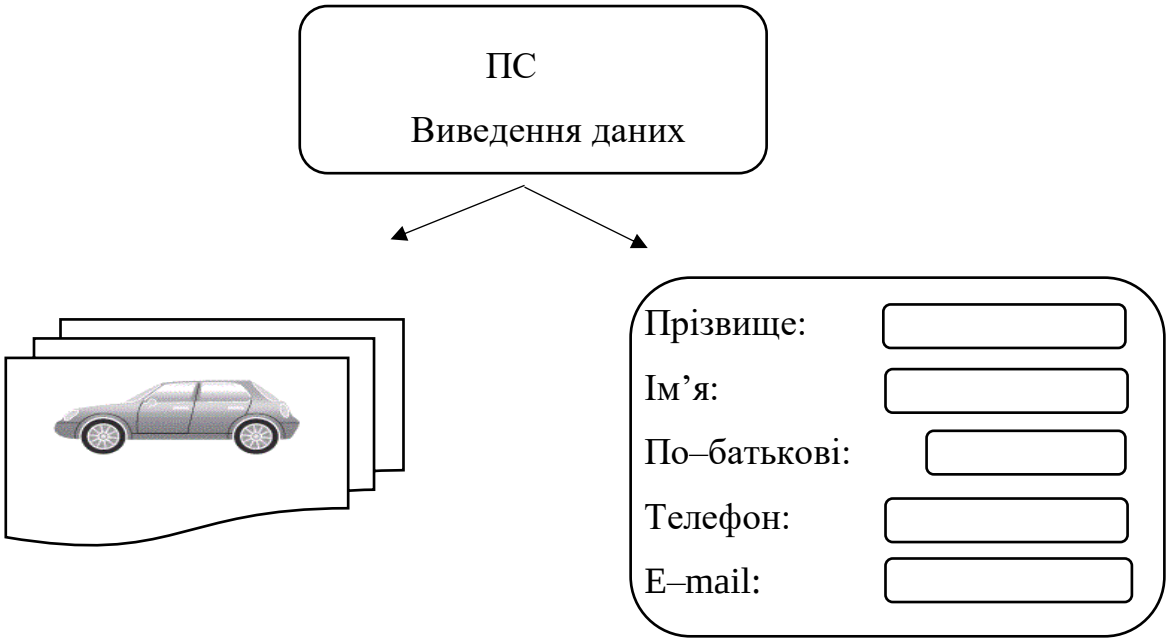

Рисунок 2.6 — Реалізація підсистеми виведення даних

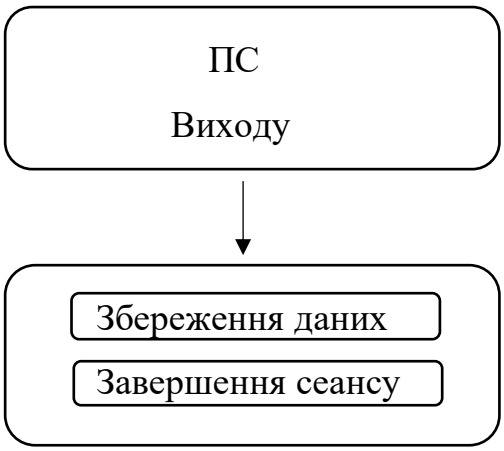

Рисунок 2.7 — Реалізація підсистеми виходу

2.2 Загальна структура програмного засобу

Перед початком розробки сайту першопочатково формується його структура, де детально прораховано та сформовано кількість сторінок, логічний зв’язок між ними. Існує 3 загальноприйнятих структри, а саме: ієрархічна, лінійна та довільна структури.

Лінійна структура побудови веб–сайту використовується при необхідності послідовної подачі інформацію, де один блок інормацію плавно переходить у інший, пов’язаний логічним зв’язком із попереднім. Дана структура характерна переходом з однієї сторінки на іншу за посиланням. В деяких випадках, з метою створення навігації по інфораційному ресурсу до сторінки також прикріплюють посилання й на попередню сторінку також [12].

Для ієрархічної структури побудови даних притаманна лише одна сторінка (головна), що не має переходів до передніх сторінок, а кожна сторінка, на яку є посилання з головної, має зворотнє посилання на основну сторінку веб–сайте. Основною перевагою ієрархічної структури є можливість розмістити не обмежену кількість посилань на довільну кількість сторінок веб–ресурсу. Дана структура підходить для сайтів з різною тематикою наповнення.

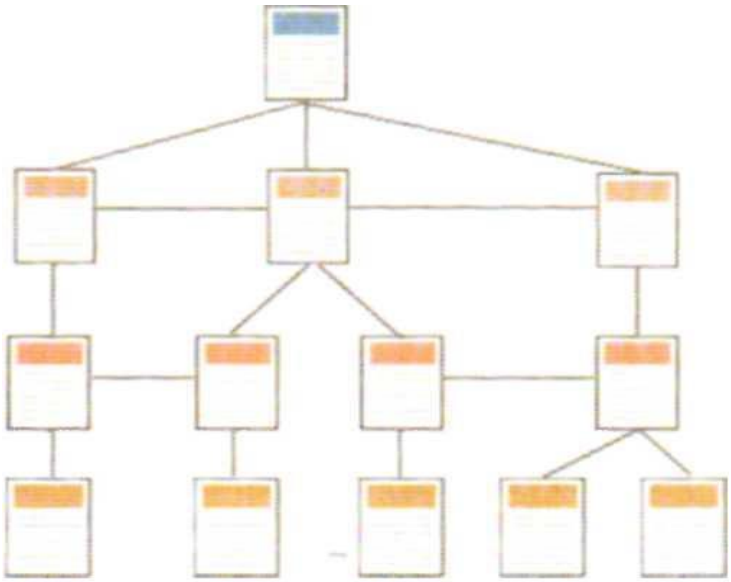

Рисунок 2.8 — Схема структури веб–сайту

У випадку з інформаційною системою підбору та замовлення авто використано ієрархічну структуру. Так як на даному сайті розміщуються матеріали різного характеру, тому створено меню з посиланнями на їх сторінки (рисунок 2.9)

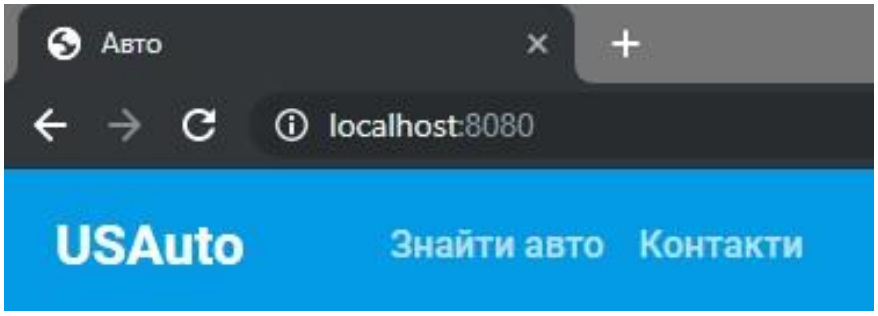

Рисунок 2.9 — Перелік пунктів меню

На головній сторінці даного ресурсу розміщена коротка інформація про те, чому потенційний клієнт має обрати саме нашу компанію, чим ми кращі за наших конкурентів (рисунок 2.10)

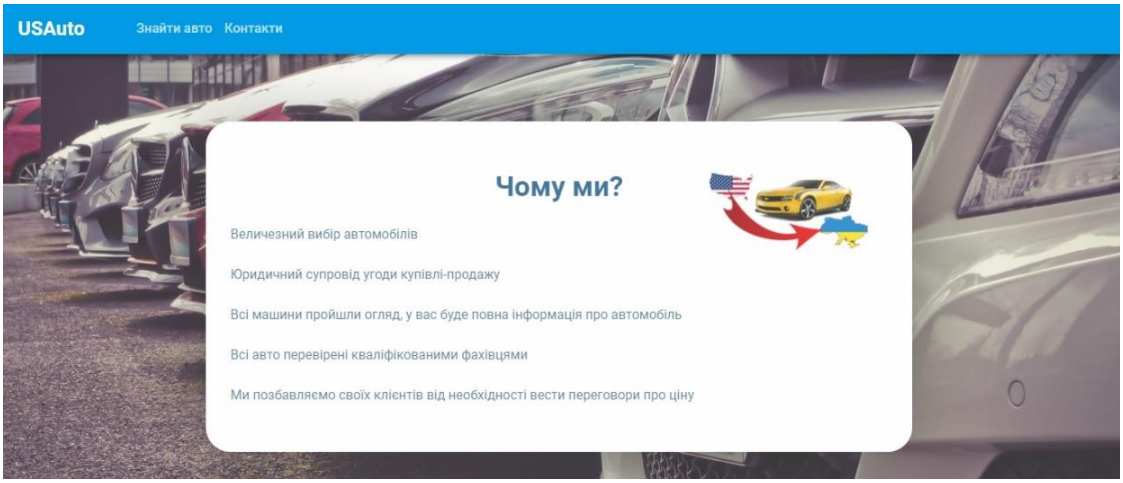

Рисунок 2.10 — Головна сторінка сайту

Інформаційну систему реалізовано таким чином, що процес пошуку та підбору може відбуватись вручну, а саме переглядом представлених авто зі списку та переходом до їх детального технічного опису, або ж автоматично, за

допомогою введення ключових параметрів та характеристик транспортного засобу (рисунок 2.11).

Для детального ознайомлення з певним автомобілем достатньо натиснути на клавіше “Перегляд”, в результаті чого буде виконано перехід на сторінку даного авто з його детальними технічними характеристиками (рисунок 2.12).

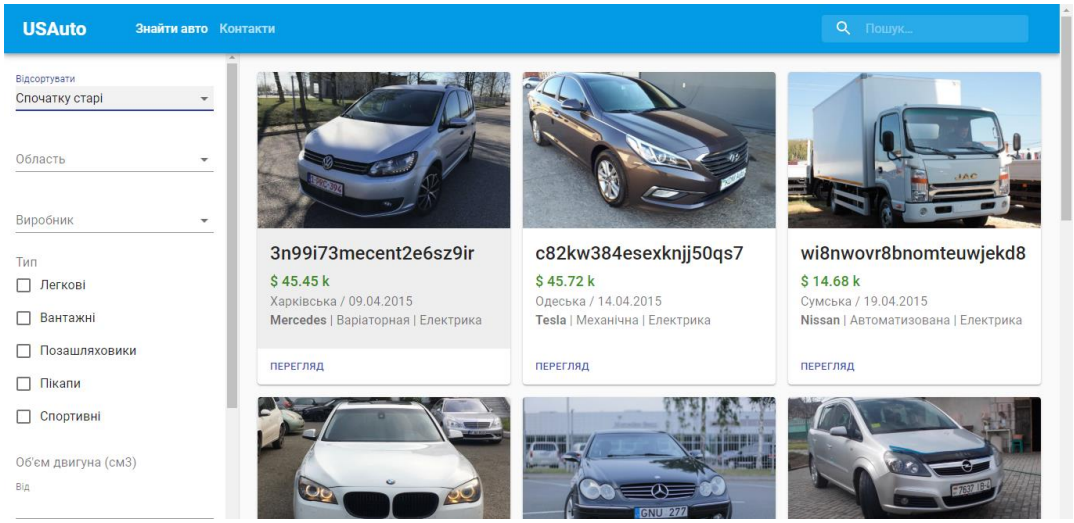

Рисунок 2.11 — Сторінка підбору та пошуку авто

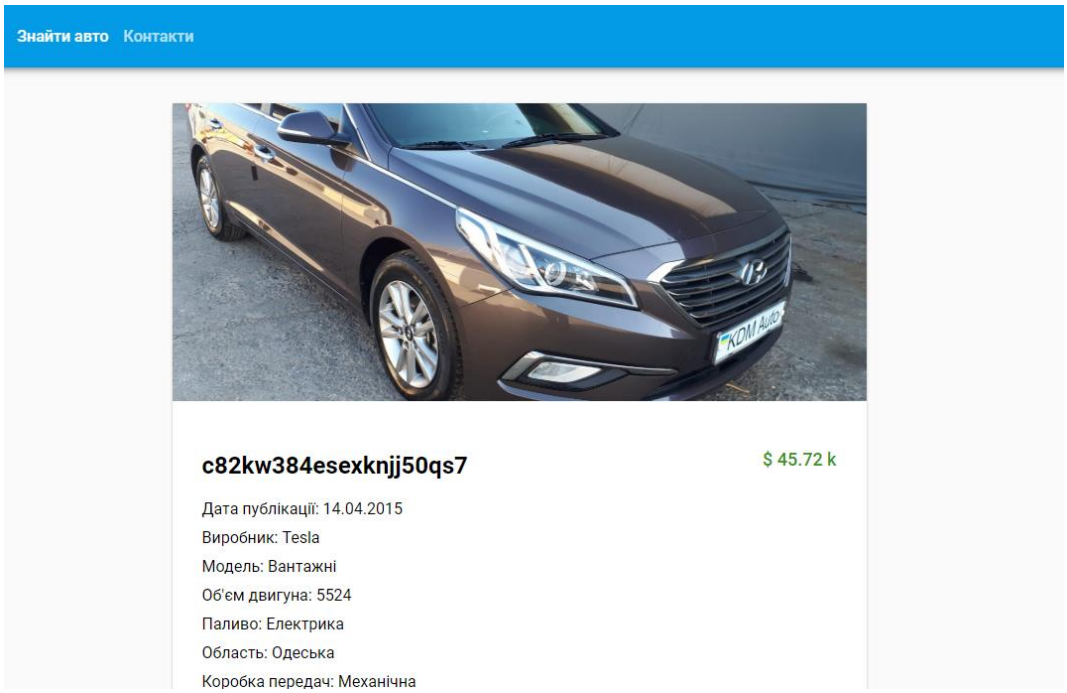

Рисунок 2.12 — Детальна інформація по авто

Після вибору авто та при бажанні замовити його, користувач переходить на сторінку контактних даних, де вводить свої особисті дані, що в подальшому зберігаються в базі даних (рисунок 2.13).

В підборі кольорової гамми для інформаційної системи підбору та пошуку авто, вибір був спрямований більше до кольорів світлої та яскравої гамми, щоб колір погружав людину в легкість.

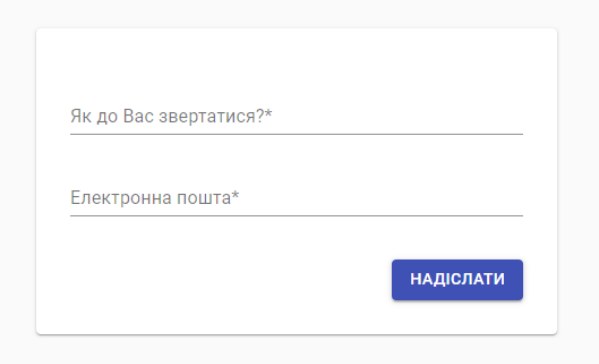

Рисунок 2.13 — Поля вводу особистих даних

Підшукавши певну кольорову палітру (рисунок 2.14), тому що не всі кольори імпонують один одному і потрібно вміти підбирати кольори, які зможуть доповнювати один одного та придавати сайту гарного вигляду. Так як світлі кольори асоціюються з гармонією, добротою, енергією та просто змушує людину залишаться повністю розслабленою, що викликає певну цікавість у перегляді інформаційних сторінок.

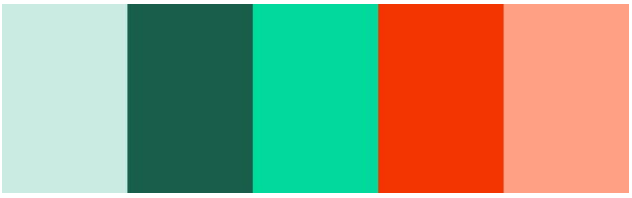

Рисунок 2.14 — Кольорова гама інформаційної системи

2.3 Алгоритм роботи та реалізація підсистеми пошуку авто

В даній підсистемі реалізований широкий вибір різних варіантів автомобілів, які запропоновані для вибору потенційним клієнтом. Процес

пошуку може відбуватись вручну, а саме переглядом представлених авто зі списку та переходом до їх детального технічного опису, або ж автоматично, за допомогою введення ключових символів ідентифікаторів (рисунок 2.15).

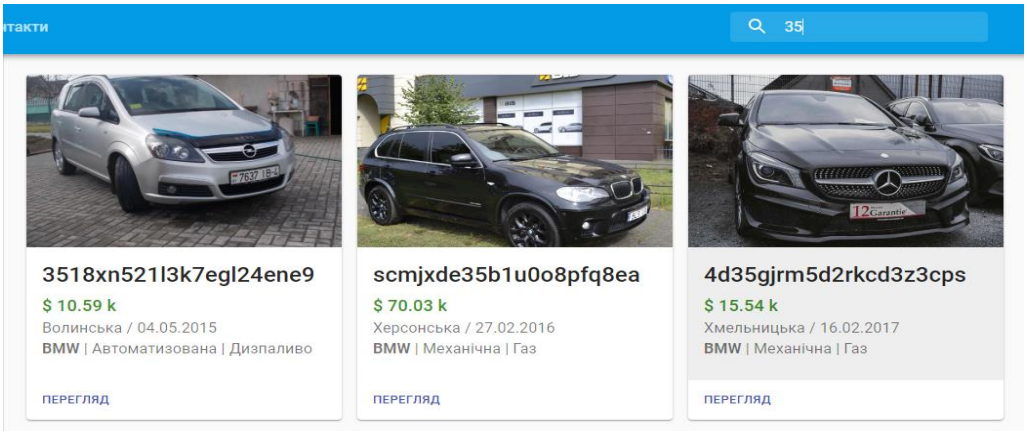

Рисунок 2.15 — Реалізація пошуку авто в інформаційній системі

Рисунок 2.16 — Алгоритм роботи підсистеми пошуку авто

Дану підсистему пошуку було створено з використанням HTML5 та CSS3 для створення графічного оформлення та позиціонування елементів на сторінці, та JavaScript для функціонування пошуку за частковою назвою ідентифікатора автомобіля.

Лістинг фрагменту коду даної підсистеми:

```
routes.get("/api/cars/:id", async (req, res) => { try {
const car = await Car.findById(req.params.id).populate(["company", "fuel", "region", "transmission", "type"])
 res.status(200).json(car)
 } catch (err) {
console.log(err)
res.status(404).json({ message: "Not found" })}
})
```

2.4 Алгоритм роботи та реалізація підсистеми підбору авто

Дана підсистема створена для продвинутого користувача, який розуміється на конструкції та технічних характеристиках автомобілів, та здатний самотійно задати усі необідні параметри, що його цікавлять, та отримує на виході декілька варіантів автомобілів, які підходять під задані умови. Процес задання параметрів відбувається вручну, користувач обирає один із можливих варіантів або ж вводить відповідні дані в строки. Список технічних характеристик включає в себе такі пункти: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто (рисунок 2.17).

Дану підсистему підбору було створено з використанням технологій HTML5 та CSS3 для створення графічного оформлення та позиціонування

елементів на сторінці, та JavaScript для функціонування підбору за технічними характеристиками авто.

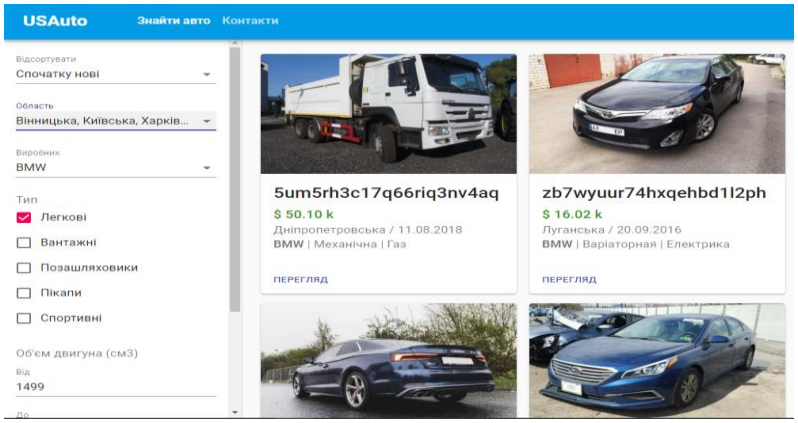

Рисунок 2.17 — Реалізація підбору авто в інформаційній системі

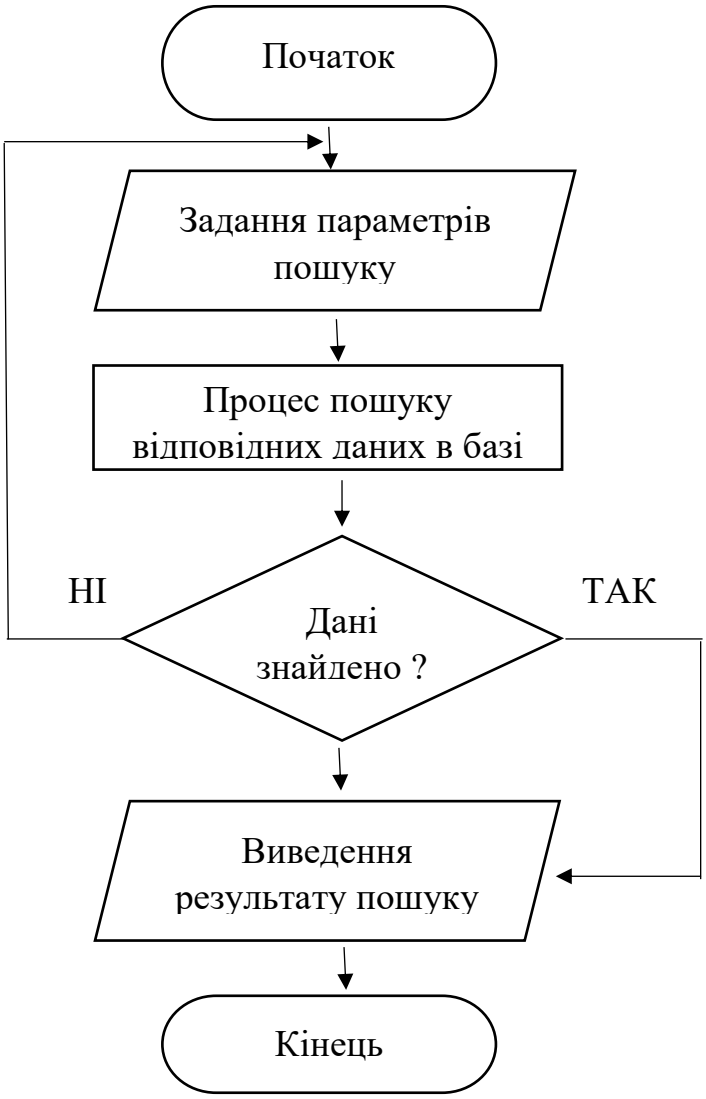

Рисунок 2.18 — Алгоритм роботи підсистеми підбору авто

Дані інформаційної системи зберігаються в базі даних mongoDB, а зв'язок з базою виконаний з допомогою програмної платформи Node.js.

Лістинг фрагменту коду даної підсистеми:

```
routes.get("/api/cars", async (req, res) => {
const query = {}
const keys = Object.keys(req.query)
keys.forEach(param => { if (
param !== "page" &&
param !== "limit" &&
param !== "sort" &&
param !== "search" &&
param !== "minCapacity" &&
param !== "maxCapacity") {
const value = req.query[param]
if (value) {
query[param] = { $in: value }}})
```

2.5 Алгоритм роботи та реалізація підсистеми виведення даних

Дана підсистема створена для аналізу обраних з запропонованого списку або ж підібраних вручну за технічними характеристиками автомобілів, та створення форми виведення їх користувачеві. Залежно від заданих параметрів, на виході буде отримано одне або ж декілька авто, що задовольняють умови пошуку (рисунок 2.19)

Після цього наступає фінальний етап — перехід до замовлення обраного автомобіля шляхом введення особистих даних, які будуть збережені на сервер (рисунок 2.20). Для цього користувач залишає свої особисті дані, а саме: ПІБ, контактний номер телефону, електронну пошту, серію та номер паспорту, ідентифікаційний код платника податків (рисунок 2.21).

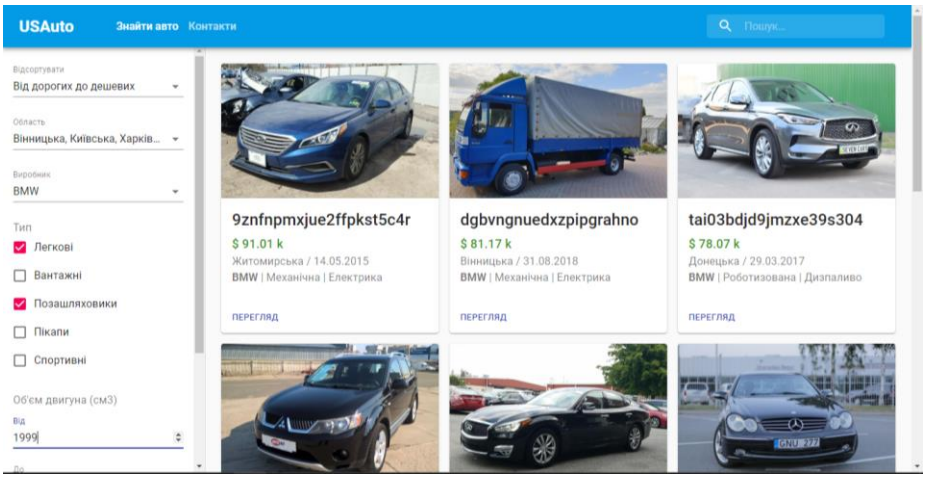

Рисунок 2.19 — Результат роботи параметрів підбору авто

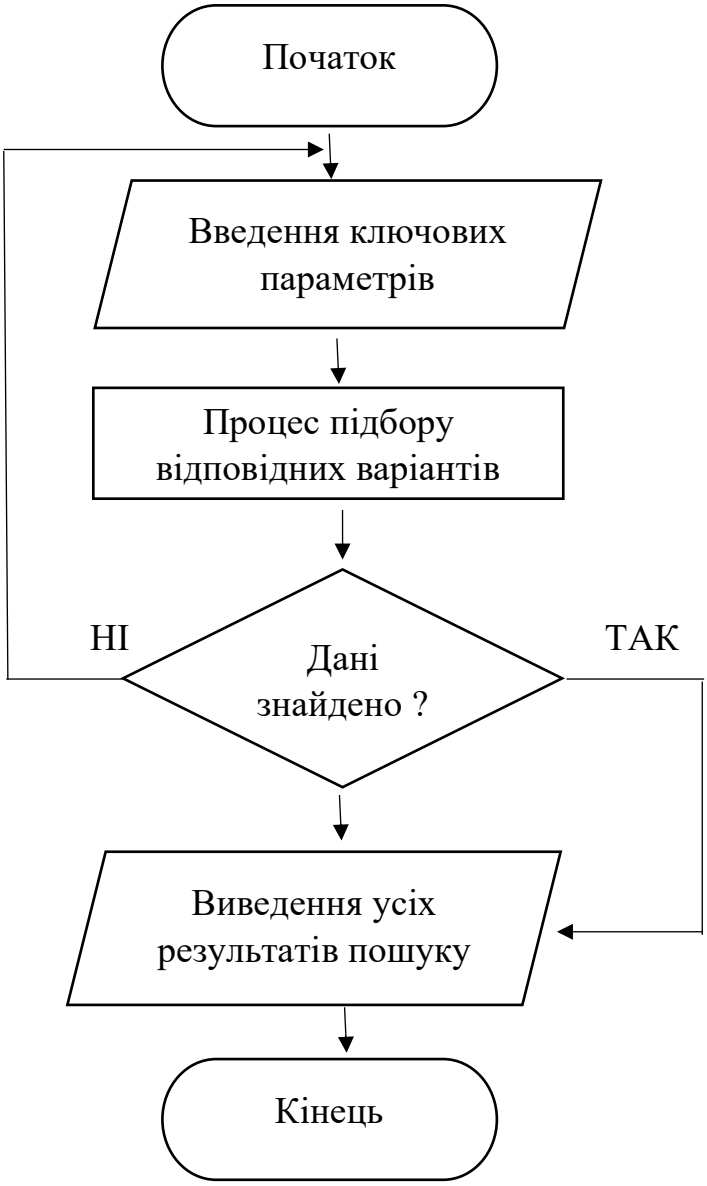

Рисунок 2.20 — Алгоритм роботи підсистеми виведення даних

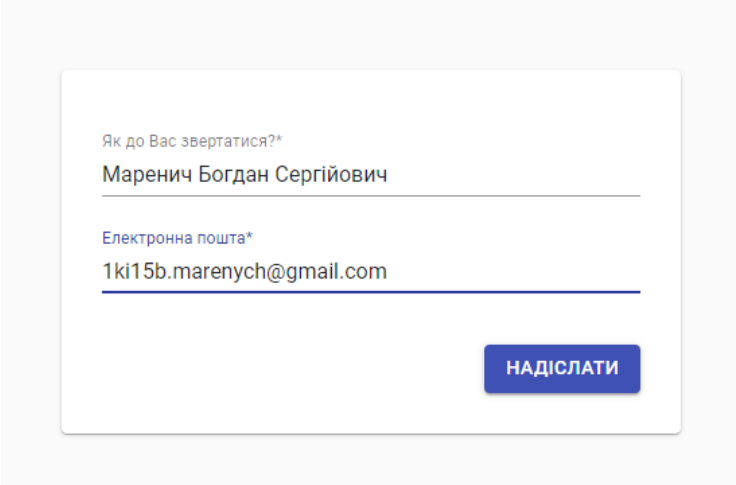

Рисунок 2.21 — Реалізація підсистеми збереження даних в базу

Дану підсистему було створено з використанням HTML5 та CSS3 для створення графічного оформлення та позиціонування елементів на сторінці, Node.js для звернення до даних сервера, які збережені в базі даних mongoDB (рисунок 2.22).

index.js - cars - Visual Studio Code

index.js ...\Contacts | Routing.js | express.js | server.js | index.js ...\routes

```
        <div style={{ fontSize: 16, color: "red", marginTop: 30 }}>{error}</div>

        <div style={{ display: "flex", justifyContent: "flex-end", marginTop: 30 }}>
            <Button variant="contained" color="primary" onClick={this.submit}>
                Надіслати
            </Button>
        </div>
    </Card>

    <Modal
        aria-labelledby="simple-modal-title"
        aria-describedby="simple-modal-description"
        open={open}
        onClose={this.handleClose}
    >
        <div style={styleOfModal}>
            <Typography variant="h6" id="modal-title">
                Заявка створена
            </Typography>
            <Typography variant="subtitle1" id="simple-modal-description">
                Протягом дня з вами зв'яжеться наш оператор
            </Typography>
        </div>
    </Modal>
</div>
```

Рисунок 2.22 — Програмна реалізація методу збереження даних

Введені в поля дані відправляються на сервер та зберігаються в базі даних mongoDB, побачити результат збереження даних можливо використавши програму Robo 3T (рисунок 2.23).

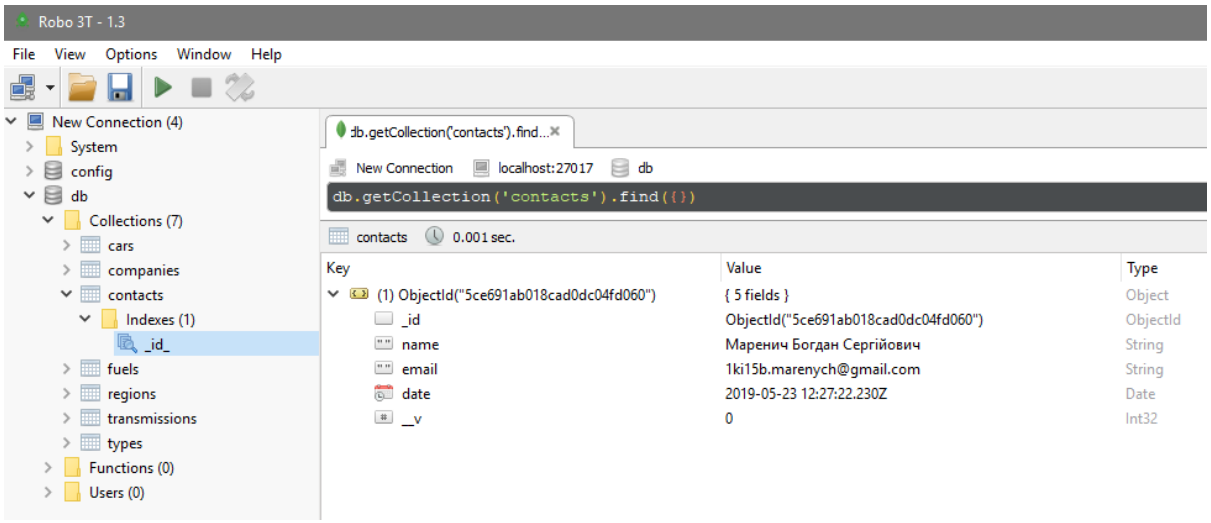

Рисунок 2.23 — Збереження введених даних у базі mongoDB

2.6 Процес пришвидшеної роботи з базою даних

Для максимального збільшення швидкодії інформаційної системи було прийнято рішення відмовитись від використання застарілих технологій та бази даних SQL. Перехід до більш сучасної MongoDB надає можливість пришвидшити роботу з базою даних на 25%, що суттєво покращує відклик серверу на запити, підвищує стабільність у роботі, зменшує час затримок у роботі інформаційної системи при суттєвих навантаженнях (рисунок 2.24).

MongoDB — система керування базами даних, яка має у собі документо–орієнтований підхід (СКБД), перевагою якої є відкритий вихідним код, що не залежна від розпису та створення схеми таблиць. MongoDB позиціонується як швидка та масштабована система, дані у якій знаходяться у форматі ключ/значення, а також реляційними СКБД, які є більш зручними та зрозумілими для програміста у формуванні запитів. Код даної нереляційної бази даних MongoDB створений на мові C++ та розповсюджюється в рамках ліцензії AGPLv3.

MongoDB за основу має зберігання документів в JSON – подібному форматі, де мова для формуванян запитів є доволі гнучкою, а також створює індекси для збереження атрибутів, без будь–яких затруднень надає можливість збергіання великих бінарних об’єктів, проводить статистику та веде облік внесени даних до БД, здійснює свою роботу відповідно до парадигми Map/Reduce, має реплікацію та створення можливих відмовостійких конфігурацій. У MongoDB вбудовані засоби набору ресурсів по серверах, за основу даної процедури взяти прототип певного ключа [13].

Рисунок 2.24 — Результат порівняння швидкодії баз даних

2.7 Висновки за розділом

В результаті виконання даної проектної частини було описано технологічний ланцюжок створення інформаційної системи та реалізовано інформаційний web–сайт, мета якого полягає у наданні корисної інформації та допомозі з підбором та пошуком обраного автомобіля.

Використання високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій за рахунок

процесу оптимізації пошуку даних дозволив пришвидшити роботу інформаційної системи на 25%.

Розроблена інформаційна система підбору та замовлення авто розміщена в мережі Internet. Наразі є актуальною серед користувачів, проблем у функціонуванні та програмних збоїв немає.

3 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Поняття якості програмного забезпечення та процесів тестування програмних засобів

Якість програмного забезпечення – множина характерних аспектів, що дають оцінку можливості надати готовий програмний продукт згідно запиту замовника. Відповідно до міжнародних стандартів оцінки рівня якості відокремлюють два основні процеси організації якості протягом усього функціонування програмного забезпечення:

— гарантування надійності ПЗ;

— інженерія якісних показників ПЗ.

Якість програмного забезпечення ― відносна властивість програмного забезпечення, пропрційне відношення виконання роботи до стандартизованих вимог. Якість ПЗ ― комплекс характеристик програмного продукту, що зумовлюють його можливість задовольнити регламентовані або прогнозовані потреби клієнта. Поняття якості дуже відносне та має різноманітні варіації залежно від вимог, поставлених до програмного комплексу.

Внутрішня атрибутика та характеристика ПЗ використовується для планування виконання потрібних загальних характеристик належності продукту. Для квантифікації загалних положень якості використовують внутрішні метрики, як інструмент відповідності якісних характеристик проміжних продуктів до загальнопоставлених умов якості, які створюються на процесах, що базуються перед проходженням тестування.

Зовнішні і внутрішні параметри якості детально позиціонують властивості самого ПЗ, а також ставлення клієнта і виконавця на дане ПЗ. Безпосереднього фінального користувача ПЗ приваблює експлуатаційна надійність ПЗ – спільний ефект від рівня якісних показників, що визначається

терміном результативності, а не особливістю самого ПЗ. Дане трактування ширше, ніж будь–яка одиничка характеристика.

Моделі базуються на різній чисельності рівнів і абсолютно або частково співпадають щодо набору характеристик надійності.

Визначення якісних характеристик та атрибутів відповідно до зально прийнятиї норм стандартизації ISO 9126:2001:

— функціональність (functionality). Можливість ПЗ в деяких умовах приймати рішення задачі, необхідні кінцевому користувачеві. Визначає, яку саме дію буде виконувати ПЗ, які завдання воно приймає до вирішення, а саме: функціональна приналежність (suitability), коректність та точність (accuracy), модливість до взаємодії (interoperability), відповідність правилас і стандартам (compliance), конфіденційність (security);

— надійність (reliability). Можливість ПЗ тримати на рівні визначену необхідну працездатність у поставлених умовах, а саме: завершеність (maturity), витривалість до відмов (fault tolerance), можливість реінкарнації (recoverability), належність до стандартів надійності (reliability compliance), комфортабельність використання (usability), практичність;

— можливість ПЗ бути комфортним у розумінні, сприйнятті та використанні, а також візуально привабливим для кінцевого користувача, а саме: зрозумілість (understandability), зручність вивчення (learnability), комфортабельність використання (operability), візуальна привабливість (attractiveness), належність до стандартів кофортабельного використання (usability compliance);

— продуктивність (efficiency), ефективність. Можливість ПЗ при встановлених вимогах забезпечувати достатню працездатність відносно наданого для виконання ресурсу. Можна визначити ресурси всіх типів, а саме: ефективність в часі (time behaviour), ефективність раціонального розподілення ресурсів (resource utilisation), належність до стандартів продуктивності (efficiency compliance);

— комфортабельність супроводу (maintainability). Можливість зрозумілого проведення всіх видів праці, пов'язаних із забезпеченням підтримки програм, а саме: аналізованість (analyzability), можливість внесення корегувань (changeability), стабільність (stability), доступність перевірки (testability), належніість до стандартів зручності супроводу (maintainability compliance);

— переносимість (portability). Можливість ПЗ утримувати працездатність при транспортуванні з одного середовища у інше, враховуючи організаційні, програмні й апаратні аспекти, а саме: адаптованість (adaptability), доступність установки (installability), можливість до співіснування (coexistence), можливість заміни (replaceability), належність до стандартів переносимості (portability compliance);

Тестування програмного забезпечення (Software Testing) – технічний процес вивлення інформації про аспекти якості програмного продукту відносно контексту, в якому він має функціонувати (використовуватись). Техніка проведення тестування включає не лише процес аналізу, пошуку помилок та дефектів, а також і тестування окремих програмних компонентів з метою оцінки коректної працездатності.

Зважаючи на чисельність різноманітних тестів навіть для нескладних програмних компонентів, стратегія тестування полягає в тому, щоб здійснити всі існуючі тести з врахуванням наданого часу та ресурсу. У результаті перевірок програмне забезпечення (ПЗ) перевіряється загальним виконанням програми з метою прояву усіх несправностей та перебоїв у роботі, баґів.

Тестування рекомендовано паралельно із створеним фрагментом виконуваного коду. Для наглядного прикладу, при поетапному виконанні процесу, основні тести відбуваються після встановлення системних вимог і тоді вони адаптовані в тестових програмах. На противагу цьому, згідно до вимог гнучкості ПЗ, програмування та тестування розробленого засобу чи його компоненту відбувається паралельно.

3.2 Методи та засоби тестування

Під час виконання тестування інформаційний ресурс перевіряється на коректність та вимоги, поставлені до нього у технічному завданню, контролюється виконання його технічних характеристик. В залежності від поставлених у технічному завданні до ресурсу вимог може здійснюватиь ряд таких перевірок :

— перегляд веб–сайту на монітрах з різною роздільною здатністю екрану;

— перегляд кросбраузерності сайту в різних браузерах;

— перевірка оптимального часу підтягування сайту при різній швидкості інтернету ;

— перевірка коректності гіперпосилань;

— перевірка коректної передачі кольорової гами на сторінках з різними налаштуваннями та типами екранів;

— перевірка передачі анімацій, шрифтів, інтервалів;

— перевірка характеристик кожного елементу сайту сайту;

— перевірка на відповідність до технічкого завдання змісту кожної з сторінок розробленого онлайн ресурсу.

Тестування веб–систем це аналіз та співвідношення реальними показникаи та запланованими, що проводиться на фінальному наборі тестів, підібраними певним алгоритмом.

Методи тестуванні web–систем:

— Record & Play;

— Functional Decomposition;

— Data–driven;

— Keyword–driven;

— Object–driven;

— Model–based;

— Capture & Playback [13].

Тестування інтерфейсу базується на таких етапах:

— аналізування поставлених вимог до запланованого інтерфейсу;

— тестування розробленого ресурсу;

— збір результатів тестування усіх елементів інтерфейсу;

— встановлення повноти виконання поставлених завдань.

Тестування на комфортабельність використання розробленого ресурсу виконується за допомогою одного з найбільш розповсюджених інтерактивних сервісів з користувачем (“usability”). Даний метод перевірки направлений на проходження таких етапів:

— вискорівневи дослідження інтерфейсу;

— оцінка прототипу та поставлених вимог;

— валідація;

— періодичне порівняльне тестування розробки на етапах його розробки.

Тестування безпеки інформаційного ресурсу засновано на безпеці серверу та можливості його динамічного збереження даних. Під тестування попадає увесь сервер, проходить повну діагностику системи на наявність помилок чи перебоїв у роботі.

Тестування навантаження являє собою штучно створену одночасну роботу сотень чи тисяч користувачів одночасно, дане тестування придназначене для виявлення можливості розробленого програмного засобу до тійкої роботи при великих об’ємах опрацювання інформації. Такого роду тести дають можливість спільну комунікаційну роботу всього комплексу ― а саме веб–серверу, апаратної частини серверу, програмного ядра.

Уся множина тестування навантаження складється з таких систем:

— продуктивність (performance testing);

— стресове тестування (stress testing);

— об'ємне тестування (volume testing);

— стабільності або надійності (stability / reliability testing);

— моделювання транзакцій (transaction simulation, TS);

— “аналіз даних на стороні клієнта” (client capture, CC);

— “аналіз мережного трафіка” (network sniffing, NS).

3.3 Тестування підсистеми пошуку авто

В даній підсистемі реалізований широкий вибір різноманітних варіантів автомобілів, які запропоновані для вибору потенційним клієнтом. Процес пошуку може відбуватись вручну, а саме переглядом представлених авто зі списку та переходом до їх детального технічного опису, або ж автоматично, за допомогою введення ключових символів ідентифікаторів. До кожного з представлених користувачеві автомобілів у списку буде прикріплене його фото, пошкодження, детально описані всі його технічні характеристики.

Для приклад було взято такі варіанти пошукового запиту:

— введення в строку пошуку цифрового ідентифікатора (рисунок 3.1);

— введення в строку пошуку буквеного ідентифікатора (рисунок 3.2, стор. 58);

— введення в строку пошуку поєднаного цифро–буквеного ідентифікатора (рисунок 3.3, стор. 58);

— введення в строку пошуку поєднаного буквено–цифрового ідентифікатора (рисунок 3.4, стор. 59).

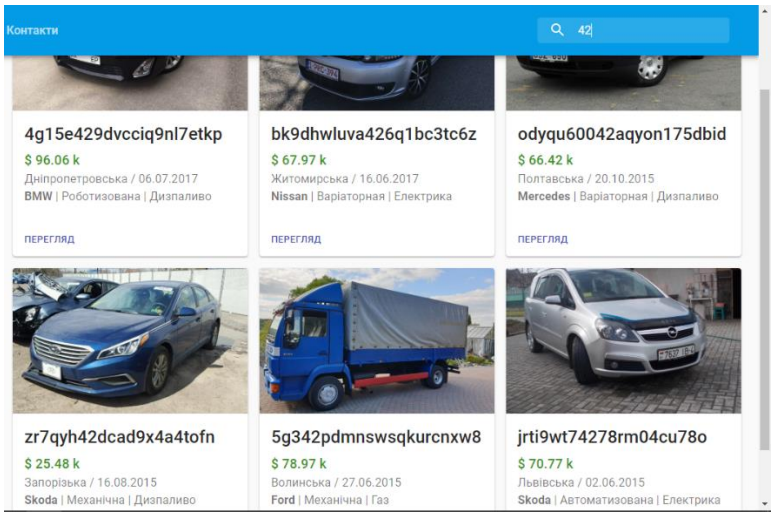

Рисунок 3.1 — Тестування при введенні цифрового ідентифікатора

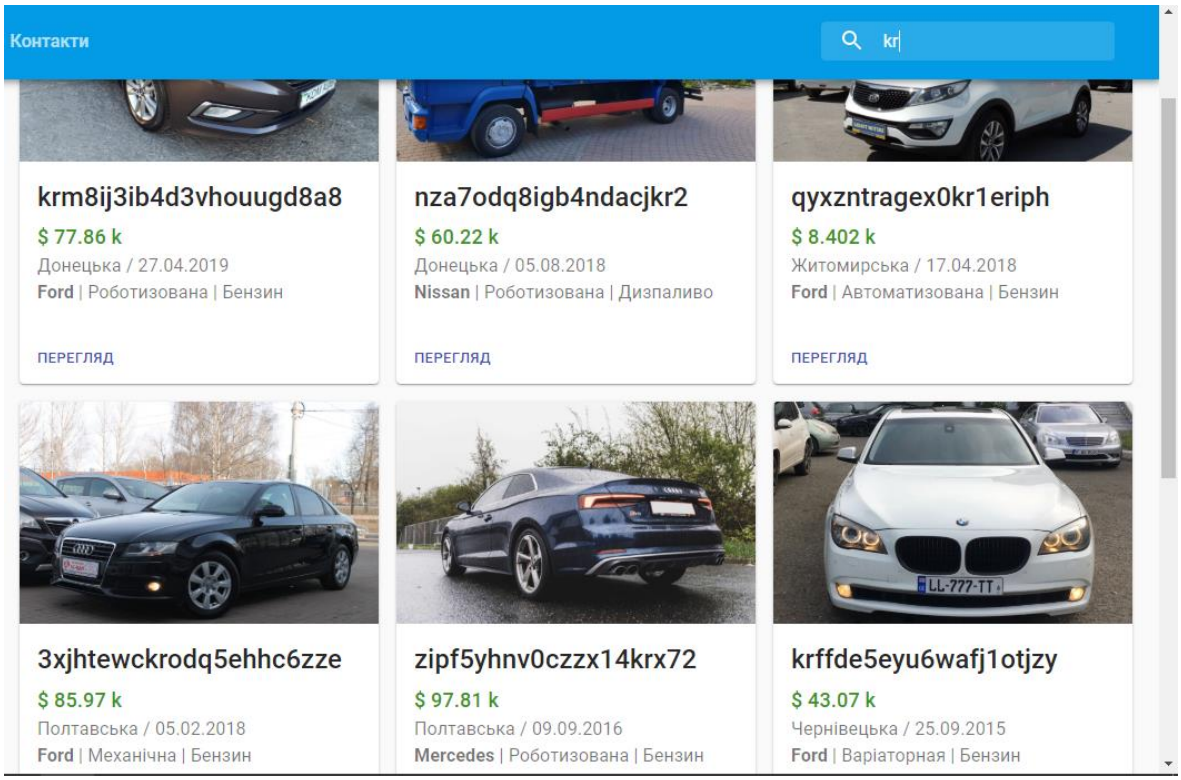

Рисунок 3.2 — Тестування при введенні буквеного ідентифікатора

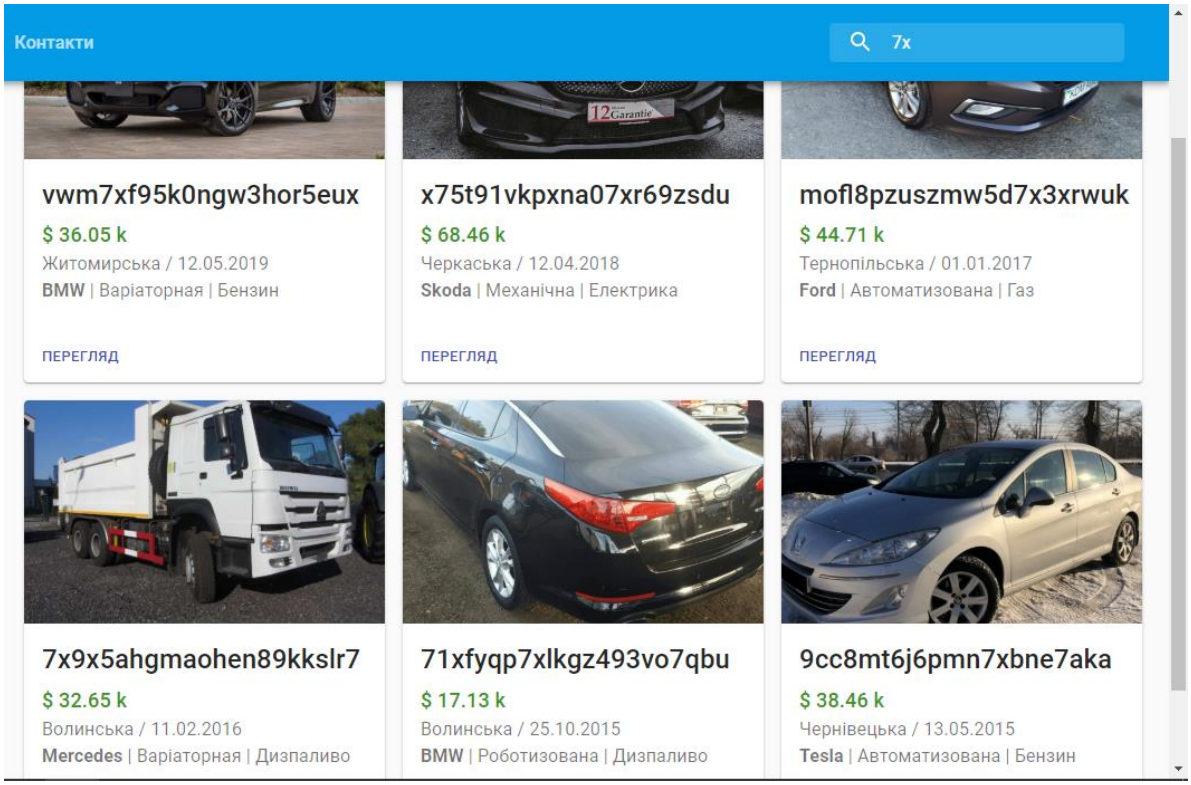

Рисунок 3.3 — Тестування при введенні цифро–буквеного ідентифікатора

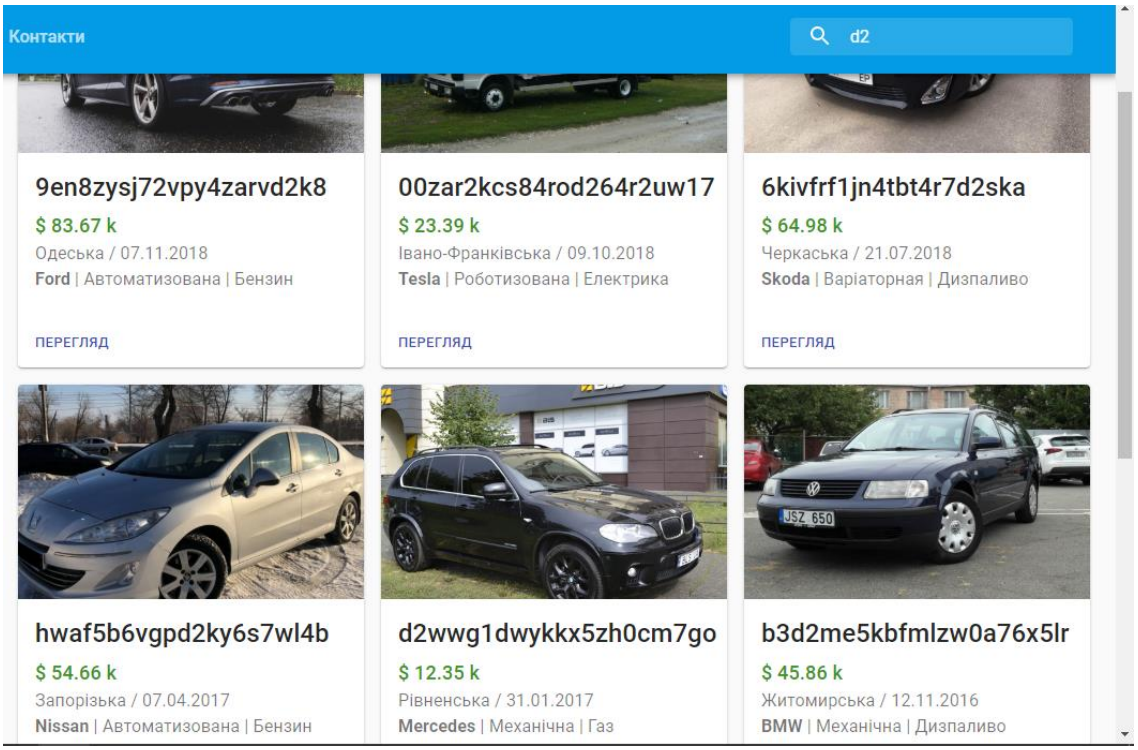

Рисунок 3.4 — Тестування при введенны буквено–цифрового ідентифікатора

3.4 Тестування підсистеми підбору авто

Підсистема підбору авто створена для продвинутого користувача, який розуміється на конструкції та технічних характеристиках автомобілів, та взмозі сам задати усі необідні параметри, що його цікавлять, та отримує на виході декілька варіантів автомобілів, які підходять під задані умови. Процес задання параметрів відбувається вручну, користувач обирає один із можливих варіантів або ж вводить відповідні дані в строки. Список технічних характеристик включає в себе такі пункти: тип кузову, тип двигуна, марка, модель, рік виробництва транспортного засобу (ТЗ), тип палива, тип коробки переключення передач (КПП), тип приводу, витрати палива на 100 кілометрів ходу, об’єм двигуна та його потужність, фактичний пробіг на момент придбання та орієнтована вартість даного авто.

Для приклад було взято такі варіанти задання параметрів підбору :

— сортування автомобілів за датою (рисунок 3.5);

— сортування автомобілів за вартістю (рисунок 3.6);

— підбір авто по областям України (рисунок 3.7);

— підбір авто за їх маркою (рисунок 3.8);

— підбір авто за типом кузова (рисунок 3.9);

— підбір авто в діапазонах об’єму двигуна (рисунок 3.10);

— підбір авто за коробкою передач (рисунок 3.11);

— підбір авто за типом палива (рисунок 3.12);

— підбір авто при заданні двох параметрів одночасно (рисунок 3.13);

— підбір авто при заданні трьох параметрів одночасно (рисунок 3.14);

— підбір авто при заданні чотирьох параметрів одночасно (рисунок 3.15);

— підбір авто при заданні п’яти параметрів одночасно (рисунок 3.16);

— підбір авто при заданні шести параметрів одночасно (рисунок 3.17);

— підбір авто при заданні семи параметрів одночасно (рисунок 3.18).

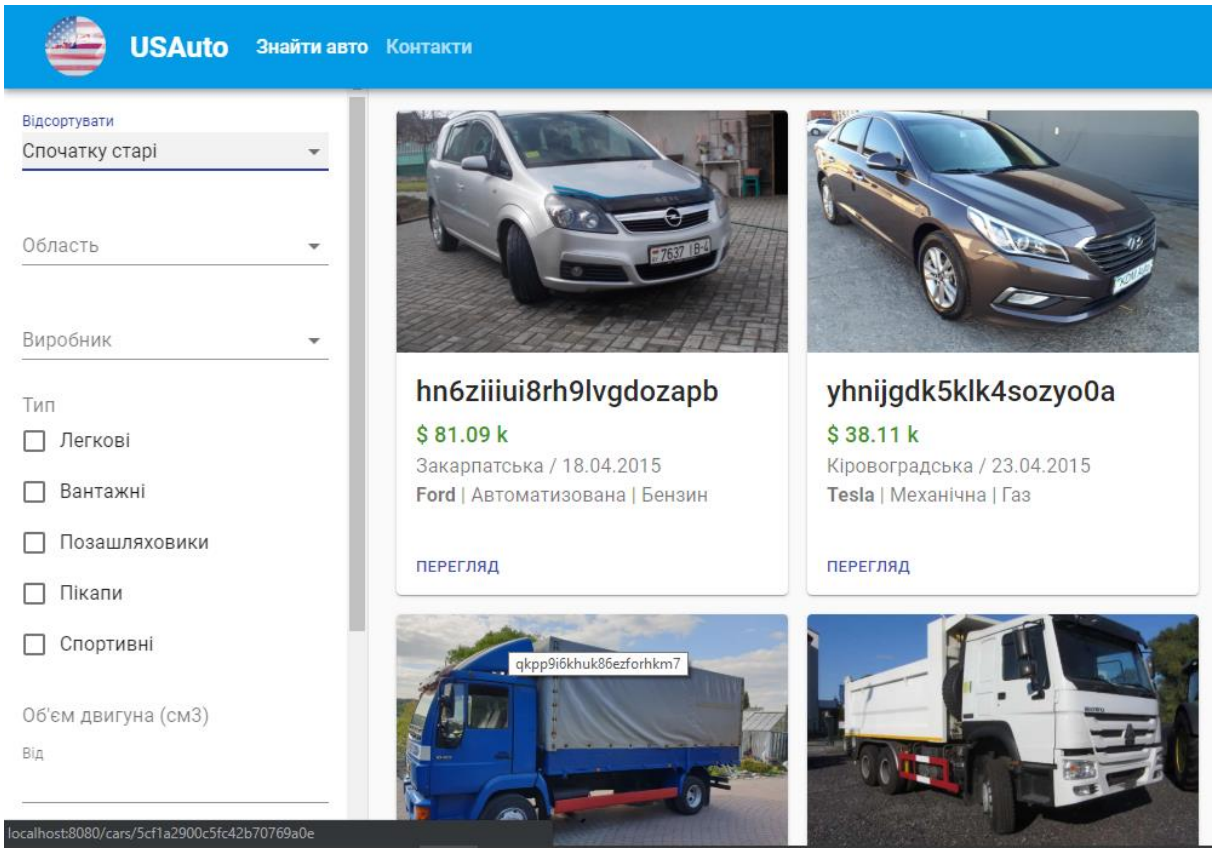

Рисунок 3.5 — Тестування сортування авто за датою

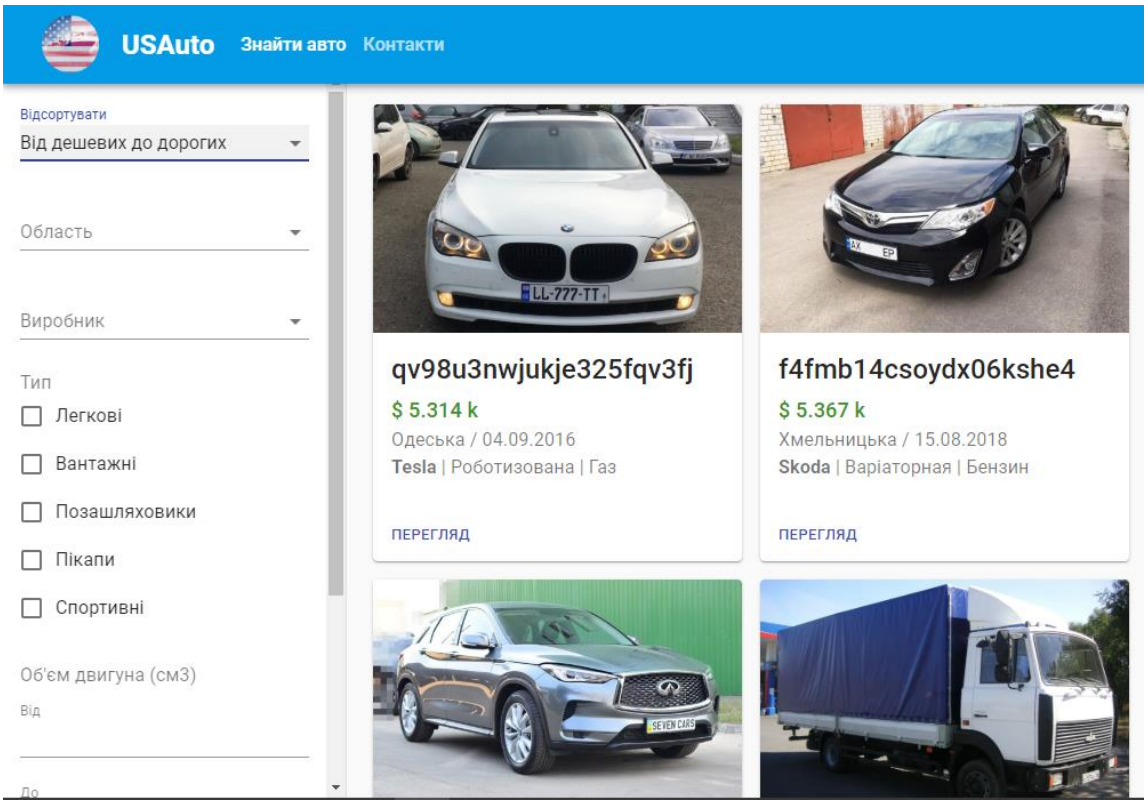

Рисунок 3.5 — Тестування сортування авто за вартістю

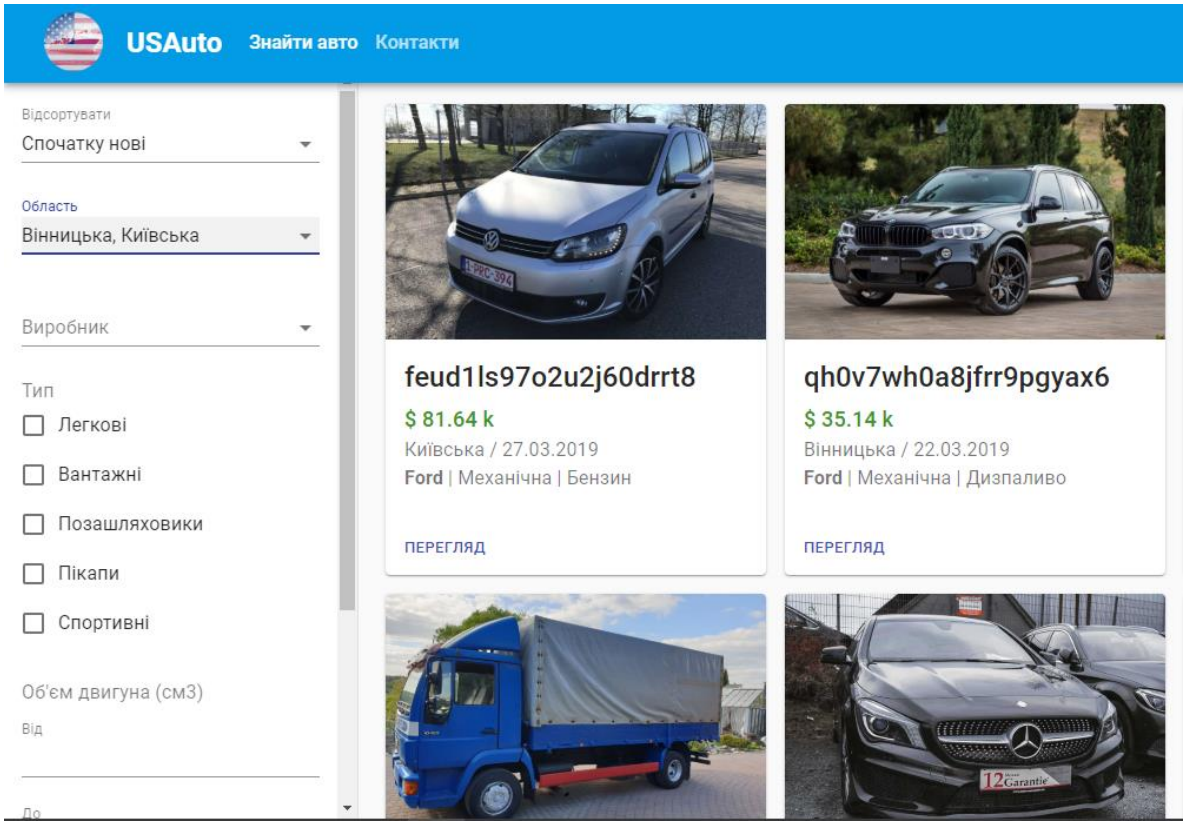

Рисунок 3.7 — Тестування підбору авто по областям України

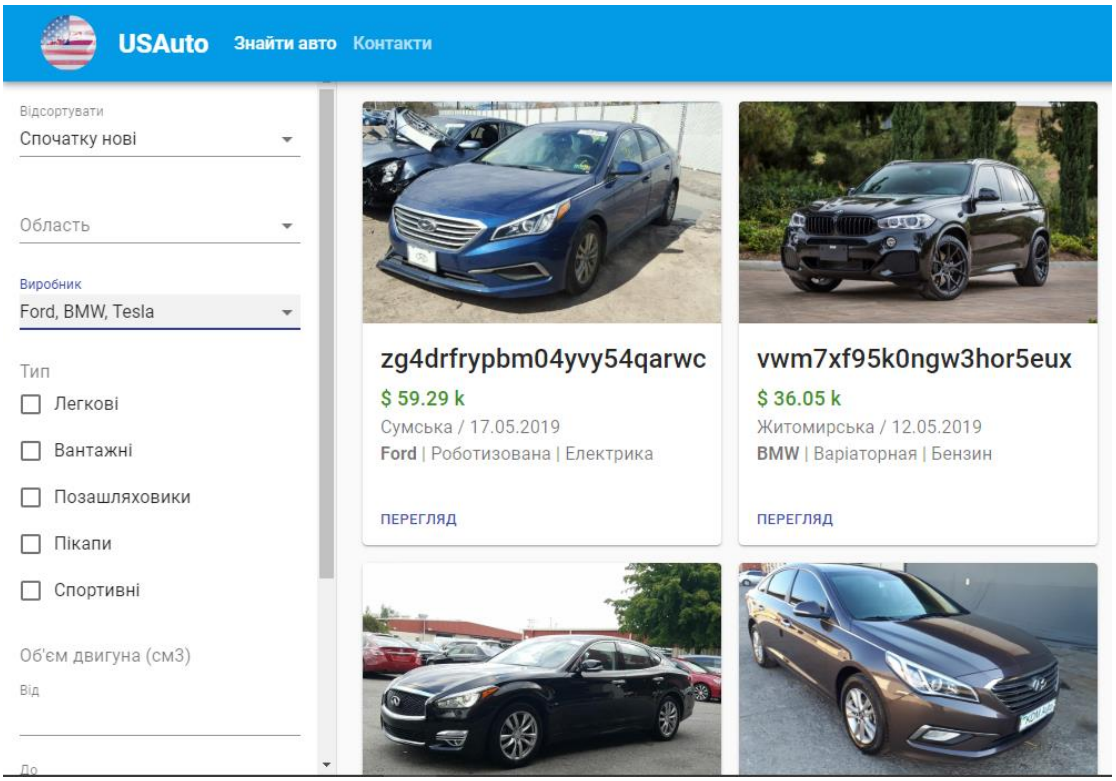

Рисунок 3.8 — Тестування підбору авто за маркою

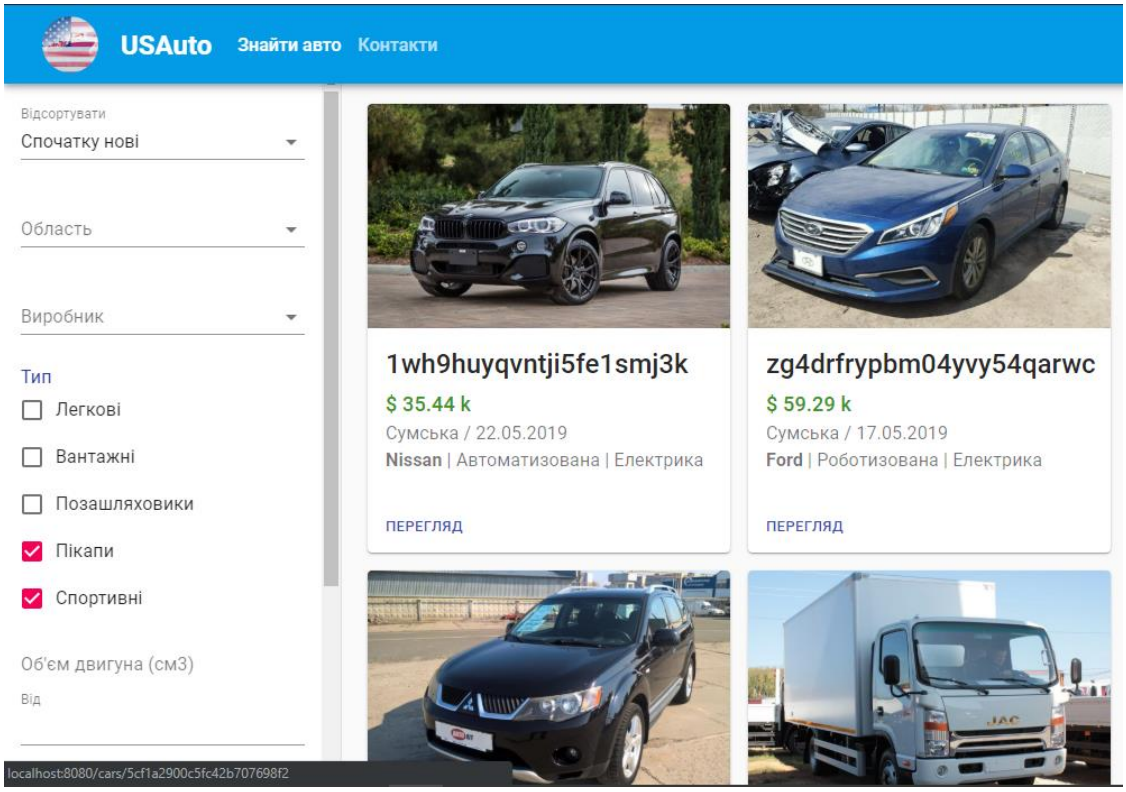

Рисунок 3.9 — Тестування підбору авто за типом кузова

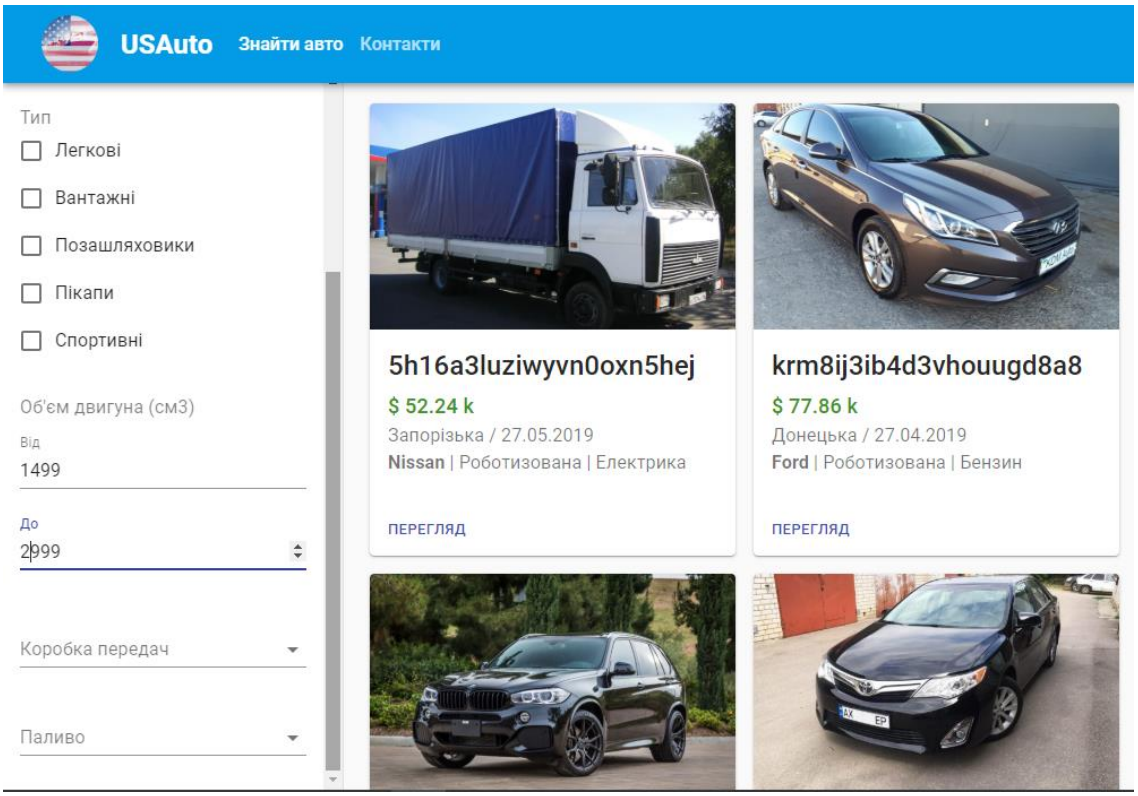

Рисунок 3.10 — Тестування підбору авто в діапазонах об'єму двигуна

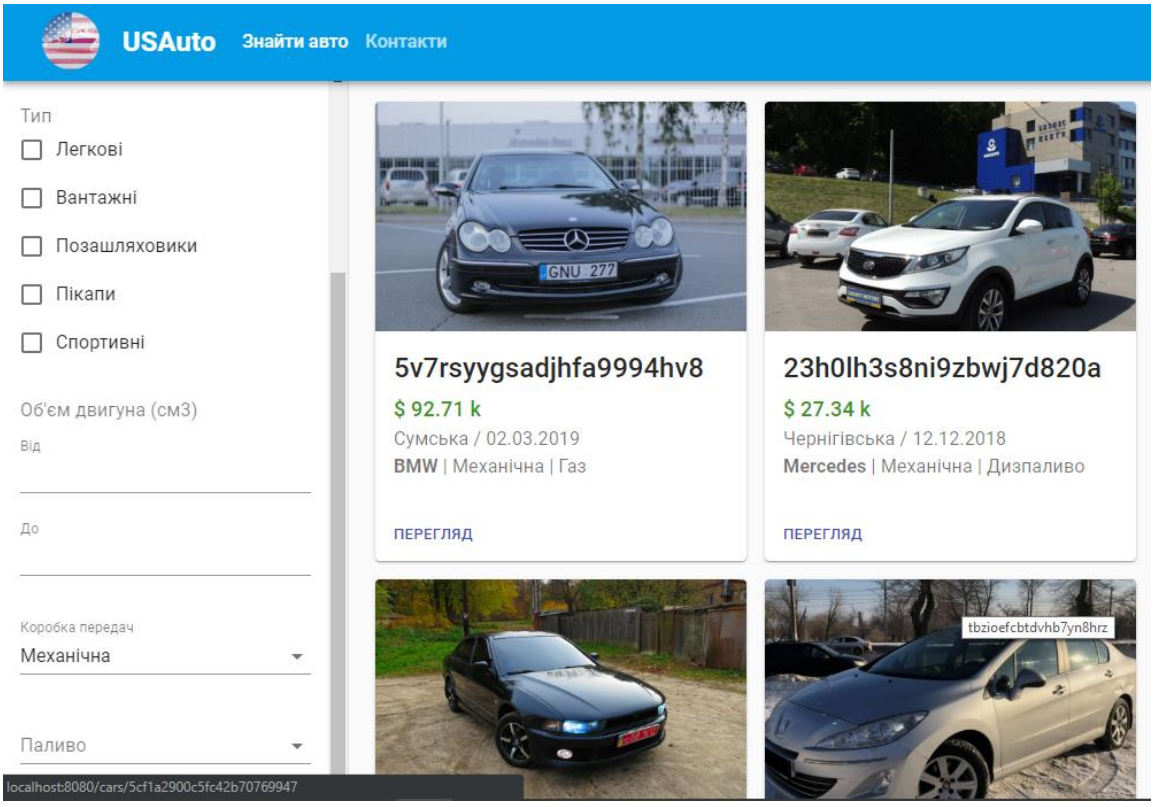

Рисунок 3.11 — Тестування підбору авто за типом КПП

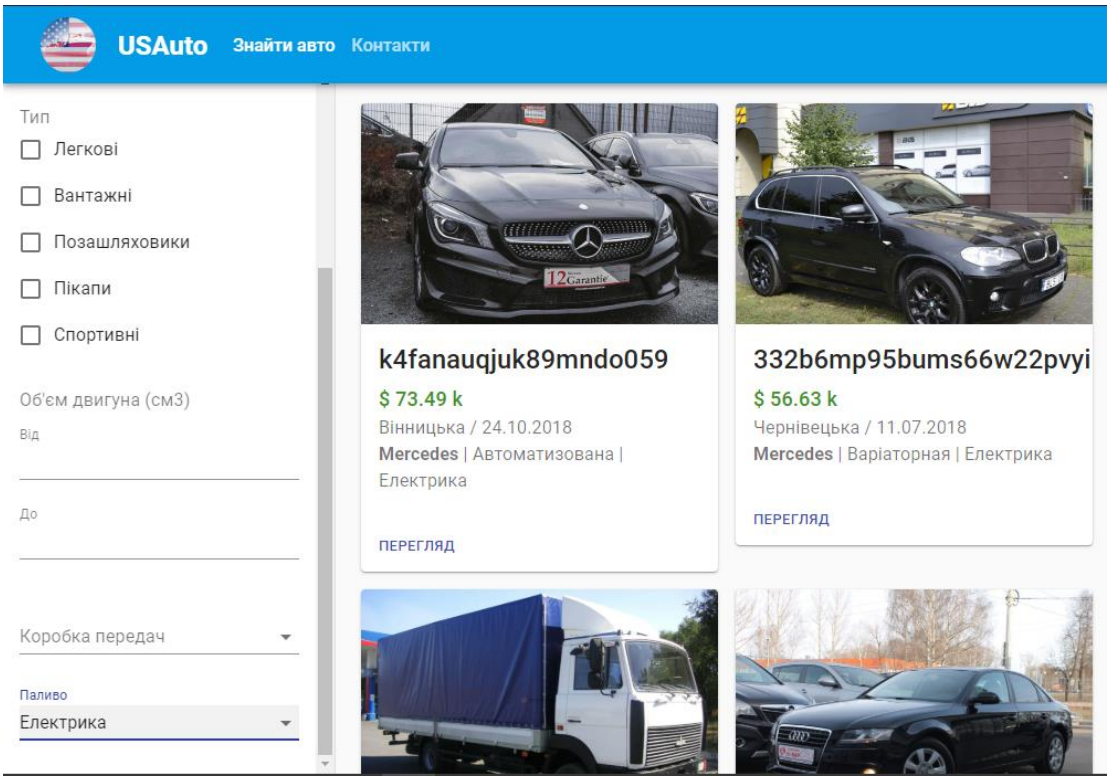

Рисунок 3.12 — Тестування підбору авто за типом палива

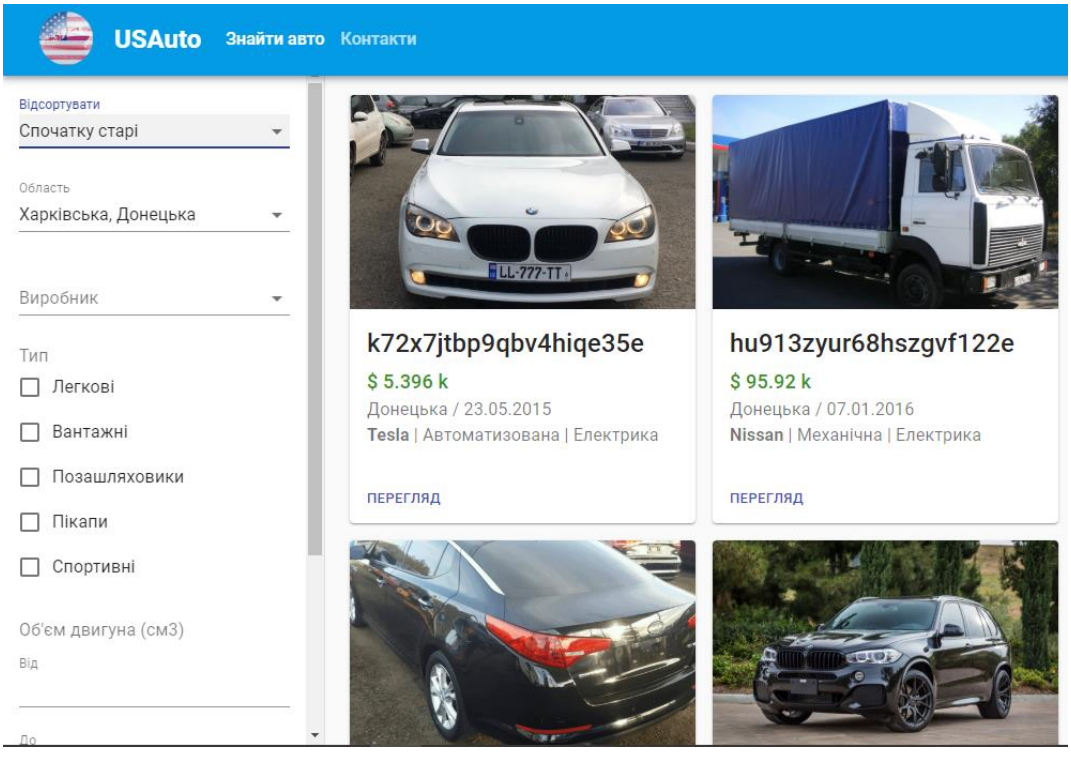

Рисунок 3.13 — Тестування підбору авто при заданні двох параметрів одночасно

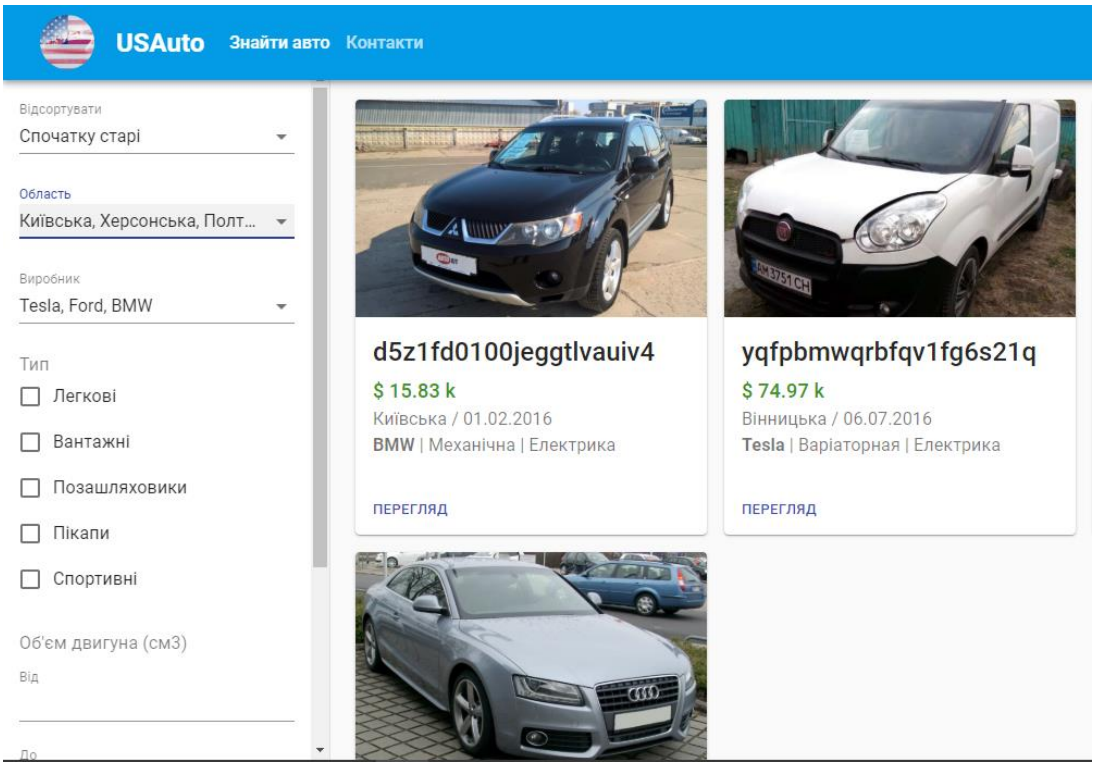

Рисунок 3.14 — Тестування підбору авто при заданні трьох параметрів одночасно

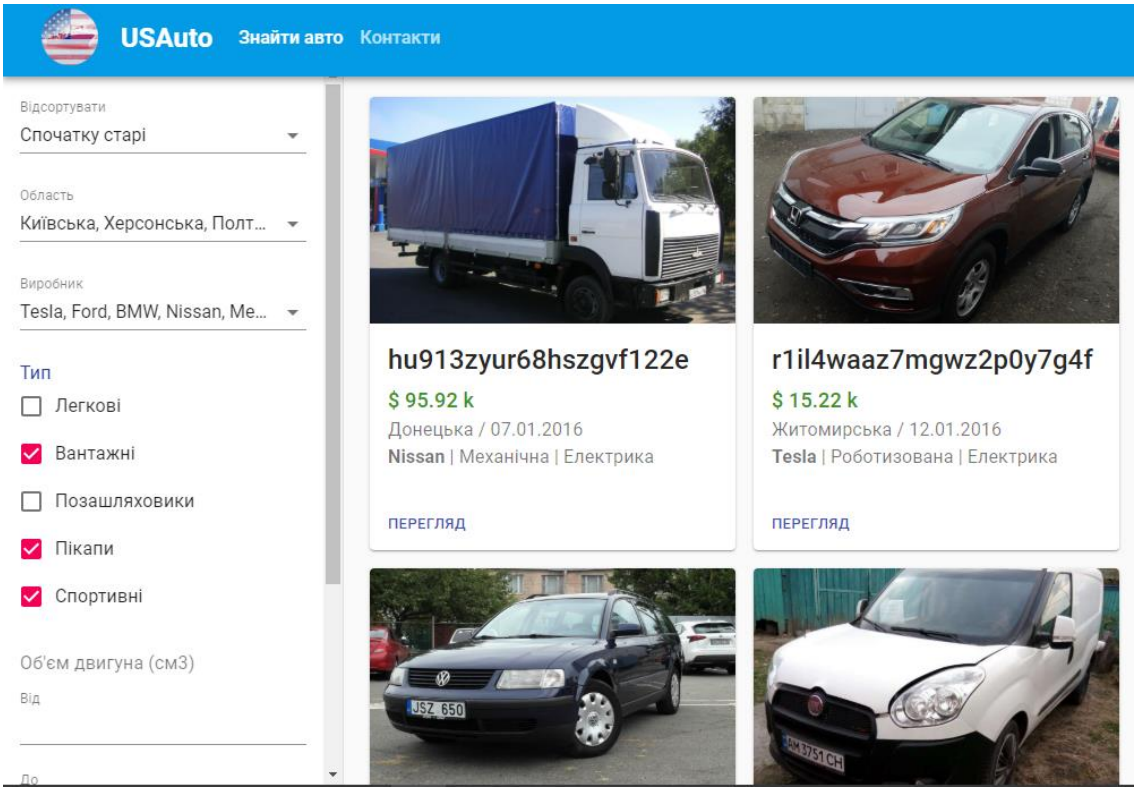

Рисунок 3.15 — Тестування підбору авто при заданні чотирьох параметрів одночасно

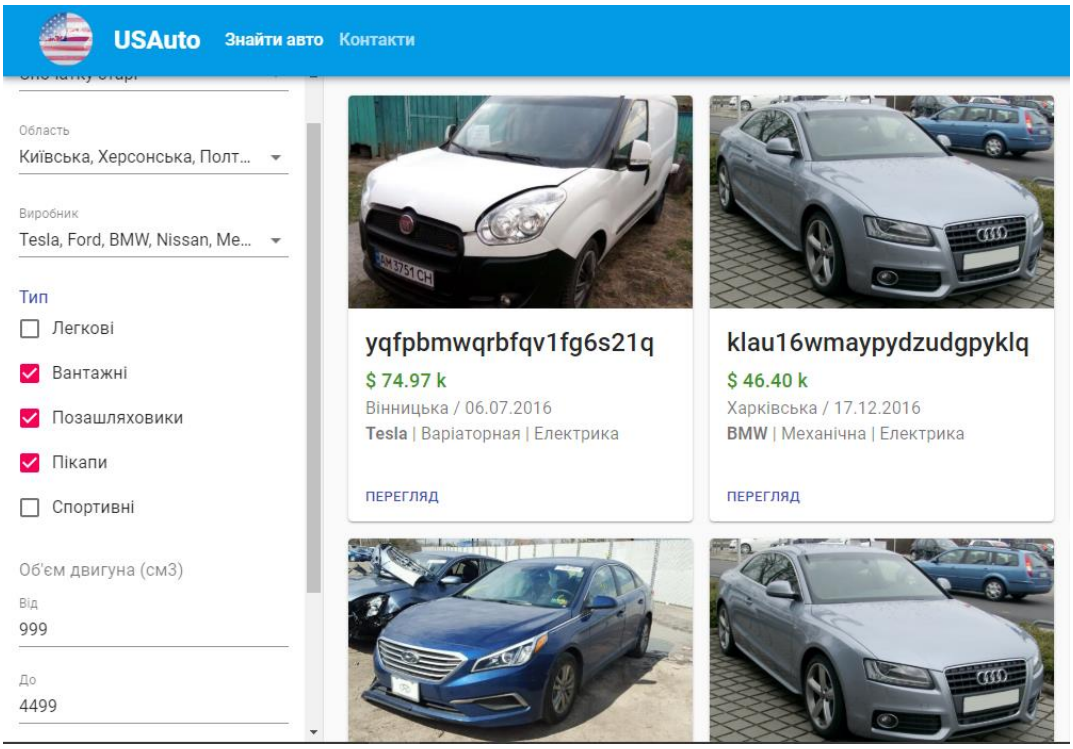

Рисунок 3.16 — Тестування підбору авто при заданні п’яти параметрів одночасно

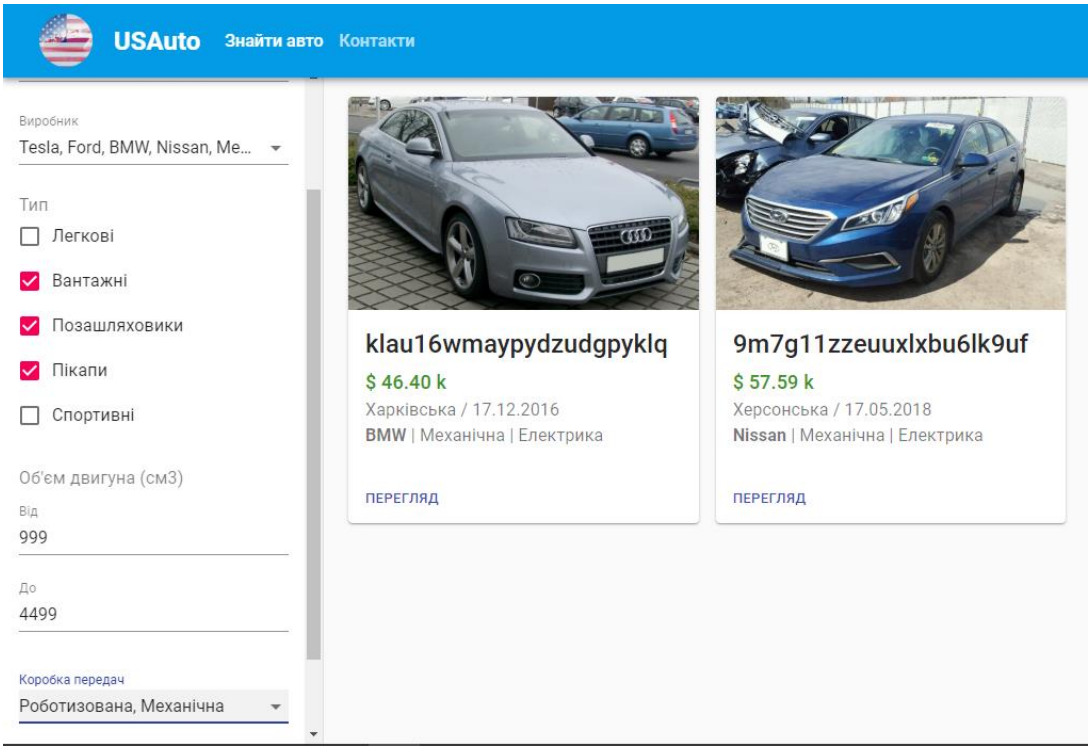

Рисунок 3.17 — Тестування підбору авто при заданні шести параметрів одночасно

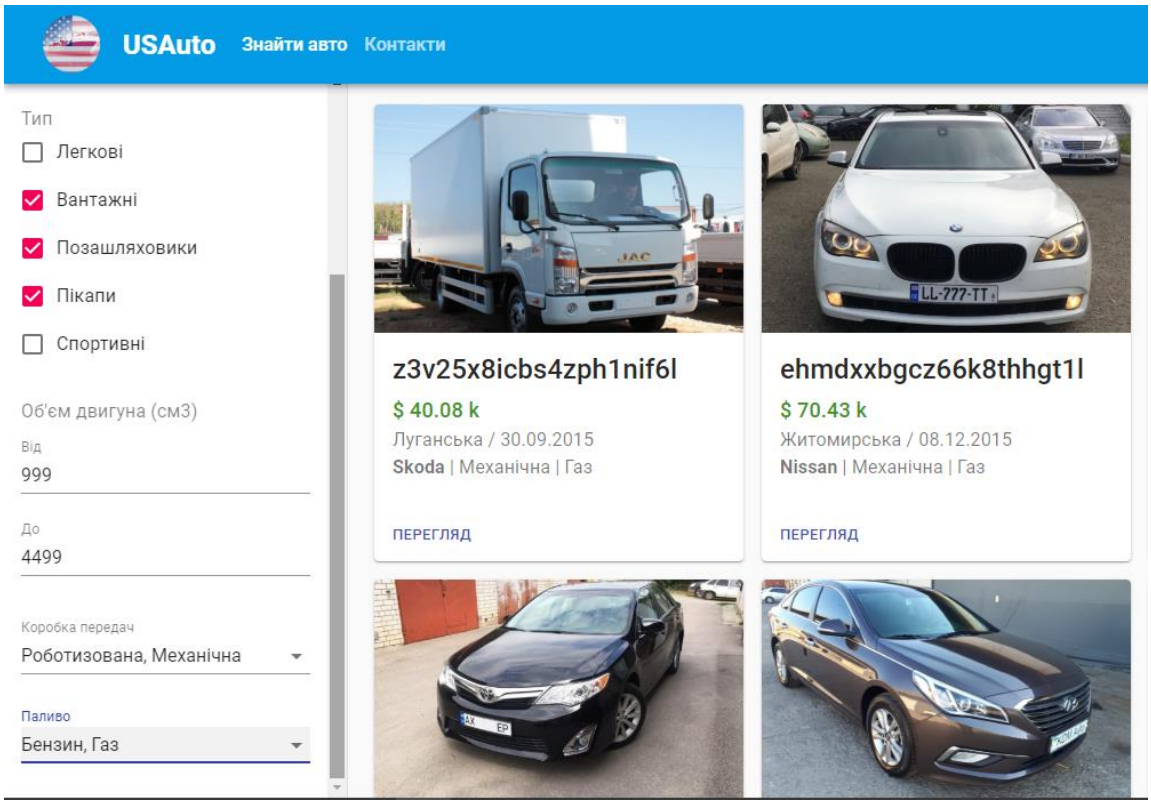

Рисунок 3.18 — Тестування підбору авто при заданні семи параметрів одночасно

3.5 Тестування підсистеми виведення даних

Підсистема виведення даних створена для аналізу обраних з запропонованого списку або ж підібраних вручну за технічними характеристиками автомобілів, та створення форми виведення їх користувачеві. Залежно від заданих параметрів, на виході буде отримано одне або ж декілька авто, що задовольняють умови пошуку. Після цього наступає фінальний етап – перехід до замовлення обраного автомобіля шляхом введення особистих даних, які будуть збережені на сервер. Для цього користувач залишає свої особисті дані, а саме: ПІБ, контактний номер телефону, електронну пошту, серію та номер паспорту, ідентифікаційний код платника податків.

Для прикладу було взято введення та збереження даних:

— введення особистих даних до інформаційної системи (рисунок 3.19);

— відображення введених даних у базі MongoDB (рисунок 3.20).

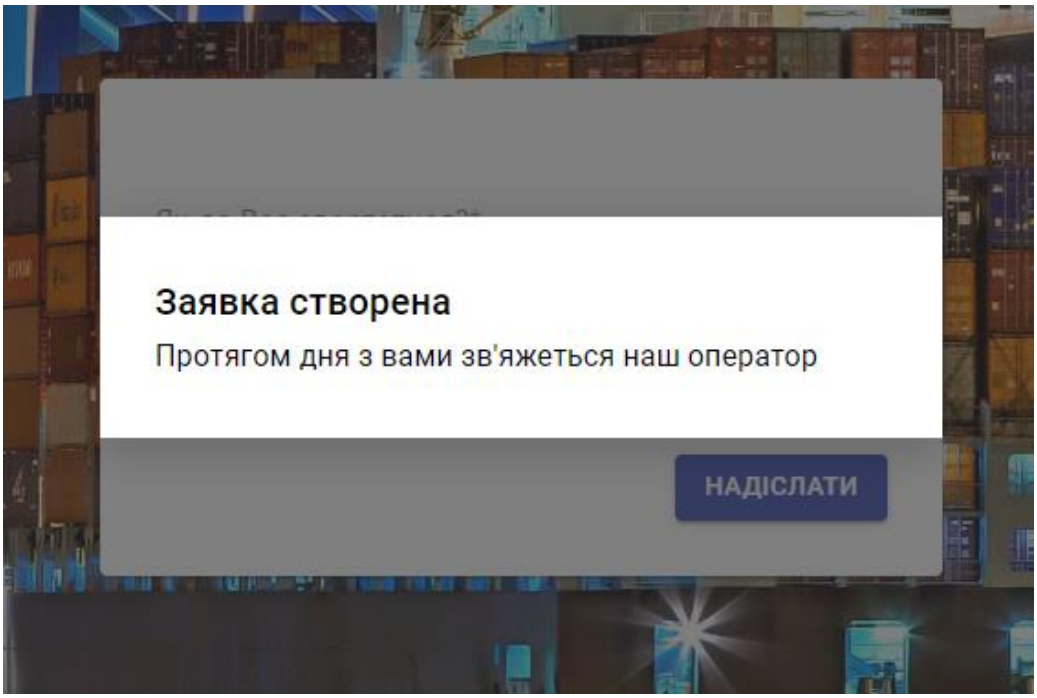

Рисунок 3.19 — Тестування результату введення особистих даних до інформаційної системи

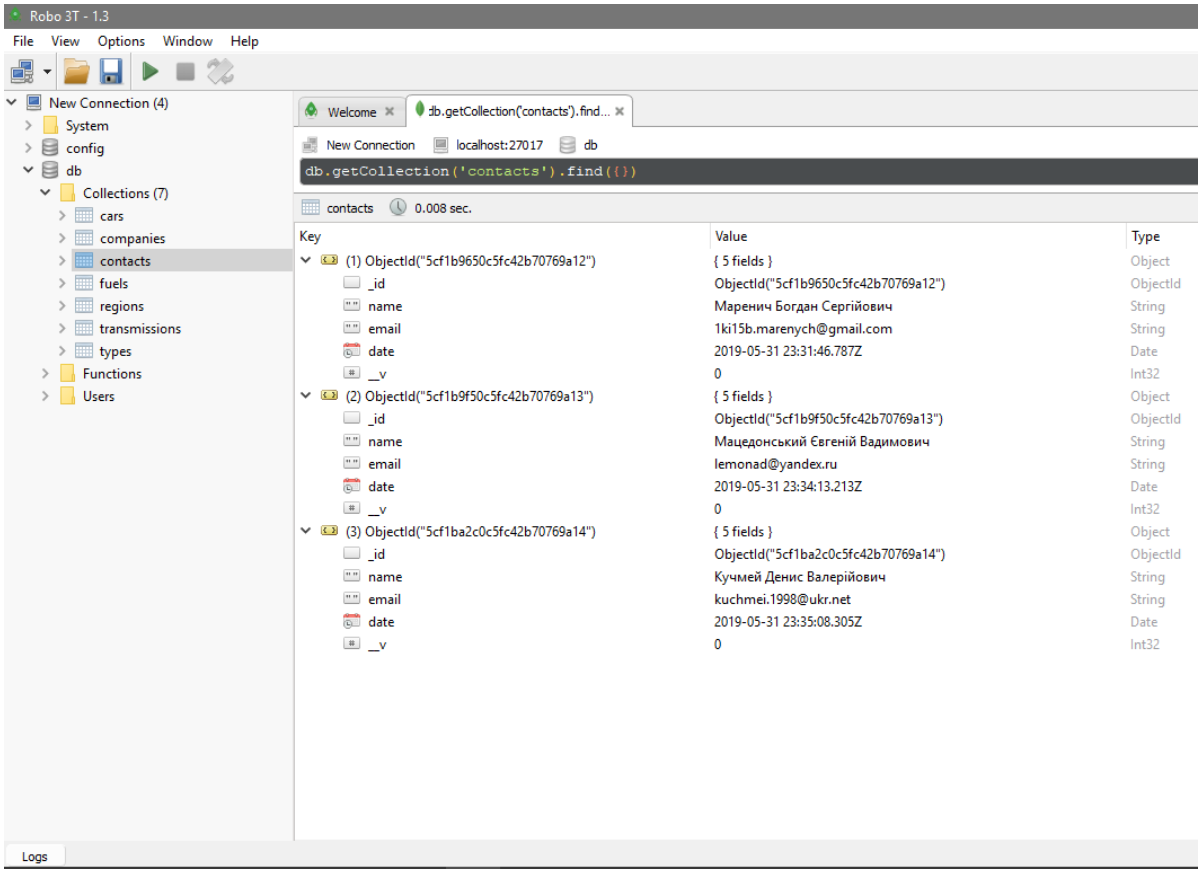

Рисунок 3.20 — Тестування відображення введених даних у базі MongoDB

3.6 Тестування адаптивності інформаційної системи

Інформаційна система підбору та замовлення авто протестована на різних екранах з різною роздільною здатністю. Ці тести покажуть наскільки досконало пророблений адаптивний дизайн даного веб–ресурсу:

— головна сторінка на екрані з роздільною здатністю 1288x558 пікселів (рисунок 3.21);

— головна сторінка на екрані з роздільною здатністю 903x820 пікселів (рисунок 3.22);

— головна сторінка на екрані з роздільною здатністю 500x820 пікселів (рисунок 3.23);

— головна сторінка на екрані з роздільною здатністю 318x656 пікселів (рисунок 3.24);

— сторінка підбору авто на екрані з роздільною здатністю 1326x558 пікселів (рисунок 3.25);

— сторінка підбору авто на екрані з роздільною здатністю 1020x744 пікселів (рисунок 3.26);

— сторінка підбору авто на екрані з роздільною здатністю 956x1116 пікселів (рисунок 3.27);

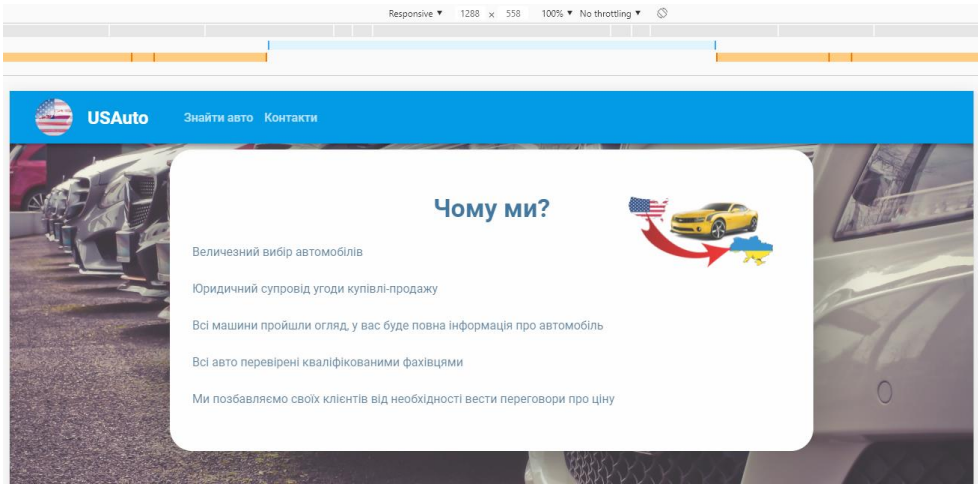

Рисунок 3. 21 — Головна сторінка на екрані з роздільною здатністю 1288x558 пікселів

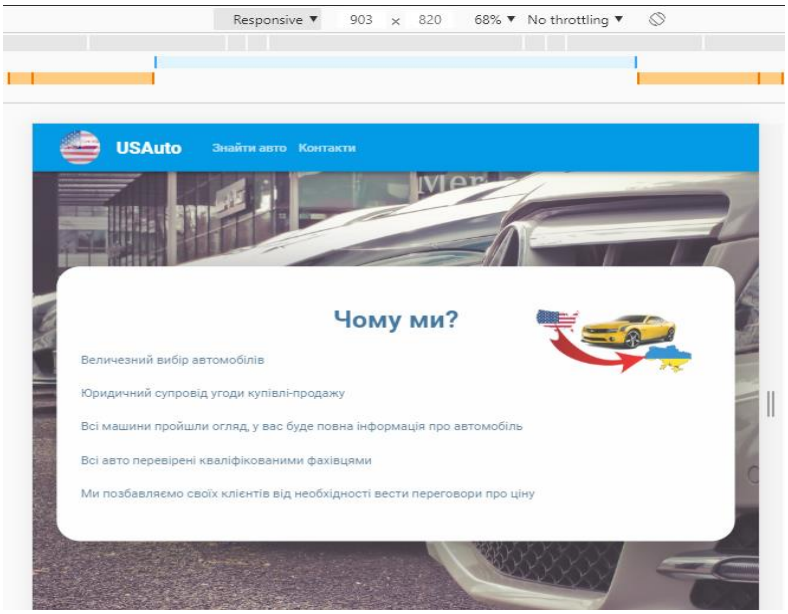

Рисунок 3.22 — Головна сторінка на екрані з роздільною здатністю 903x820 пікселів

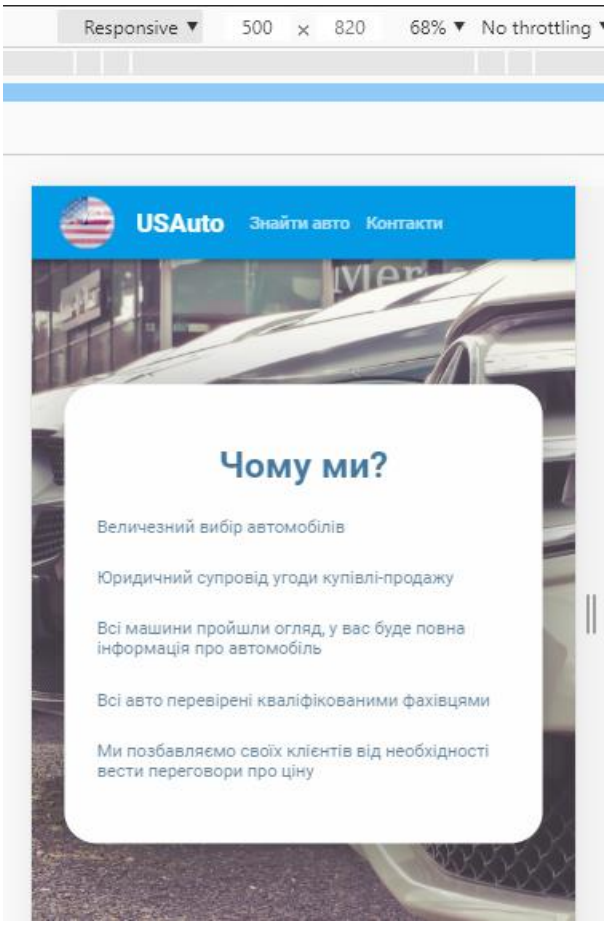

Рисунок 3.23 — Головна сторінка на екрані з роздільною здатністю 500x820 пікселів

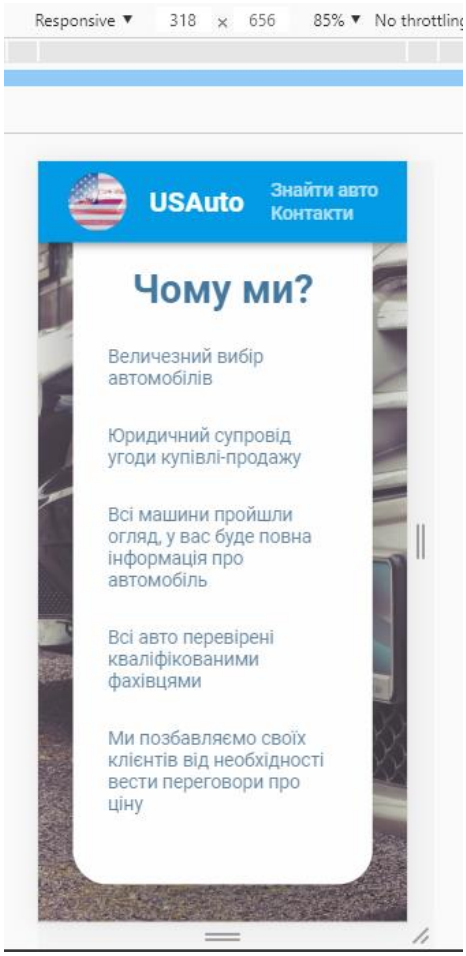

Рисунок 3.24 — Головна сторінка на екрані з роздільною здатністю 318x656 пікселів

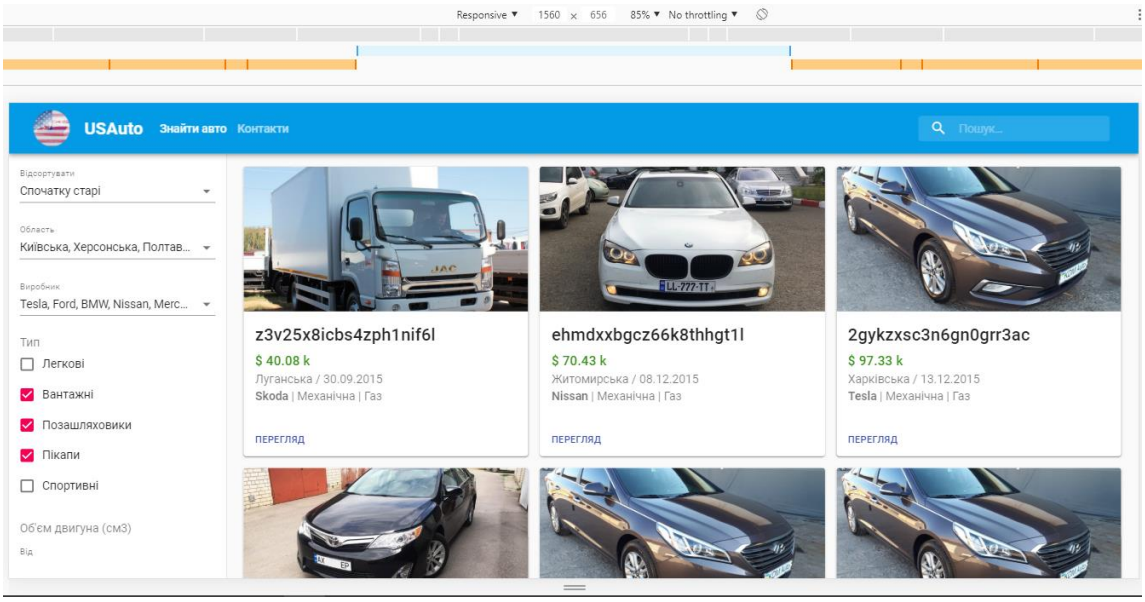

Рисунок 3.25 — Сторінка підбору авто на екрані з роздільною здатністю 1326x558 пікселів

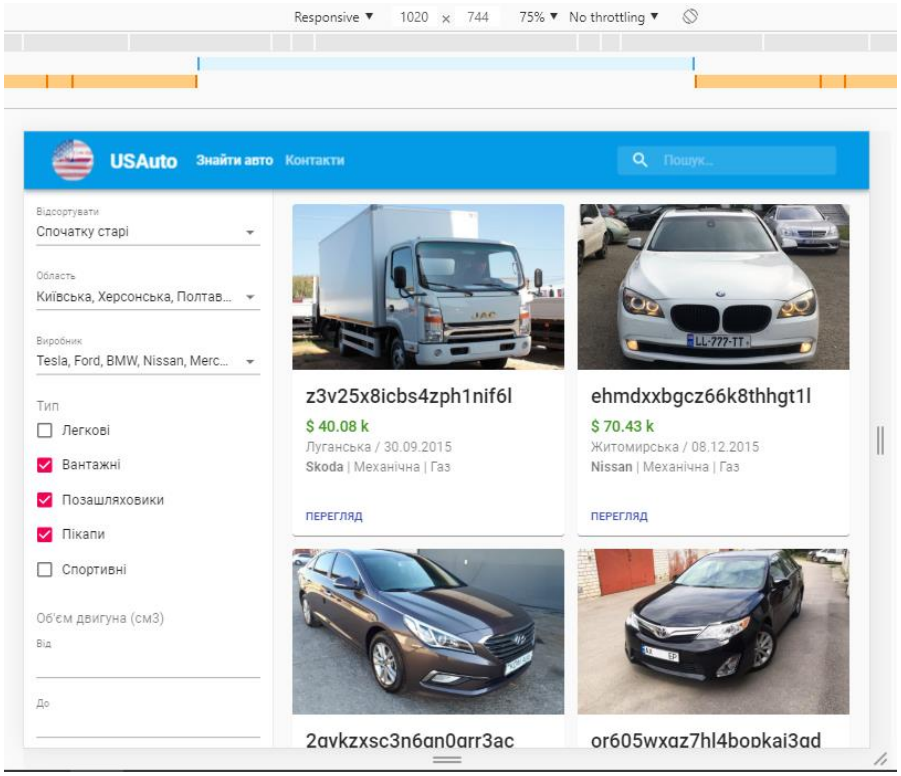

Рисунок 3.26 — Сторінка підбору авто на екрані з роздільною здатністю 1020x744 пікселів

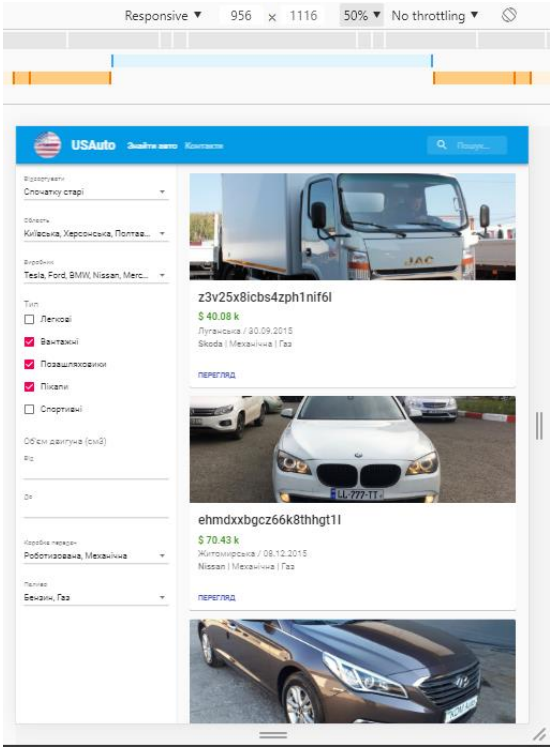

Рисунок 3.27 — Сторінка підбору авто на екрані з роздільною здатністю 956x1116 пікселів

3.7 Інтегральне тестування розробленого програмного засобу

Тестування інтеграції зберігає цілісність системи. Основою інтегрального тестування є: скомпонувати усі модулі та протестовані елементи, та пожднати у одну цілісну програмну структуру згідно з поставленими вимогами до інормаційного ресурсу.

Як і дл тестування програмного засобу, так і для його відлагодження існує два способи, які опрацьовують проце проходження інтеграції: визхідне та низхідне тестування.

Верстку сторінок даної інформаційної системи протестовано за допомогою валідатора — validator.w3.org. Валідатор показав, що ні одна із сторінок не має помилок (рисунок 3.28).

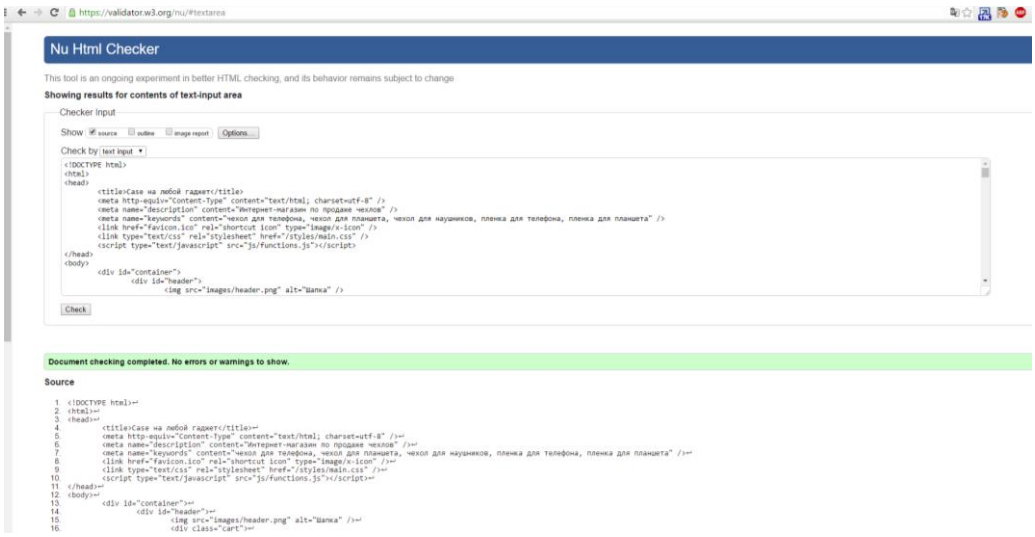

Рисунок 3.28 — Результати тесту валідатора

3.8 Висновки за розділом

В даному розділі кваліфікаційної роботи проведено детальне тестування основних підсистем інформаційної системи та її адаптивності :

— тестування підсистеми пошуку авто;

— тестування підсистеми підбору авто;

— тестування підсистеми виведення даних;

— тестування адаптивності інформаційної системи;

— інтегральне тестування розробленого програмного засобу.

Було протестовано всі можливі варіанти пошуку автомобілів за їхніми ідентифікаторами, проаналізовано та скореговано всі можливі помилки, які можуть допустити користувачі при використанні даної інформаційної системи.

Підсистема підбору авто була детально протестована шляхом задання одиничних параметрів для автомобіля та перевірки коректності виведених даних інформаційною системою. Було проведено перевірку на відповідність виведених результатів підбору при використанні відразу двох та більше технічних характеристик автомобіля.

Значну увагу було приділено тестуванню підсистеми збереження даних до бази даних, перевірено її роботу, усунено помилки при збереженні й подальшому відображенні їх у базі MongoDB.

Для коректного відображення даних інформаційної системи було пройдено ряд перевірок на різних екранах з різною роздільною здатністю, що чудово демонструє адаптивний дизайн даного ресурсу.

Загальну перевірку усієї інформаційної системи протестовано за допомогою валідатора ─ validator.w3.org. Валідатор показав, що ні одна із сторінок не має помилок та все працює коректно.

4 ЕКОНОМІЧНА ЧАСТИНА

4.1. Оцінювання комерційного потенціалу розробки “Web–орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля”

Метою економічної частини магістерської кваліфікаційної роботи є довести економічну доцільність та ефективність впровадження наукової розробки, для цього необхідно виконати такі етапи:

—оцінити комерційний потенціал розробки;

—спрогнозувати витрати на виконання наукової роботи та впровадження її результатів;

—спрогнозувати комерційній ефект від реалізації розробки;

—розрахувати ефективність вкладених інвестицій та їх окупність.

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки, створеної в результаті науково–технічної діяльності.

Для проведення технологічного аудиту залучено трьох незалежних експертів. Кожен з експертів повинен ознайомитися з запропонованою розробкою, та заповнити таблицю, яка визначає рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можливу оцінку в балах. Після виконання цього, підраховується середньоарифметична сума балів та визначається який рівень комерційного потенціалу має нова розробка.

Для визначення комерційного потенціалу дослідження необхідно залучити 2–х або 3–х незалежних експертів. Залучимо таких експертів:

—Захарченко С.М., доцент кафедри ОТ;

—Колеснік І.С., доцент кафедри ОТ;

—Біліченко Н.О., доцент кафедри ОТ.

Оцінювання комерційного потенціалу дослідження здійснюється за 12–ма критеріями, що наведені в таблиці 4.1.

Таблиця 4.1 — Критерії оцінювання комерційного потенціалу дослідження та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5–ти бальною шкалою)					
Кри–терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає

Продовження таблиці 4.1

Критерії оцінювання та бали (за 5–ти бальною шкалою)					
Кри–терій	0	1	2	3	4
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово–промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10–ти років	Термін реалізації ідеї від 3–х до 5–ти років. Термін окупності інвестицій більше 5–ти років	Термін реалізації ідеї менше 3–х років. Термін окупності інвестицій від 3–х до 5–ти років	Термін реалізації ідеї менше 3–х років. Термін окупності інвестицій менше 3–х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові–домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь–які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу дослідження заносимо в таблицю 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу дослідження

Критерії	Прізвище, ініціали, посада експерта		
	Захарченко В.П., доцент кафедри ОТ;	Колеснік І.С., доцент кафедри ОТ;	Біліченко Н.О., доцент кафедри ОТ;
	Бали, виставлені експертами:		
1	4	3	3
2	3	3	3
3	2	2	2
4	3	3	3
5	2	2	1
6	4	4	4
7	3	3	3
8	4	4	4
9	2	2	2
10	3	2	3
11	4	4	4
12	4	4	4
Сума балів	$СБ_1 = 38$	$СБ_2 = 36$	$СБ_3 = 36$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = 36{,}67 \approx 37$		

За результатом оцінювання комерційного потенціалу дослідження експертами вийшло, що рівень комерційного потенціалу дослідження вище середнього.

4.2 Оцінювання комерційного потенціалу

Кошторис витрат на дослідження може складатися з таких етапів:

1–й етап*:* кошторис витрат на дослідження передбачає розрахунок таких основних витрат:

Основна заробітна плата розробників розраховується згідно за формулою [4.1]:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \quad (4.1)$$

де, М — місячний посадовий оклад конкретного розробника [грн];

$Т_р$ — число робочих днів в місяці, $Т_р = 20$;

t — число робочих днів роботи розробника.

Підставимо дані до формули 5.1 та отримаємо:

$$З_о = \frac{15500}{20} \cdot 10 = 7750 \text{ (грн)},$$

$$З_о = \frac{10000}{20} \cdot 35 = 17500 \text{ (грн)},$$

$$\sum З_о = 7750 + 17500 = 25250 \text{(грн)}.$$

Зроблені розрахунки заносимо до таблиці 4.3.

Таблиця 4.3 — Витрати на заробітну плату розробників

Найменування посади виконавця	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на оплату праці, грн.
Науковий керівник	15500	775	10	7750
Інженер–програміст	10000	500	35	17500
Всього				$\sum З_о = 25250$

Додаткова заробітна плата ($З_д$) всіх розробників та робітників, які приймали участь в дослідженні [4.2].

Додаткову заробітну плату розраховуємо як 10...12% від суми основної заробітної плати всіх розробників та робітників. Приймаємо додаткову заробітну плату 10% від основної [4.2]:

$$З_д = 0{,}10 \cdot (З_о + З_р). \quad (4.2)$$

$$З_д = \frac{(25250) \cdot 10}{100\%} = 2525{,}0 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{зп}$ розробників та робітників, які брали участь у виконанні даної НДДКР становить 22 % від суми основної та додаткової заробітної плати [4.2]:

$$Н_{зп} = (З_о + З_р + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (4.3)$$

де, $З_о$ — основна заробітна плата розробників, грн.;

$З_р$ — основна заробітна плата робітників, грн.;

$З_д$ — додаткова заробітна плата всіх розробників і робітників, грн.;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %. $\beta = 22\%$.

$$Н_{зп} = (25250 + 2525) \cdot \frac{22}{100} = 6110{,}5 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів та приміщень А, які використовувались для дослідження. У спрощеному вигляді амортизаційні відрахування по кожному виду обладнання та приміщенням можуть бути розраховані за формулою [4.4]:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ [грн]}, \quad (4.4)$$

де, Ц — загальна балансова вартість даного виду обладнання (приміщень), грн.;

$Н_а$ — річна норма амортизаційних відрахувань. $Н_а = (5 \dots 25)\%$. Обираємо $Н_а = 20\%$;

Т — термін використання обладнання (приміщень), місяці.

Всі проведені розрахунки амортизаційних відрахувань заносимо в таблицю 4.4.

Таблиця 4.4 — Розрахунок амортизаційних відрахувань

Найменування комплектуючих	Балансова вартість, грн.	Норма амортизації, %	Термін використання міс.	Величина амортизаційних відрахувань, грн.
1.Комп’ютер	17300	20	2	574
2.Програмне забезпечення MongoDB, Microsoft Visual Studio	6750	20	2	223,29
3.Приміщення лабораторії	110 000	5	3	1371
Всього				$А = 2168{,}29$

Витрати на послуги, що були використані під час виконання наукової розробки, заносимо у таблицю 4.5

Таблиця 4.5 — Розрахунок витрат на послуги

Найменування послуг	Термін використання, місяців	Ціна за місяць, грн	Вартість послуг
Хостинг	6	750	4500
Доменне ім'я	12	35	420
Всього			4920

Витрати на силову електроенергію $В_е$ розраховуємо за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п \text{ [грн]}, \quad (4.5)$$

де, В — вартість 1 кВт–год. електроенергії, в 2020 році складає $В = 3{,}5$ грн./кВт;

П — установлена потужність обладнання, кВт. Споживана електрична потужність комп'ютера та інших пристроїв при проведенні досліджень $П = 0{,}2$ кВт;

Ф — фактична кількість годин роботи обладнання, годин. Фактична кількість годин роботи комп'ютера та інших пристроїв при проведенні досліджень $Ф = 20 \cdot 8 = 160$ годин;

$К_п$ — коефіцієнт використання потужності, $К_п < 1$. Обираємо $К_п = 0{,}9$.

Отже, витрати на силову електроенергію становлять:

$$В_е = 3{,}5 \cdot 0{,}2 \cdot 160 \cdot 0{,}9 = 100{,}8 \text{ (грн)}.$$

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану НДДКР, тобто:

$$В_{ін} = (З_о + З_р) \cdot (100\% \dots 300\%)\ [грн], \quad (4.6)$$

$$В_{ін} = \frac{(25250) \cdot 125\%}{100\%} = 31975\ (грн).$$

Сума всіх попередніх статей витрат дає загальні витрати на виконання даної частини НДДКР — В.

$$В = З_о + З_д + Н_{зп} + А + М + К + В_е + В_{ін}\ [грн], \quad (4.7)$$

$$В = 25250 + 2525 + 6110{,}5 + 2168{,}29 + \\ +4920 + 100{,}8 + 31975 = 73048{,}55\ (грн).$$

2–й етап: розрахунок загальних витрат на дослідження. Загальна вартість всієї НДДКТ $В_{заг}$ визначається за формулою:

$$В_{заг} = \frac{В}{\alpha}\ [грн], \quad (4.8)$$

де, α — частка витрат, які безпосередньо здійснює виконавець даної НДДКР, у відн. одиницях. Приймемо $\alpha = 0{,}7$.

$$В_{заг} = \frac{73048{,}55}{0{,}9} = 81165{,}05\ (грн).$$

3–й етап: прогнозування загальних витрат на виконання та впровадження наукової розробки. Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної НДДКР здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}\ [грн], \quad (4.9)$$

де, β — коефіцієнт, який характеризує етап виконання даної НДДКР. Приймаємо $\beta = 0{,}9$.

$$ЗВ = \frac{81165{,}55}{0{,}9} = 90183{,}38(грн).$$

4.3 Прогнозування комерційних ефектів від реалізації результатів дослідження

Кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи.

Спочатку необхідно:

— вказати, скільки часу займе виконання даної наукової роботи та впровадження її результатів.

Приймаємо, що виконання наукової роботи та впровадження її результатів займе 1 рік.

— зазначити, коли саме (після впровадження виконаної наукової роботи) очікуються основні позитивні результати від впровадження дослідження.

Приймаємо, що основні позитивні результати від впровадження дослідження очікуються протягом 3–х років.

— назвати ці позитивні результати та кількісно їх оцінити по роках.

В умовах ринку узагальнюючим позитивним результатом, що його отримує підприємство від впровадження результатів дослідження, є збільшення чистого прибутку підприємства.

Для прогнозування зростання чистого прибутку підприємства можливі два основні випадки. Скористаємося 2–м випадком, коли не можливо прямо оцінити зростання чистого прибутку підприємства від впровадження результатів наукового дослідження. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із 3–х років, протягом яких очікується отримання позитивних результатів від впровадження дослідження, розраховується за формулою [4.10]:

$$\Delta П_i = \sum_{1}^{n} (\Delta Ц_0 \cdot N + Ц_0 \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\vartheta}{100}\right) \text{ [грн]}, \qquad (4.10)$$

де, $\Delta Ц_0$ — покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_0$ — основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ — коефіцієнт, який враховує сплату податку на додану вартість. У 2018 році ставка податку на додану вартість встановить на рівні 20%, а коефіцієнт $\lambda = 0{,}8333$;

ρ — коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати $\rho = 0{,}2 \ldots 0{,}3$. Приймемо $\rho = 0{,}25$;

ϑ — ставка податку на прибуток. У 2018 році $\vartheta = 18\%$.

Припустимо що в результаті впровадження, на території України, результатів наукового дослідження покращується якість розробки, що дозволяє підвищити ціну їх реалізації на 600 грн. Кількість одиниць реалізованої продукції також збільшиться: протягом першого року — на 5000 шт., протягом другого року — ще на 3000 шт., протягом третього року – ще на 1000 шт.

Орієнтовно: реалізація продукції до впровадження результатів розробки складає 12000 шт., а її ціна — 2800 грн.

Тоді збільшення прибутку підприємства $\Delta П_i$ за перший рік складе:

$$\Delta П_1 = [600 \cdot 12000 + (2800 + 600) \cdot 5000] \cdot 0{,}8333 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = \\ = 4134001 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукового дослідження) складе:

$$\Delta П_2 = [600 \cdot 12000 + (2800 + 600) \cdot (5000 + 3000)] \cdot 0{,}8333 \cdot 0{,}25 \cdot \cdot \left(1 - \frac{18}{100}\right) = 5876431 \text{ (грн)}.$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом третього року (відносно базового року, тобто року до впровадження результатів наукового дослідження) складе:

$$\Delta П_3 = [600 \cdot 12000 + (2800 + 600) \cdot (5000 + 3000 + 1000)] \cdot \cdot 0{,}8333 \cdot 0{,}25 \cdot \left(1 - \frac{18}{100}\right) = 6457241 \text{ (грн)}.$$

4.4. Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

1–й крок. Розраховують теперішню вартість інвестицій PV, що вкладають в наукове дослідження. Такою вартістю можемо вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану раніше за формулою (4.11), тобто будемо вважати, що $ЗВ = PV = 90183{,}38 (грн)$.

2–й крок. Розраховують очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство від впровадження результатів наукового дослідження, для кожного із років, починаючи з першого року впровадження. Таке збільшення прибутку також було розраховане раніше за формулою (4.12). За даною формулою: $\Delta П_1 = 4134001 \text{ (грн)}$, $\Delta П_2 = 5876431 \text{ (грн)}$, $\Delta П_3 = 6457241 \text{ (грн)}$.

3–й крок. Для спрощення подальших розрахунків будують вісь часу, на яку наносять всі платежі (інвестиції та прибутки), що мають місце під час виконання науково–дослідної роботи та впровадження її результатів.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рисунку 4.1.

Рисунок 4.1 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

4–й крок. Розраховують абсолютну ефективність вкладених інвестицій $\text{Е}_{\text{абс}}$ за формулою [4.11]:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV)\ [\text{грн}], \tag{4.11}$$

де, ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукового дослідження, грн.;

PV – теперішня вартість інвестицій $PV = \text{ЗВ}$, грн.

Приведена вартість всіх чистих прибутків ПП розраховується за формулою [4.12]:

$$\text{ПП} = \sum_{1}^{\text{Т}} \frac{\Delta\text{П}_i}{(1+\tau)^t}\ [\text{грн}], \tag{4.12}$$

де, $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн.;

Т — період часу, протягом якого виявляються результати впровадження НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівня 0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до точки "0".

$$ПП = \frac{4134001}{(1+0{,}1)^2} + \frac{5876431}{(1+0{,}1)^3} + \frac{6457241}{(1+0{,}1)^4} = 12241961 \text{ (грн)}.$$

Тепер обрахуємо абсолютну ефективність вкладених інвестицій:

$$Е_{абс} = (12241961 - 90183{,}38) = 12151777{,}6 \text{ (грн)}.$$

Оскільки $Е_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток.

5–й крок. Розраховують відносну (щорічну) ефективність вкладених в наукове дослідження інвестицій $Е_в$ за формулою 4.13

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1, \qquad (4.13)$$

де, $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн.;

PV — теперішня вартість інвестицій $PV = ЗВ$, грн.;

$Т_ж$ — життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[4]{1 + \frac{12151777{,}6}{90183{,}38}} - 1 = 2{,}41 \text{ або } 241\%.$$

Отже, відносна (щорічна) ефективність вкладених інвестицій буде становити 319,9%.

Далі ефективність вкладених інвестицій потрібно порівняти з мінімальною (бар'єрною) ставкою дисконтування $\tau_{\text{мін}}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{\text{мін}}$ визначається за формулою:

$$\tau_{\text{мін}} = d + f, \tag{4.14}$$

де, d — середньозважена ставка за депозитними операціями в комерційних банках; $d = (0{,}14 \dots 0{,}2)$;

f — показник, що характеризує ризикованість вкладень, зазвичай величина $f = (0{,}05 \dots 0{,}1)$.

$$\tau_{\text{мін}} = 0{,}2 + 0{,}1 = 0{,}3 \text{ або } 30\%.$$

Як видно з розрахунків $\text{Е}_{\text{в}} = 319{,}9\% > \tau_{\text{мін}} = 30\%$, тому потенційний інвестор буде зацікавлений у фінансуванні даного наукового дослідження, а не у вкладенні коштів на депозит у комерційному банку.

6–й крок. Розраховують термін окупності вкладених у реалізацію наукового проекту інвестицій $\text{Т}_{\text{ок}}$ за формулою [4.15]:

$$\text{Т}_{\text{ок}} = \frac{1}{\text{Е}_{\text{в}}} \text{ [років]}, \tag{4.15}$$

$$\text{Т}_{\text{ок}} = \frac{1}{2{,}41} = 0{,}41 \text{ (року)}$$

Оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій знаходиться в допустимих межах $\text{Т}_{\text{ок}} = 3{,}6$ місяця $< 3 \dots 5 -$ ти років, то фінансування даної наукової розробки є доцільним.

4.6 Висновки до розділу

В даному розділі був проведений попередній розрахунок економічної частини наукової розробки “Web–орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля”. Було проведено оцінювання комерційного потенціалу розробки трьома незалежними експертами та визначено, що рівень комерційного потенціалу розробки вище середнього. Під час розрахунку було складено кошторис витрат, пораховано витрати на заробітну плату розробників та робітників, додаткову заробітну плату всіх розробників та робітників, витрати на нарахування на заробітну плату, витрати на амортизацію, витрати на матеріали та комплектуючі, витрати на електроенергію, та інші додаткові витрати. Пораховано, що загальні витрати на виконання даної частини НДДКР становлять $73048{,}55$ гривень. Також пораховано, що загальна вартість всієї НДДКР становить $81165{,}05$ гривень. Здійснено прогнозування загальних витрат на виконання та впровадження результатів виконаної НДДКР та встановлено, що $ЗВ = 90183{,}38$ гривень.

Проведено прогнозування збільшення чистого прибутку підприємства від впровадження результатів наукового дослідження у кожному з трьох років відносно базового.

Було проаналізовано ефективності вкладених інвестицій та періоду їх окупності. Розрахунок абсолютної ефективності вкладених інвестицій показав, що результат від проведення наукових досліджень та їх впровадження принесе прибуток. Розрахунок відносної (щорічної) ефективності вкладених в наукове дослідження інвестицій показав, що потенційний інвестор буде зацікавлений у фінансуванні даного наукового дослідження, а не у вкладенні коштів на депозит у комерційному банку. Фінансування даного наукового дослідження є доцільним, оскільки термін окупності вкладених у реалізацію наукового проекту інвестицій дорівнює $Т_{ок} = 0{,}41$ року.

ВИСНОВКИ

В даній кваліфікаційній роботі розроблено web–орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля, в якому створено шість основних підсистем, а саме: підсистема логін/пароль, підсистема первинного огляду даних, підсистема пошуку авто, підсистема підбору авто, підсистема виведення даних, підсистема виходу з інформаційної системи. Досконало розглянуто питання доцільності створення даного ресурсу, розглянуто найпоширеніші методи та середовища створення онлайн ресурсу, проведено детальний аналіз та опрацьовано висновки на грунтуванні результатів дослідження можливостей використання технологій HTML5, CSS3, JavaScript, Node.js, MongoDB. Виконано аналіз сучасного стану розвитку інформаційних систем пошуку та підбору даних. Створено технологічний ланцюжок роботи даної інформаційної системи. Покращено процес пошуку та підбору автомобілів за рахунок використання сучасних інформаційних технологій веб–розробки, фреймворків та високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій на 25% за рахунок процесу оптимізації пошуку даних.

У підсумку до даної магістерскьої кваліфікаційної роботи, були проаналізовані та використані головні аспекти проектування, розробки і тестування програмних продуктів, що робить можливим працювати над проектами масштабного рівня. Розроблений сайт відповідає всім вимогам, які були висунуті під час його проектування. Розроблений програмний засіб підбору та пошуку авто є актуальною серед користувачів, проблем у функціонуванні та програмних збоїв немає.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1.Н. А. Прохоренок, В. А. Дронов HTML, JavaScript, PHP и MySQL. Джентльменский набор Web–мастера /БХВ–Петербург. – 2015. – 768с.

2.Трояновська Т. І. Метод покращення візуальним керуванням галереями графічних файлів / Т. І. Трояновська, Л. А. Савицька, І. А. Жарий // Інформаційні технології та комп'ютерна інженерія. – Вінниця, 2016. – №3, с. 33–37.

3.Troianovska T. I. Quality of content delivery in computer specialists training system / Olexiy D. Azarov, Tetiana I. Troianovska, Liudmyla A. Savytska and 7 others // Proc. SPIE 10445, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2017, 104452S(2017/08/07);doi:10.1117/12.2281229;http://dx.doi.org/10.1117/12.2281229 (Scopus).

4.Трояновська Т. І. Методи та засоби популяризації комерційних веб–ресурсів / Т. І. Трояновська, Л. А. Савицька, В. Ю. Тарануха // Інформаційні технології та комп'ютерна інженерія. – Вінниця, 2017. – №2, С. 23–30.

5.Трояновська Т. І. Інформаційна технологія доставки контенту у системах комп'ютеризованої підготовки спеціалістів. // Гороховський О. І., Трояновська Т. І., Азаров О. Д. Монографія. Вінниця : ВНТУ, 2016.– 160 с.

6.PHP: об'єкти, шаблони і методики програмування. / Мет Зандстра // Діалектика–Вільямс, 4 видання, 2019р., с. 576.

7.Захарченко С. М. Застосування односторінкових веб–орієнтованих інтерфейсів в соціально значущих проектах. / С. М. Захарченко, Т. І. Трояновська, О. В. Бойко В. С. Рибаченко // Вісник ХНУ, №3, 2016р., с. 33–39.

8.Трояновська, Т. Алгоритм структурованої візуалізації XML–файлів [Текст] / Трояновська Т. І., Бойко О. В. // „Intrenet–Education–Science” : Міжнародна науково–технічна конференція, 11–14 жовтня 2016 р. – Вінниця : КІВЦ ВНТУ, 2016. – С. 142–144. ISBN 966–641–102–4.

9.Д. Макфарланд Большая книга CSS. / СПб.: Питер, 2016. – 720 с.

10. Трояновська Т. І. Інформаційна технологія побудови системи комп’ютеризованої підготовки спеціалістів / Т. І. Трояновська // «Новини на научния прогрес» : IX Міжнародна науково–практична конференція, 17–25 серпня 2013 р. : тези доповідей. – София : «Бял ГРАД–БГ», 2013. – С. 36–42. – ISBN 978–966–8736–05–6.

11. Аналіз методів та засобів просування веб–ресурсів / Трояновська Т. І., Савицька Л. А.. Тарануха В. Ю., Отришко В. О. // Збірник Матеріалів XLVI Науково–технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2017). Режим доступу: https://conferences.vntu.edu.ua/index.php/all–fitki/all–fitki–2017/paper/view/1971/1536.

12. Troianovska T. I. Adaptive compression methods of data based on Fibonacci linear forms / Volodymyr A. Luzhetsky, Liudmyla A. Savytska, Tetiana I. Troianovska and 4 others // Proc. SPIE 10445, Photonics Applications in Astronomy, Communications, Industry, and High Energy Physics Experiments 2017, 1044524 (2017/08/07); doi: 10.1117/12.2280944; http://dx.doi.org/10.1117/12.2280944 (Scopus).

13. Інформаційна система підбору та замовлення авто / Коробейнікова Т. І., Маренич Б. С. // Збірник Матеріалів XLVIII Науково–технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2019). Режим доступу: https://conferences.vntu.edu.ua/index.php/all–fitki/all–fitki–2019/paper/view/6877/5639.

14. Н. А. Прохоренок, В. А. Дронов HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера /БХВ-Петербург. – 2015. – 768с.

15. Трояновська Т. І. Метод покращення візуальним керуванням галереями графічних файлів / Т. І. Трояновська, Л. А. Савицька, І. А. Жарий // Інформаційні технології та комп’ютерна інженерія. – Вінниця, 2016. – №3, с. 33-37.

ДОДАТОК А

Технічне завдання

Міністерство освіти та науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖЕНО

Завідувач кафедри ОТ

_____проф., д.т.н., О. Д. Азаров

«___»_____2020 р.

Технічне завдання

до магістерської кваліфікаційної роботи на тему:

" Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля "

08–23.МКР.007.00.000 ТЗ

Керівник роботи:

_____ доц., к.т.н. Савицька Л. А.

«___»_____2020 р.

Виконав: студент гр. КІ-19м

_____ Маренич Б. С.

«___»_____2020р.

2020 – Вінниця

1 Назва і галузь застосування

Магістерська кваліфікаційна робота на тему: “Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля”. Галузь застосування – пошук та підбір автомобілів в США.

2 Підстава для розробки

Магістерська кваліфікаційна робота виконується на підставі завдання керівника роботи доц., к.т.н., Савицької Л. А. .

3 Мета і призначення розробки

Метою роботи є покращення процесу пошуку та підбору автомобілів за рахунок використання сучасних інформаційних технологій веб–розробки, фреймворків та високопродуктивної нереляційної бази даних, де процес обробки інформації є швидшим за попереднє покоління технологій за рахунок процесу оптимізації пошуку даних.

Об’єкт: процес оптимізації пошуку актуальних даних при підборі автомобілів.

Предмет: система оптимального підбору авто за заданими параметрами та пошуку на ринку автомобілів.

Для виконання поставленої у магістерській дипломній роботі мети необхідно виконати таке завдання:

— виконати аналіз сучасного стану розвитку інформаційних систем пошуку та підбору даних;

— створити технологічний ланцюжок роботи інформаційної системи підбору та замовлення авто;

— розробити алгоритм роботи даної інформаційної системи;

— вдосконалити процес оптимізації пошуку даних;

— детальне тестування розробленого програмного забезпечення.

4 Джерела розробки

Аналіз методів та засобів просування веб-ресурсів / Трояновська Т. І., Савицька Л. А.. Тарануха В. Ю., Отришко В. О. // Збірник Матеріалів XLVI Науково-технічної конференції факультету інформаційних технологій та комп'ютерної інженерії (2017). Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2017/paper/view/1971/1536.

5 Вимоги до програмного засобу

5.1 Вимоги до характеристик інформаційної системи. Інформаційна система повинна забезпечити:

— можливість працювати з усіма сучасними браузерами;

— створення дружнього до користувача інтерфейсу;

— реалізовувати взаємодію між компонентами за загальним алгоритмом роботи системи;

— реалізовувати алгоритм пошуку команд;

— швидко надавати потрібну інформацію;

— інформаційна система повинна мати лише актуальну інформацію;

— кросплатформеність програмного забезпечення.

5.2 Вимоги до інтерфейсу.

— розробити інтерфейс дружнім до користувача;

— розробити інтерфейс здатним відображати зміни при зміні розширення екрану;

— коректне натиснення на кнопки;

— правильний перехід на інші сторінки;

— швидке відображення медіа даних.

6 Стадії та етапи розробки

6.1 Робота виконується в три етапи, таблиця 6.1.

Таблиця 6.1 – Етапи виконання роботи.

Етап	Зміст	Початок	Кінець	Результат
1	Інформаційний пошук та огляд літературних джерел			Розділи 1
2	Розробка інформаційної системи			Розділ 2
3	Тестування програмного засобу			Розділ 3
4	Підготовка матеріалів пояснювальної записки			Пояснювальна записка.

7 Порядок контролю та прийому

7.1 До приймання магістерської квалфікаційної роботи надається:

— пояснювальна записка з відповідними узгодженнями;
— демонстраційні плакати;
— відгук керівника роботи;
— стороння рецензія.

7.2 Для проведення захисту магістерської квалфікаційної роботи утворюється державна екзаменаційна комісія.

Технічне завдання до виконання прийняв _____ Маренич Богдан Сергійович

ДОДАТОК Б

Лістинг програми

Backend:

Car.js

```
const mongoose = require("mongoose")
const mongoosePaginate = require("mongoose-paginate-v2")
const Schema = mongoose.Schema
const Car = new Schema({
    id: Schema.ObjectId,
    date: Date,
    image: String,
    price: Number,
    title: String,
    capacity: Number,
    type: { type: Schema.Types.ObjectId, ref: "Type" },
    company: { type: Schema.Types.ObjectId, ref: "Company" },
    fuel: { type: Schema.Types.ObjectId, ref: "Fuel" },
    region: { type: Schema.Types.ObjectId, ref: "Region" },
    transmission: { type: Schema.Types.ObjectId, ref: "Transmission" },
})
Car.plugin(mongoosePaginate)
mongoose.model("Car", Car)
```

Company.js

```
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Company = new Schema({
    id: Schema.ObjectId,
```

```
    name: String,
})
mongoose.model("Company", Company)
```

Contact.js

```
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Contact = new Schema({
    id: Schema.ObjectId,
    date: Date,
    name: String,
    email: String,
})
mongoose.model("Contact", Contact)
```

Fuel.js

```
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Fuel = new Schema({
    id: Schema.ObjectId,
    name: String,
})
mongoose.model("Fuel", Fuel)
```

Region.js

```
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Region = new Schema({
    id: Schema.ObjectId,
    name: String,
})
```

```
mongoose.model("Region", Region)

Transmission.js
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Transmission = new Schema({
    id: Schema.ObjectId,
    name: String,
})
mongoose.model("Transmission", Transmission)

Type.js
const mongoose = require("mongoose")
const Schema = mongoose.Schema
const Type = new Schema({
    id: Schema.ObjectId,
    name: String,
})
mongoose.model("Type", Type)

index.js
const routes = require("express").Router()
const mongoose = require("mongoose")
const Car = mongoose.model("Car")
const Contact = mongoose.model("Contact")
routes.get("/api/cars/:id", async (req, res) => {
    try {
        const car = await Car.findById(req.params.id).populate(["company", "fuel",
"region", "transmission", "type"])
        res.status(200).json(car)
```

```
    } catch (err) {
        console.log(err)
        res.status(404).json({ message: "Not found" })
    }
})
routes.get("/api/cars", async (req, res) => {
    const query = {}
    const keys = Object.keys(req.query)
    keys.forEach(param => {
        if (
            param !== "page" &&
            param !== "limit" &&
            param !== "sort" &&
            param !== "search" &&
            param !== "minCapacity" &&
            param !== "maxCapacity"
        ) {
            const value = req.query[param]
            if (value) {
                query[param] = { $in: value }
            }
        }
    })
    const minCapacity = req.query.minCapacity
    const maxCapacity = req.query.maxCapacity
    if (minCapacity || maxCapacity) {
        query.capacity = { $gt: minCapacity || 0, $lt: maxCapacity || 6000 }
    }
    const search = req.query.search || ""
    const page = req.query.page || 1
```

```
    const limit = req.query.limit || 12
    const sort = req.query.sort || "date"
    if (search) {
        query.title = new RegExp(search, "i")
    }
    const data = await Car.paginate(query, {
        customLabels: { docs: "items" },
        page,
        limit,
        sort,
        populate: ["company", "fuel", "region", "transmission", "type"],
    })
    res.status(200).json(data)
})
routes.get("/api/enums", async (req, res) => {
    try {
        const modelsNames = mongoose.modelNames()
        const enums = {}
        for (const moduleName of modelsNames) {
            if (moduleName !== "Car") {
                enums[moduleName.toLowerCase()] = await
mongoose.model(moduleName).find()
            }
        }
        res.status(200).json(enums)
    } catch (err) {
        console.log(err)
    }
})
routes.post("/api/contact", async (req, res) => {
```

```
    try {
        const contact = await Contact.create(req.body)
        res.status(200).json(contact)
    } catch (err) {
        console.log(err)
        res.status(404).json({ message: "Error" })
    }}) module.exports = routes
```

express.js

```
const bodyParser = require("body-parser")
const routes = require("./routes")
module.exports = function(app) {
    app.use(bodyParser.json())
    app.use(bodyParser.urlencoded({ extended: true }))
    app.use("/", routes)
}
```

package.json

```
{
    "name": "backend",
    "version": "1.0.0",
    "main": "index.js",
    "license": "MIT",
    "scripts": {
        "dev": "nodemon server.js"
    },
    "dependencies": {
        "body-parser": "^1.19.0",
        "express": "^4.16.4",
        "mongoose": "^5.5.7",
        "mongoose-paginate-v2": "^1.2.1"
```

```
  },
  "devDependencies": {
    "nodemon": "^1.19.0"
  }}
server.js
const fs = require("fs")
const join = require("path").join
const express = require("express")
const app = express()
const mongoose = require("mongoose")
const models = join(__dirname, "models")
const db = mongoose.connection
const pathToImages = "/images/cars/"
fs.readdirSync(models)
  .filter(file => ~file.search(/^[^.].*\.js$/))
  .forEach(file => require(join(models, file)))
require("./express")(app)
connect()
function randomInteger(min, max) {
  var rand = min - 0.5 + Math.random() * (max - min + 1)
  rand = Math.round(rand)
  return rand
}
async function listen() {
  try {
    const data = {
      Company: ["BMW", "Mercedes", "Tesla", "Ford", "Nissan", "Skoda"],
      Fuel: ["Бензин", "Дизпаливо", "Газ", "Електрика"],
      Region: [
        "Одеська",
```

```
            "Дніпропетровська",
            "Чернігівська",
            "Харківська",
            "Житомирська",
            "Полтавська",
            "Херсонська",
            "Київська",
            "Запорізька",
            "Луганська",
            "Донецька",
            "Вінницька",
            "Крим",
            "Миколаївська",
            "Кіровоградська",
            "Сумська",
            "Львівська",
            "Черкаська",
            "Хмельницька",
            "Волинська",
            "Рівненська",
            "Івано-Франківська",
            "Тернопільська",
            "Закарпатська",
            "Чернівецька",
        ],
        Transmission: ["Механічна", "Автоматизована", "Роботизована",
"Варіаторная"],
        Type: ["Легкові", "Вантажні", "Позашляховики", "Пікапи", "Спортивні"],
    }
    const collections = await mongoose.connection.db.collections()
```

```
for (let collection of collections) {
    await collection.drop()
}
const modelsNames = mongoose.modelNames()
const ids = {}
for (const moduleName of modelsNames) {
    if (moduleName !== "Car" && moduleName !== "Contact") {
        const getModel = mongoose.model(moduleName)
        for (const item of data[moduleName]) {
            await getModel.create({ name: item })
            ids[moduleName] = await getModel.distinct("_id")
        }
    }
}
const carsLimit = 300
let startDate = new Date()
for (let i = 0; i <= carsLimit; i++) {
    const generateDate = new Date(startDate.getTime() - 5 * 24 * 60 * 60 * 1000)
    mongoose.model("Car").create({
        date: generateDate,
        image: pathToImages + randomInteger(1, 31) + ".jpg",
        price: randomInteger(5000, 100000),
        title:
            Math.random()
                .toString(36)
                .substring(2, 15) +
            Math.random()
                .toString(36)
                .substring(2, 15),
        type: ids.Type[Math.floor(Math.random() * ids.Type.length)],
```

```
            company: ids.Company[Math.floor(Math.random() *
ids.Company.length)],
            capacity: randomInteger(1000, 6000),
            fuel: ids.Fuel[Math.floor(Math.random() * ids.Fuel.length)],
            region: ids.Region[Math.floor(Math.random() * ids.Region.length)],
            transmission: ids.Transmission[Math.floor(Math.random() *
ids.Transmission.length)],
        })
        startDate = generateDate
    }
    app.listen(8000)
    console.log("App listening on port 8000!")
  } catch (err) {
    console.log(err)
  }
}
function connect() {
  db.on("error", console.log)
    .on("disconnected", connect)
    .once("open", listen)
  return mongoose.connect("mongodb://localhost:27017/db", { useNewUrlParser:
true })
}
```

Frontend:

index.html

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Авто</title>
```

```
        <meta charset="utf-8" />
        <meta name="viewport" content="width=device-width, initial-scale=1.0, shrink-to-fit=no" />
        <meta http-equiv="X-UA-Compatible" content="IE=edge" />
        <meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
        <meta http-equiv="Cache-Control" content="no-cache" />
        <link href="https://fonts.googleapis.com/css?family=Roboto:300,400,500,700" rel="stylesheet" />
        <link rel="stylesheet" href="https://cdn.materialdesignicons.com/3.4.93/css/materialdesignicons.min.css" />
    </head>
    <body>
        <div id="app"></div>
    <script type="text/javascript" src="/js/main.bundle.7d2d8c23344f8872db08.js"></script></body>
</html>
```

index.js

```
import { createBrowserHistory } from "history"
import { Provider } from "mobx-react"
import React from "react"
import { render } from "react-dom"
import { Router } from "react-router-dom"
import App from "./components/App"
import "./sass/general.scss"
import store from "./store"
const history = createBrowserHistory()
const app = document.getElementById("app")
render(
    <Provider {...store}>
```

```
        <Router history={history}>
            <App />
        </Router>
    </Provider>,
    app,
)
```

Header.js

```
import { AppBar, InputBase, Toolbar } from "@material-ui/core/"
import { makeStyles } from "@material-ui/core/styles"
import { fade } from "@material-ui/core/styles/colorManipulator"
import SearchIcon from "@material-ui/icons/Search"
import React from "react"
import { Link, NavLink } from "react-router-dom"
const useStyles = makeStyles(theme => ({
    appBar: {
        zIndex: 1100,
    },
    menuButton: {
        marginRight: theme.spacing(2),
    },
    hide: {
        display: "none",
    },
    link: {
        color: "rgba(255,255,255,.7)",
        fontWeight: 500,
        marginLeft: 15,
        "&:hover": {
            color: "#fff",
```

```
    },
},
search: {
    position: "relative",
    borderRadius: theme.shape.borderRadius,
    backgroundColor: fade(theme.palette.common.white, 0.15),
    "&:hover": {
        backgroundColor: fade(theme.palette.common.white, 0.25),
    },
    marginRight: theme.spacing(2),
    marginLeft: 0,
    width: "100%",
    [theme.breakpoints.up("sm")]: {
        marginLeft: theme.spacing(3),
        width: "auto",
    },
},
searchIcon: {
    width: theme.spacing(7),
    height: "100%",
    position: "absolute",
    pointerEvents: "none",
    display: "flex",
    alignItems: "center",
    justifyContent: "center",
},
inputRoot: {
    color: "inherit",
},
inputInput: {
```

```
        padding: theme.spacing(1, 1, 1, 7),
        width: "100%",
        [theme.breakpoints.up("md")]: {
            width: 200,
        },
    },
}))
const Header = ({ cars }) => {
    const classes = useStyles()
    const open = window.location.pathname === "/cars"
    return (
        <AppBar position="fixed" style={{ backgroundColor: "#039be5" }}>
            <Toolbar style={{ display: "flex", justifyContent: "space-between" }}>
                <Link to="/" style={{ color: "#fff", fontWeight: 600, fontSize: 24, display:
"flex", alignItems: "center" }}>
                    <img src="/images/logo.png" style={{ width: 50, maxWidth: "100%",
height: 50, padding: 10 }}/> <span style={{ marginLeft: 10 }}>USAuto</span>
                </Link>
                <div style={{ marginLeft: 58, width: "100%", display: "flex", flexFlow:
"row wrap" }}>
                    <NavLink onClick={!open ? cars.resetParams : null} to="/cars"
className={classes.link}>
                        Знайти авто
                    </NavLink>
                    <NavLink to="/contacts" className={classes.link}>
                        Контакти
                    </NavLink>
                </div>
                <div className={classes.search} style={{ display: open ? "block" : "none"
}}>
```

```
                <div className={classes.searchIcon}>
                  <SearchIcon />
                </div>
                <InputBase
                  placeholder="Пошук…"
                  classes={{
                    root: classes.inputRoot,
                    input: classes.inputInput,
                  }}
                  onChange={cars.onSearch}
                />
              </div>
            </Toolbar>
          </AppBar>
  )}
export default Header
```

index.js

```
import { inject, observer } from "mobx-react"
import React, { Suspense } from "react"
import { Route, withRouter } from "react-router-dom"
import Header from "./Header"
import Routing from "./Routing"
@inject("cars")
@withRouter
@observer
class App extends React.Component {
  componentDidMount() {
    const { cars } = this.props
    cars.fetchEnums()
  }
```

```
    render() {
        const { cars } = this.props
        const routesMap = Routing.map(route => (
            <Route key={route.path} path={route.path} exact={route.exact} render={()
=> <route.component />} />
        ))
        return (
            <React.Fragment>
                <Header cars={cars} />
                <Suspense fallback={<div />}>{routesMap}</Suspense>
            </React.Fragment>
        ) }
}
export default App
```

Routing.js

```
import { lazy } from "react"
const Home = lazy(() => import("../Home"))
const Cars = lazy(() => import("../Cars"))
const Detail = lazy(() => import("../Cars/Detail"))
const Contacts = lazy(() => import("../Contacts"))
const Routing = [
    {
        path: "/",
        exact: true,
        component: Home,
    },
    {
        path: "/cars",
        exact: true,
```

```
        component: Cars,
    },
    {
        path: "/cars/:id",
        exact: true,
        component: Detail,
    },
    {
        path: "/contacts",
        exact: true,
        component: Contacts,
    },
]
export default Routing
```

index.js

```
import { inject, observer } from "mobx-react"
import React from "react"
import { withRouter } from "react-router-dom"
import Aside from "./Aside"
import Items from "./Items"
@inject("cars")
@withRouter
@observer
class Cars extends React.Component {
    componentDidMount() {
        const { cars } = this.props
        if (!cars.cars.length) {
            cars.fetchCars()
        }
```

```
    window.addEventListener("scroll", this.onScroll)
  }
  componentWillUnmount() {
    window.removeEventListener("scroll", this.onScroll)
  }
  onScroll = () => {
    const { cars } = this.props
    if (cars.loading) return
    const { scrollTop } = document.documentElement
    const height = document.body.scrollHeight - window.innerHeight
    const scrollPercentage = scrollTop / height

    if (scrollPercentage > 0.8 && cars.page < cars.totalPages) {
      cars.fetchCars(true)
    }
  }
  render() {
    const { cars } = this.props
    return (
      <div style={{ display: "flex" }}>
        <Aside cars={cars} />
        <Items cars={cars} />
      </div>
    )
  }
}
export default Cars
```

index.js

```
import {
```

```
    Checkbox,
    Drawer,
    FormControl,
    FormControlLabel,
    FormGroup,
    FormLabel,
    Input,
    InputLabel,
    ListItemText,
    MenuItem,
    Select,
    TextField,
} from "@material-ui/core/"
import { makeStyles } from "@material-ui/core/styles"
import React from "react"
import { Observer } from "mobx-react"
const drawerWidth = 300
const useStyles = makeStyles(theme => ({
    drawer: {
        width: drawerWidth,
        flexShrink: 0,
        zIndex: 1099,
    },
    drawerPaper: {
        width: drawerWidth,
        top: 60,
        bottom: 0,
        right: 0,
        paddingTop: 30,
        height: "initial",
```

```
    },
    formControl: {
        marginBottom: 30,
        marginLeft: 15,
        marginRight: 15,
    },
    textField: {
        width: "100%",
    },
}))
const sortMap = [
    { _id: "-price", name: "Від дорогих до дешевих" },
    { _id: "price", name: "Від дешевих до дорогих" },
    { _id: "-date", name: "Спочатку нові" },
    { _id: "date", name: "Спочатку старі" },
]
const selectProps = {
    multiple: true,
    input: <Input id="select-multiple-checkbox" />,
    renderValue: selected => selected.map(item => item.name).join(", "),
    style: { width: "100%" },}
const createSelectItems = (array, checkItems) =>
    array.map(item => (
        <MenuItem key={item._id} value={item}>
            <Checkbox checked={checkItems.some(checkItem => checkItem._id ===
item._id)} />
            <ListItemText primary={item.name} />
        </MenuItem>
    ))
const Aside = props => {
```

```
    const classes = useStyles()
    return <Observer render={() => render(props, classes)} />
}
const render = (props, classes) => {
    const { cars } = props
    const {
        currentCompany,
        currentFuel,
        currentRegion,
        currentTransmission,
        currentSort,
        onCompany,
        onFuel,
        onRegion,
        onTransmission,
        onSort,
        onMinCapacity,
        onMaxCapacity,
        onType,
        enums,
    } = cars
    const open = window.location.pathname === "/cars"
    const type = enums.type.map(item => (
        <FormControlLabel
            key={item._id}
            control={<Checkbox checked={item.checked}
onChange={onType(item._id)} value={item._id} />}
            label={item.name}
        />
    ))
```

```
    return (
        <Drawer
            className={classes.drawer}
            variant="persistent"
            anchor="left"
            open={open}
            classes={{
                paper: classes.drawerPaper,
            }}
        >
            <FormControl className={classes.formControl}>
                <InputLabel htmlFor="select-multiple-
checkbox">Відсортувати</InputLabel>
                <Select value={currentSort} onChange={onSort} style={{ width: "100%"
}}>
                    {sortMap.map(item => (
                        <MenuItem key={item._id} value={item._id}>
                            {item.name}
                        </MenuItem>
                    ))}
                </Select>
            </FormControl>
            <FormControl className={classes.formControl}>
                <InputLabel htmlFor="select-multiple-checkbox">Область</InputLabel>
                <Select {...selectProps} value={currentRegion} onChange={onRegion}>
                    {createSelectItems(enums.region, currentRegion)}
                </Select>
            </FormControl>
            <FormControl className={classes.formControl}>
```

```
            <InputLabel htmlFor="select-multiple-
checkbox">Виробник</InputLabel>
            <Select {...selectProps} value={currentCompany}
onChange={onCompany}>
                {createSelectItems(enums.company, currentCompany)}
            </Select>
        </FormControl>
        <FormControl component="fieldset" className={classes.formControl}>
            <FormLabel component="legend">Тип</FormLabel>
            <FormGroup>{type}</FormGroup>
        </FormControl>
        <FormControl className={classes.formControl}>
            <FormLabel component="legend">Об'єм двигуна (см3)</FormLabel>
            <div className={classes.container}>
                <TextField
                    onChange={onMinCapacity}
                    label="Від"
                    type="number"
                    className={classes.textField}
                    InputLabelProps={{
                        shrink: true,
                    }}
                    margin="normal"
                />
                <TextField
                    onChange={onMaxCapacity}
                    type="number"
                    label="До"
                    className={classes.textField}
                    InputLabelProps={{
```

```
                    shrink: true,
                }}
                margin="normal"
              />
            </div>
          </FormControl>
          <FormControl className={classes.formControl}>
            <InputLabel htmlFor="select-multiple-checkbox">Коробка передач</InputLabel>
            <Select {...selectProps} value={currentTransmission} onChange={onTransmission}>
              {createSelectItems(enums.transmission, currentTransmission)}
            </Select>
          </FormControl>
          <FormControl className={classes.formControl}>
            <InputLabel htmlFor="select-multiple-checkbox">Паливо</InputLabel>
            <Select {...selectProps} value={currentFuel} onChange={onFuel}>
              {createSelectItems(enums.fuel, currentFuel)}
            </Select>
          </FormControl>
        </Drawer>
    )}
export default Aside
```

index.js

```
import moment from "moment"
import React from "react"
import { inject, observer } from "mobx-react"
import { withRouter } from "react-router-dom"
import styles from "./index.scss"
```

```
import numbro from "numbro"
const formateDate = date => moment(date).format("DD.MM.YYYY")
const stylesOfImage = url => ({
    display: "block",
    backgroundSize: "cover",
    backgroundRepeat: "no-repeat",
    backgroundPosition: "center",
    height: 300,
    backgroundImage: `url(${url})`,
})
@inject("cars")
@withRouter
@observer
class Detail extends React.Component {
    componentDidMount() {
        const { cars, match } = this.props
        cars.fetchCar(match.params.id)
    }
    render() {
        const { cars } = this.props
        const { _id, date, price, title, type, company, capacity, fuel, region, transmission,
image } = cars.car
        return (
            <div className={styles.wrapper}>
                <div className={styles.content}>
                    <div style={stylesOfImage(image)} />

                    <div className={styles.info}>
                        <div className={styles.info__top}>
                            <h2>{title}</h2>
```

```
                    <span
className={styles.price}>{numbro(price).formatCurrency()}</span>
                </div>
                <span>Дата публікації: {formateDate(date)}</span>
                <span>Виробник: {company && company.name}</span>
                <span>Модель: {type && type.name}</span>
                <span>Об'єм двигуна: {capacity}</span>
                <span>Паливо: {fuel && fuel.name}</span>
                <span>Область: {region && region.name}</span>
                <span>Коробка передач: {transmission &&
transmission.name}</span>
              </div>
            </div>
          </div>
      )
   }
}
export default Detail
```

index.scss

```
.wrapper {
   display: flex;
   justify-content: center;
   padding: 30px;
}
.content {
   max-width: 700px;
   width: 100%;
   margin-top: 80px;
   background: #fff;
```

```
    box-shadow: 0px 1px 3px 0px rgba(0, 0, 0, 0.2), 0px 1px 1px 0px rgba(0, 0, 0,
0.14),
        0px 2px 1px -1px rgba(0, 0, 0, 0.12);
    border-radius: 4px;
}
.info {
    display: flex;
    flex-direction: column;
    padding: 30px;
    span {
        margin-bottom: 10px;
    }
}
.info__top {
    display: flex;
    justify-content: space-between;
    align-items: center;
}
.price {
    font-size: 18px;
    font-weight: 500;
    color: #409422;
}
```

index.js

```
import { Grid } from "@material-ui/core/"
import { makeStyles } from "@material-ui/core/styles"
import clsx from "clsx"
import React from "react"
import Car from "./Car"
```

```
import { Observer } from "mobx-react"
const drawerWidth = 240
const useStyles = makeStyles(theme => ({
    drawerHeader: {
        display: "flex",
        alignItems: "center",
        padding: "0 8px",
        ...theme.mixins.toolbar,
        justifyContent: "flex-end",
    },
    content: {
        flexGrow: 1,
        padding: theme.spacing(3),
        marginLeft: -drawerWidth,
    },
    contentShift: {
        marginLeft: 0,
    },
}))
const render = (cars, classes) => {
    const open = window.location.pathname === "/cars"
    return (
        <main
            className={clsx(classes.content, {
                [classes.contentShift]: open,
            })}
        >
            <div className={classes.drawerHeader} />
            <Grid direction="row" container spacing={2}>
                {cars.cars.map(car => (
```

```
                <Car key={car._id} {...car} />
            ))}
        </Grid>
    </main>
  )
}
const Items = ({ cars }) => {
  const classes = useStyles()
  return <Observer render={() => render(cars, classes)} />
}
export default Items
```

index.js

```
import { Button, Card, CardActionArea, CardActions, CardContent, CardMedia,
Grid, Typography } from "@material-ui/core"
import { makeStyles } from "@material-ui/core/styles"
import moment from "moment"
import numbro from "numbro"
import React from "react"
import { Link } from "react-router-dom"
const formateDate = date => moment(date).format("DD.MM.YYYY")

const useStyles = makeStyles({
  media: {
    height: 200,
  },
  price: {
    fontSize: 18,
    fontWeight: 500,
    color: "#409422",
```

```
  },})
const Car = props => {
  const { _id, date, price, title, type, company, capacity, fuel, region, transmission,
image } = props
  const classes = useStyles()
  const simpleText = { color: "rgba(0, 0, 0, 0.54)" }
  return (
    <Grid item xs={12} sm={12} md={6} lg={4} xl={3}>
      <Card className={classes.card}>
        <CardActionArea>
          <Link to={`/cars/${_id}`}>
            <CardMedia className={classes.media} image={image} title={title}
/>
            <CardContent>
              <Typography
                gutterBottom
                variant="h5"
                component="h2"
                style={{ color: "#282828", fontWeight: 500 }}
              >
                {title}
              </Typography>
              <Typography component="p" className={classes.price}>
                {numbro(price).formatCurrency()}
              </Typography>
              <Typography component="p" style={simpleText}>
                {region.name} / {formateDate(date)}
              </Typography>
              <Typography component="p" style={simpleText}>
                <b>{company.name}</b> | {transmission.name} | {fuel.name}
```

```
                    </Typography>
                </CardContent>
            </Link>
        </CardActionArea>
        <CardActions>
            <Link to={`/cars/${_id}`}>
                <Button size="small" color="primary">
                    Перегляд
                </Button>
            </Link>
        </CardActions>
    </Card>
</Grid>
)
}
export default Car
```

index.js

```
import { Button, Card, TextField, Modal, Typography } from "@material-ui/core"
import { inject, observer } from "mobx-react"
import React from "react"
import { withRouter } from "react-router-dom"

const styleOfModal = {
    position: "absolute",
    width: 400,
    backgroundColor: "#fff",
    boxShadow: "0 5px 25px rgba(0,0,0,.2)",
    padding: 30,
    outline: "none",
```

```
    top: "50%",
    left: "50%",
    transform: "translate(-50%, -50%)",}
@inject("cars")
@withRouter
@observer
class Contacts extends React.Component {
    state = {
        name: "",
        email: "",
        error: "",
        open: false,
    }
    submit = () => {
        const { cars } = this.props
        const { name, email } = this.state
        if (!name || !email) {
            this.setState({
                error: "Всі поля обов'язкові",
            })
            return
        }
        cars.saveContact({ name, email, date: new Date() }, () => {
            this.setState({
                name: "",
                email: "",
                open: true,
            })
        })
    }
```

```
setFormItem = e => {
    const { value, id } = e.target
    this.setState({
        [id]: value,
        error: "",
    })
}
handleClose = () => {
    this.setState({
        open: false,
    })
}
render() {
    const { name, email, error, open } = this.state
    return (
        <div
            style={{
                display: "flex",
                flexDirection: "column",
                justifyContent: "center",
                alignItems: "center",
                flex: 1,
                height: "100%",
                width: "100%",+
                height: "100%",
                backgroundImage: `url("/images/contacts.jpg")`,
                backgroundSize: "cover",
                backgroundRepeat: "no-repeat",
                backgroundPosition: "center",
                opacity: ".78",
```

```
        }}
      >
        <Card style={{ maxWidth: 400, width: "100%", padding: 30 }}>
          <div style={{ display: "flex", flexDirection: "column", flex: 1 }}>
            <TextField
              id="name"
              value={name}
              onChange={this.setFormItem}
              style={{ width: "100%" }}
              label="Як до Вас звертатися?*"
              type="text"
              margin="normal"
            />
            <TextField
              id="email"
              value={email}
              onChange={this.setFormItem}
              style={{ width: "100%" }}
              label="Електронна пошта*"
              type="text"
              margin="normal"
            />
          </div>

          <div style={{ fontSize: 16, color: "red", marginTop: 30
}}>{error}</div>
          <div style={{ display: "flex", justifyContent: "flex-end", marginTop: 30
}}>
            <Button variant="contained" color="primary" onClick={this.submit}>
              Надіслати
```

```
                </Button>
              </div>
            </Card>
            <Modal
              aria-labelledby="simple-modal-title"
              aria-describedby="simple-modal-description"
              open={open}
              onClose={this.handleClose}
            >
              <div style={styleOfModal}>
                <Typography variant="h6" id="modal-title">
                  Заявка створена
                </Typography>
                <Typography variant="subtitle1" id="simple-modal-description">
                  Протягом дня з вами зв'яжеться наш оператор
                </Typography>
              </div>
            </Modal>
          </div>
        )
    }
}
export default Contacts
```

index.js

```
import { inject, observer } from "mobx-react"
import React from "react"
import { withRouter } from "react-router-dom"
import styles from "./index.scss"
@inject("cars")
```

```
@withRouter
@observer
class Home extends React.Component {
   render() {
      const styleOfCard = {
         paddig: 30,
         textAlign: "left",
         width: "100%",
         background: "#fff",
         color: "#386282",
         marginBottom: 30,
      }
      return (
         <div
            style={{
               display: "flex",
               justifyContent: "center",
               alignItems: "center",
               flexDirection: "column",
               flex: 1,
               width: "100%",
               height: "100%",
               backgroundImage: `url("/images/bg.jpg")`,
               backgroundSize: "cover",
               backgroundRepeat: "no-repeat",
               backgroundPosition: "center",
               opacity: ".78",
            }}
            className={styles.wrapper}
         >
```

```
            <div style={{
               display: "flex",
               width: "100%",
               justifyContent: "center"
            }}>
               <div
                  style={{
                     maxWidth: 800,
                     width: "100%",
                     background: "#fff",
                     borderRadius: 30,
                     padding: 30,
                     margin: 30,
                     boxShadow: "0 14px 90px rgba(0,0,0,.6)",
                     position: "relative",
                  }}
               >
                  <h2 style={{ color: "#105285", fontSize: 34, textAlign: "center"
}}>Чому ми?</h2>
                  <div style={{ display: "flex", flexFlow: "row wrap" }}>
                     <div style={styleOfCard}>Величезний вибір автомобілів</div>
                     <div style={styleOfCard}>Юридичний супровід угоди купівлі-
продажу</div>
                     <div style={styleOfCard}>
                        Всі машини пройшли огляд, у вас буде повна інформація про
автомобіль
                     </div>
                     <div style={styleOfCard}>Всі авто перевірені кваліфікованими
фахівцями</div>
                     <div style={styleOfCard}>
```

```
                    Ми позбавляємо своїх клієнтів від необхідності вести
переговори про ціну
                    </div>
                </div>
                <div style={{
                    position: "absolute",
                    right: 50,
                    top: 60,
                }}>
                    <img style={{
                        width: 200,
                    }} src="/images/home_page_1.png"
className={styles.right__image}/>
                </div>
            </div>
        </div>
    )
  }
}
export default Home
```

index.scss

```
@mixin for-size($size) {
  @if $size == phone-only {
    @media (max-width: 599px) { @content; }
  } @else if $size == tablet-portrait-up {
    @media (min-width: 600px) { @content; }
  } @else if $size == tablet-landscape-up {
    @media (min-width: 900px) { @content; }
```

```
    } @else if $size == desktop-up {
        @media (min-width: 1200px) { @content; }
    } @else if $size == big-desktop-up {
        @media (min-width: 1800px) { @content; }
    }
}
.right__image {
    @include for-size(phone-only) {
        display: none;
    }
    @include for-size(tablet-portrait-up) {
        display: none;
    }
    @include for-size(tablet-landscape-up) {
        display: block;
    }
}
.wrapper {
    @include for-size(phone-only) {
        padding-top: 0px;
    }
    @include for-size(tablet-portrait-up) {
        padding-top: 0px;
    }
    @include for-size(tablet-landscape-up) {
        padding-top: 0px;
    }
}
```

general.scss

```
:global {
```

```
    html,
    body {
        margin: 0;
        padding: 0;
    }
    html {
        height: 100%;
        min-height: 100%;
    }
    body {
        width: 100%;
        height: 100%;
        min-height: 100%;
        background: hsl(0, 0%, 98%);
        font-family: "Roboto", sans-serif;
    }
    ul {
        list-style: none;
    }
    a {
        text-decoration: none;
    }
    .active {
        color: #fff !important;
    }
    #app {
        width: 100%;
        height: 100%;
        min-height: 100%;    }}
```

index.js

```
import axios from "axios"
export const get = (url, params) => {
    return axios.get(url, { params })
}
export const post = (url, payload) => {
    return axios.post(url, payload)
}
export const put = (url, payload) => {
    return axios.put(url, payload)
}
export const patch = (url, payload) => {
    return axios.patch(url, payload)
}
export const del = (url, payload) => {
    return axios.delete(url, { data: payload })
}
```

cars.js

```
import { flow, observable } from "mobx"
import { get, post } from "../api"
const API_URL = "/api"
class cars {
    @observable loading = false
    @observable cars = []
    @observable car = {}
    @observable enums = {
        company: [],
        fuel: [],
        region: [],
        transmission: [],
```

```
    type: [],
}
@observable minCapacity = ""
@observable maxCapacity = ""
@observable page = 1
@observable limit = 0
@observable totalPages = 0
@observable search = ""
@observable currentRegion = []
@observable currentCompany = []
@observable currentTransmission = []
@observable currentFuel = []
@observable currentSort = "-date"
getIds = array => array.map(item => item._id)
fetchCars = flow(function*(scroll) {
    this.loading = true
    try {
        const params = {
            region: this.getIds(this.currentRegion) || null,
            company: this.getIds(this.currentCompany) || null,
            transmission: this.getIds(this.currentTransmission) || null,
            fuel: this.getIds(this.currentFuel) || null,
            type: this.getIds(this.enums.type.filter(item => item.checked)) || null,
            minCapacity: this.minCapacity || null,
            maxCapacity: this.maxCapacity || null,
            page: scroll ? this.page + 1 : this.page,
            search: this.search || null,
            sort: this.currentSort,
        }
        const res = yield get(`${API_URL}/cars`, params)
```

```
        this.cars = params.page > 1 ? [...this.cars, ...res.data.items] : res.data.items
        this.page = res.data.page
        this.limit = res.data.limit
        this.totalPages = res.data.totalPages
        this.loading = false
    } catch (err) {
        this.loading = false
        console.log(err)
    }
})
fetchCar = flow(function*(id) {
    this.loading = true
    try {
        const res = yield get(`${API_URL}/cars/${id}`)
        this.car = res.data
        this.loading = false
    } catch (err) {
        this.loading = false
        console.log(err)
    }
})
fetchEnums = flow(function*() {
    this.loading = true
    try {
        const res = yield get(`${API_URL}/enums`)
        res.data.type.forEach(item => {
            item.checked = false
            return item
        })
        this.enums = res.data
```

```
        this.loading = false
    } catch (err) {
        this.loading = false
        console.log(err)
    }
})
saveContact = flow(function*(payload, cb) {
    this.loading = true
    try {
        yield post(`${API_URL}/contact`, payload)
        this.loading = false
        if (cb) cb()
    } catch (err) {
        this.loading = false
        console.log(err)  }  })
onSearch = event => {
    this.search = event.target.value
    this.preFetchCars()
}
onRegion = event => {
    this.currentRegion = event.target.value
    this.preFetchCars()
}
onCompany = event => {
    this.currentCompany = event.target.value
    this.preFetchCars()
}
onTransmission = event => {
    this.currentTransmission = event.target.value
    this.preFetchCars()
```

```
}
onFuel = event => {
    this.currentFuel = event.target.value
    this.preFetchCars()
}
onSort = event => {
    this.currentSort = event.target.value
    this.preFetchCars()
}
onType = id => event => {
    this.enums.type = this.enums.type.map(item => {
        if (id === item._id) {
            item.checked = event.target.checked
        }
        return item
    })
    this.preFetchCars()
}
onMinCapacity = event => {
    this.minCapacity = event.target.value
    this.preFetchCars()
}
onMaxCapacity = event => {
    this.maxCapacity = event.target.value
    this.preFetchCars()
}
preFetchCars = () => {
    this.resetParams()
    this.fetchCars()
}
```

```
    resetParams = () => {
        window.scrollTo(0, 0)
        this.page = 1
        this.limit = 0
        this.cars = []
    }
}
export default new cars()
```

index.js

```
import cars from "./modules/cars"
const store = {
    cars,
}
export default store
```

postcss.config.js

```
module.exports = {
    plugins: [
        require('autoprefixer')
    ],
}
```

package.json

```
{
    "name": "project",
    "version": "1.0.0",
    "description": "Project",
    "author": "",
    "license": "MIT",
```

```
"main": "index.js",
"scripts": {
    "dev": "webpack-dev-server --mode=development --watch --progress",
    "prod": "webpack --mode=production --watch --progress"
},
"browserslist": [
    ">0.2%",
    "not dead",
    "not ie <= 11",
    "not op_mini all",
    "last 2 versions"
],
"dependencies": {
    "@material-ui/core": "^4.0.0-beta.1",
    "@material-ui/icons": "^4.0.0-beta.0",
    "axios": "^0.18.0",
    "classnames": "^2.2.6",
    "history": "^4.9.0",
    "mobx": "^5.9.4",
    "mobx-react": "^5.4.3",
    "moment": "^2.24.0",
    "numbro": "^2.1.2",
    "react": "^16.8.6",
    "react-dom": "^16.8.6",
    "react-router-dom": "^5.0.0"
},
"devDependencies": {
    "@babel/core": "^7.4.4",
    "@babel/plugin-proposal-class-properties": "^7.4.4",
    "@babel/plugin-proposal-decorators": "^7.4.4",
```

```
        "@babel/plugin-syntax-dynamic-import": "^7.2.0",
        "@babel/plugin-transform-object-assign": "^7.2.0",
        "@babel/polyfill": "^7.4.4",
        "@babel/preset-env": "^7.4.4",
        "@babel/preset-react": "^7.0.0",
        "autoprefixer": "^9.5.1",
        "babel-loader": "^8.0.5",
        "babel-plugin-import": "^1.11.0",
        "css-loader": "^2.1.1",
        "file-loader": "^3.0.1",
        "html-webpack-harddisk-plugin": "^1.0.1",
        "html-webpack-plugin": "^3.2.0",
        "node-sass": "^4.12.0",
        "postcss-loader": "^3.0.0",
        "sass-loader": "^7.1.0",
        "style-loader": "^0.23.1",
        "terser-webpack-plugin": "^1.2.3",
        "webpack": "^4.31.0",
        "webpack-cli": "^3.3.2",
        "webpack-dev-server": "^3.3.1"
    }
}
```

.babelrc

```
{
    "presets": [
        [
            "@babel/preset-env",
            {
                "loose": true,
```

```
            "modules": false
          }
        ],
        "@babel/preset-react"
    ],
    "plugins": [
        ["@babel/plugin-proposal-decorators", { "legacy": true }],
        ["@babel/plugin-proposal-class-properties", { "loose": true }],
        "@babel/plugin-transform-object-assign"
    ]
}
```

ДОДАТОК В

Узагальнений технологічний ланцюжок інформаційної системи

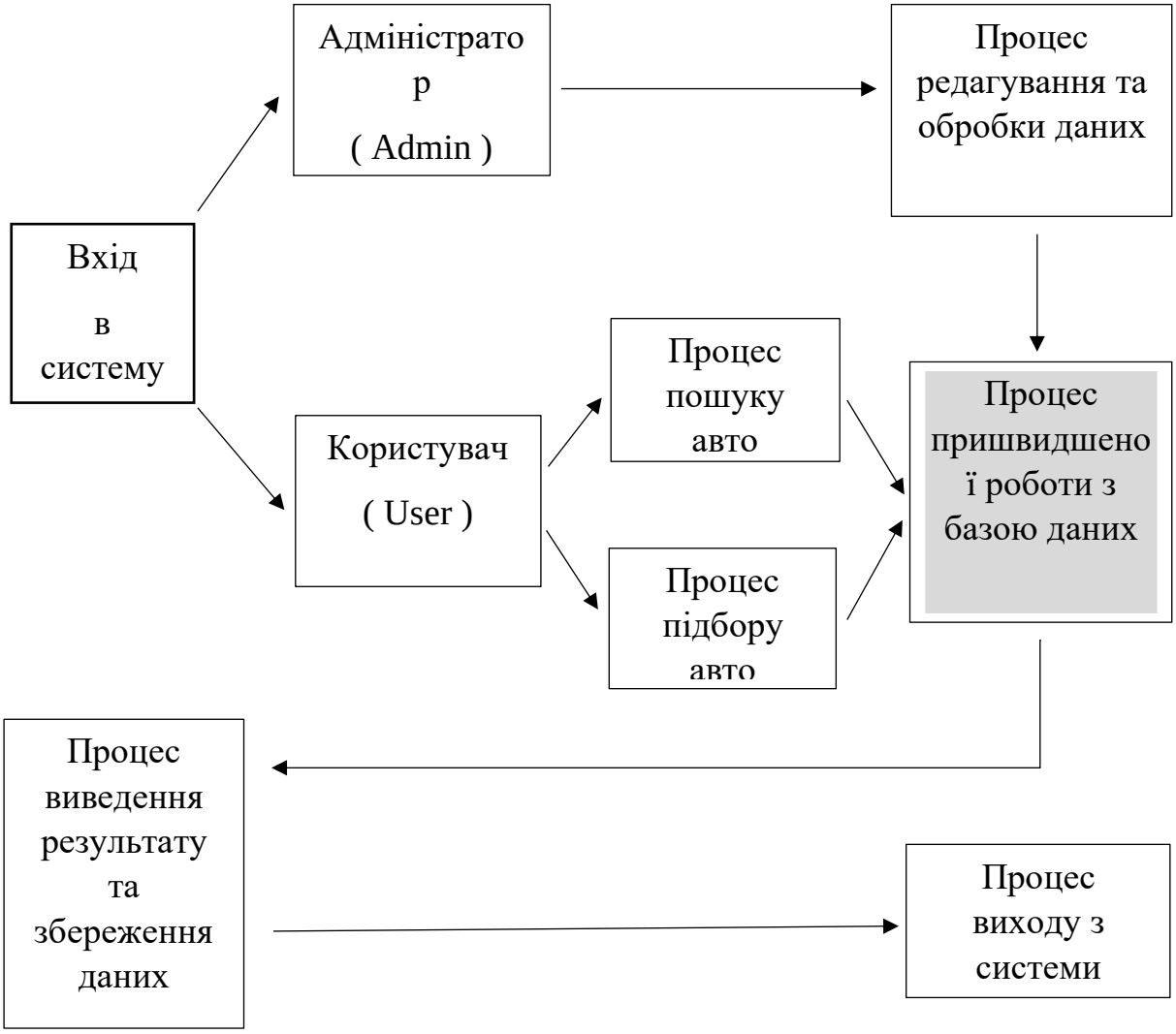

Рисунок В.1 — Узагальнений технологічний ланцюжок ІС

					08-23.МКР.007.00.002 ТЗ		
					Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розробив		Маренич Б.С.					
Керівник		Савицька Л. А.			Додаток В	Арк. 1	Аркушів 3	
Рецензент		Яремчук Ю. Є.					
Н. Контроль		Швець С. І.				ВНТУ, гр. КІ-19м	
Затверджую		Азаров О. Д.					

ДОДАТОК Г

Головна сторінка сайту

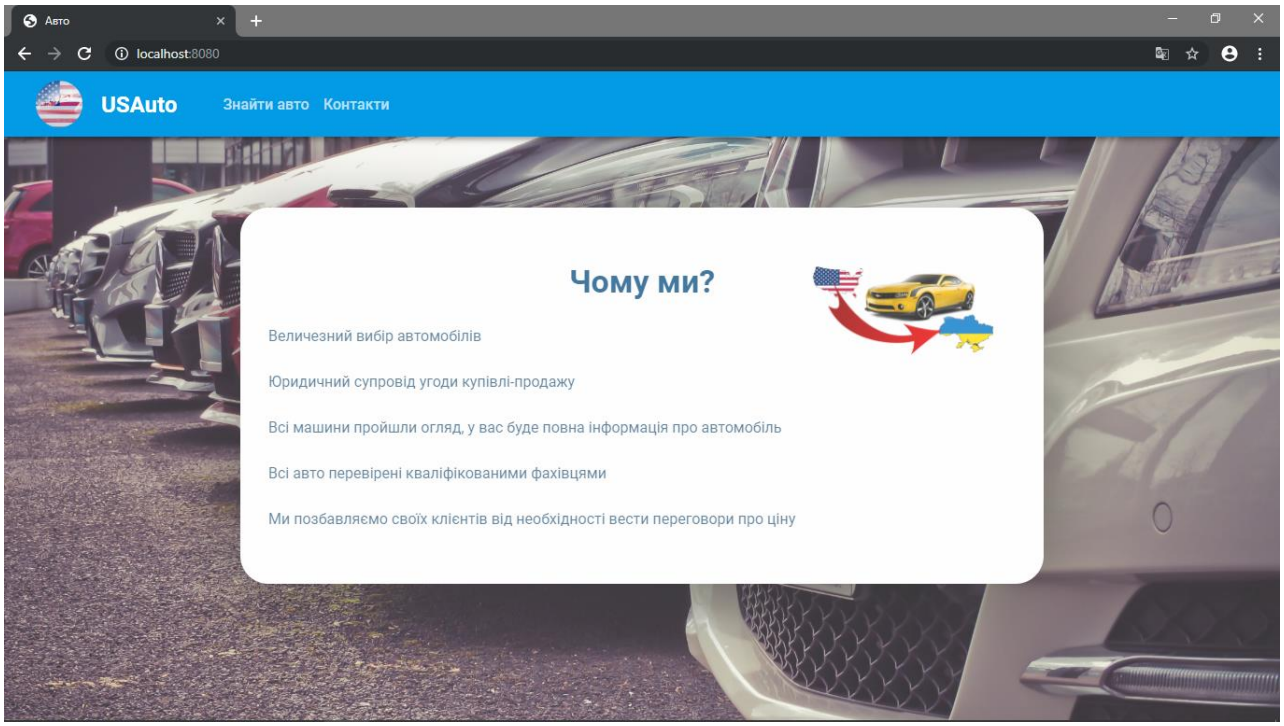

Рисунок Г.1 — Головна сторінка сайту

ДОДАТОК Д

Результат пошуку авто

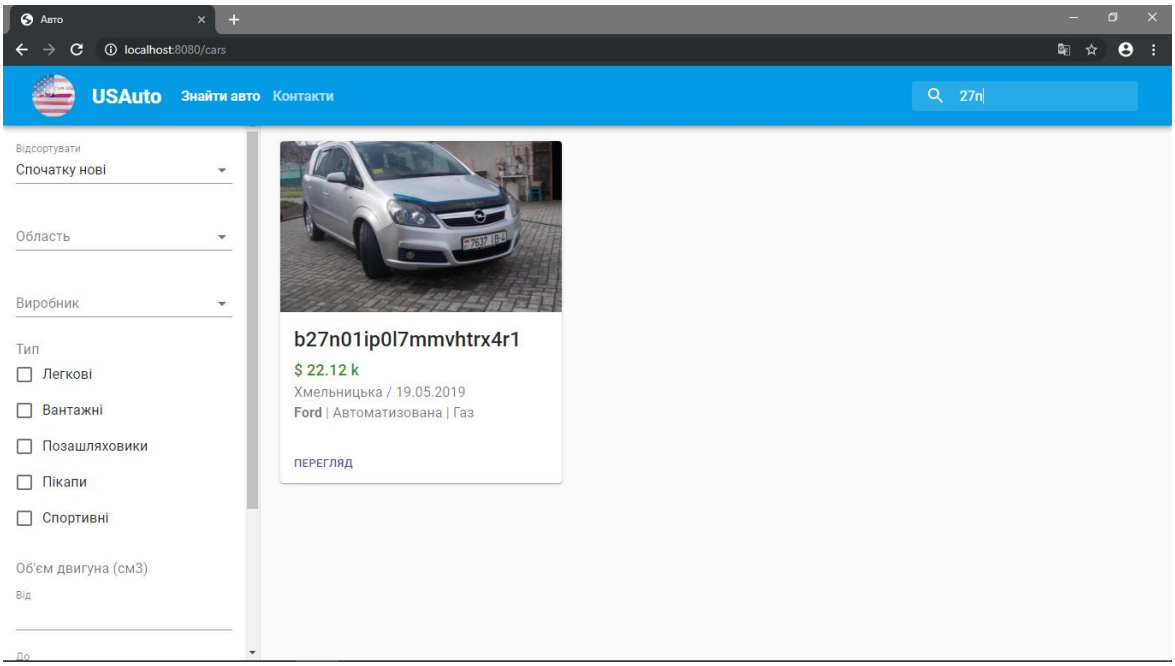

Рисунок Д.1 — Результат пошуку авто

					08-23.МКР.007.00.004 ТЗ		
					Web-орієнтований програмний засіб для організації та супроводження процесу підбору і замовлення автомобіля	Літ.	Маса	Масштаб
Змн.	Арк.	№ докум.	Підпис	Дата				
Розробив		Маренич Б.С.						
Керівник		Савицька Л. А.						
Рецензент		Яремчук Ю. Є.			Додаток Д	Арк. 3	Аркушів 3	
Н. Контроль		Швець С. І.				ВНТУ, гр. КІ-19м		
Затверджую		Азаров О. Д.						