

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

на тему:

**«Інформаційна система забезпечення підбору
автомобілів за заданими критеріями»**

Виконав студент групи ІСТ-20-1

Шумак Андрій Миколайович

Керівник роботи:

ЛЕВКІВСЬКИЙ Віталій Леонідович

Рецензент:

СУГОНЯК Інна Іванівна

Житомир – 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

ЗАТВЕРДЖУЮ

Завідувач кафедри комп'ютерних наук

Марина ГРАФ

«25» січня 2024 р.

ЗАВДАННЯ

на кваліфікаційну роботу освітнього ступеня «бакалавр»
за спеціальністю 126 «Інформаційні системи та технології»
(освітня програма «Системи бізнес-аналітики»)

Здобувач вищої освіти: **ШУМАК Андрій Миколайович**

Керівник роботи: **ЛЕВКІВСЬКИЙ Віталій Леонідович**

Тема роботи: **«Інформаційна система забезпечення підбору автомобілів за заданими критеріями»**,

затверджена наказом закладу вищої освіти від **«25» січня 2024 р., №24/с**

Термін здачі закінченої роботи: **«10» червня 2024 р.**

Вихідні дані роботи (зазначається предмет і об'єкт дослідження): об'єктом дослідження є процес розробки вебдодатків для підбору автомобілів. Предметом дослідження є використання вебтехнологій при розробці додатків для підбору автомобілів.

Консультанти випускної кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Дата	
		Завдання видав	Завдання прийняв

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі. Пошук, огляд та аналіз аналогічних розробок. Формулювання технічного завдання. Опрацювання літературних джерел.	2 лютого 2024 – 5 березня 2024	Виконано
2	Проектування структури системи забезпечення підбору автомобілів за заданими критеріями	6 березня 2024 – 25 березня 2024	Виконано
3	Написання програмного коду	26 березня 2024 – 26 квітня 2024	Виконано
4	Тестування системи	27 квітня 2024 – 14 травня 2024	Виконано
5	Оформлення пояснювальної записки	15 травня 2024 – 1 червня 2024	
6	Попередній захист	1 червня 2024	Виконано

Здобувач вищої освіти **Андрій ШУМАК**
Керівник **Віталій ЛЕВКІВСЬКИЙ**

РЕФЕРАТ

Кваліфікаційна робота включає в себе вебсистему забезпечення підбору автомобілів за заданими критеріями та супровідну пояснювальну записку. Остання містить у собі вступ, три розділи, висновки, список використаної літератури та додатки.

Текстова частина викладена на 64 сторінках друкованого тексту.

Пояснювальна записка має 57 сторінок додатків. Список використаних джерел містить 20 найменувань і займає 2 сторінки. В роботі наведено 31 рисунок та 9 таблиць. Загальний обсяг роботи – 121 сторінок.

Ключові слова: HTML, CSS, REACT, TYPESCRIPT, BOOTSTRAP, REDUX, REACT-HOOK-FORM, REACT-IMAGE-GALLERY

					ІСТ.КР.Б – 126 – 24 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.	А.М. ШУМАК				Інформаційна система забезпечення підбору автомобілів за заданими критеріями Пояснювальна записка	Літ.	Арк.	Аркушів
Керівник	В.Л. ЛЕВКІВСЬКИЙ						3	64
Рецензент	І. І. СУГОНЯК					Житомирська політехніка, група ІСТ-20-1		
Н. контр.								
Зав. каф.	М.С. ГРАФ							

ABSTRACT

The qualification work includes the process of developing a system for ensuring the selection of cars according to the specified criteria and an accompanying explanatory note. The latter includes an introduction, three chapters, conclusions, a list of references and an appendix.

The textual part is presented on 64 pages of printed text.

The explanatory note consists of 57 pages of appendices. The list of used sources includes 20 titles and occupies 2 pages. The work includes 31 figures and 9 tables. The total volume of the work is 121 pages.

Keywords: HTML, CSS, REACT, TYPESCRIPT, BOOTSTRAP, REDUX, REACT-HOOK-FORM, REACT-IMAGE-GALLERY

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		4

ЗМІСТ

РЕФЕРАТ	4
ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ.....	9
1.1 Постановка задачі	9
1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.....	11
1.3. Визначення архітектури сайту	18
1.4 Обґрунтування вибору інструментальних засобів	21
Висновки до першого розділу	24
РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНТЕГРАЦІЙНОЇ СИСТЕМИ .	25
2.1 Визначення функціональних вимог системи.	25
2.2. Проектування та реалізація алгоритмів роботи системи.....	30
2.3. Проектування бази даних.....	37
2.4. Розробка функціоналу системи	41
Висновки до другого розділу.....	48
РОЗДІЛ 3. ОПИС РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ.....	49
3.1. Встановлення та налаштування системи.....	49
3.2 . Структура інтерфейсу та опис роботи з системою	51
3.3. Тестування роботи системи	57
Висновки до третього розділу	61
ВИСНОВКИ.....	62
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	63
ДОДАТКИ.....	65

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ПЗ	Програмне забезпечення
HTML	HyperText Markup Language
CSS	Cascading Style Sheets
JS	JavaScript
UML	(Unified Modeling Language) – це уніфікована мова моделювання, яка використовується для візуалізації

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми. Розробка систем для забезпечення підбору автомобілів за заданими критеріями є надзвичайно необхідною з кількох причин.

1. Зростання онлайн-торгівлі та цифрових послуг. Все більше користувачів шукають інформацію про автомобілі онлайн. Наявність спеціалізованих сайтів для підбору автомобілів дозволяє спростити процес пошуку і вибору, роблячи його більш зручним та ефективним.

2. Персоналізація послуг. Сучасні споживачі очікують індивідуального підходу. Сайти, які можуть пропонувати автомобілі за заданими критеріями (наприклад, марка, модель, рік випуску, пробіг, ціна тощо), забезпечують саме такий підхід, що підвищує задоволеність клієнтів.

3. Конкуренція на ринку. Автомобільні дилери та приватні продавці постійно змагаються за увагу покупців. Сайти для підбору автомобілів допомагають виділитися серед конкурентів, пропонуючи зручні і швидкі інструменти для пошуку.

4. Ефективність і економія часу. Замість того, щоб переглядати численні оголошення і витратити час на пошук, користувачі можуть швидко знайти автомобілі, які відповідають їхнім критеріям, за допомогою спеціалізованих сайтів.

5. Інтеграція з іншими сервісами. Такі сайти можуть бути інтегровані з іншими корисними сервісами, як-от оцінка автомобілів, фінансові консультації, страхування, що робить їх ще більш привабливими для користувачів.

6. Аналіз ринку. Для дилерів і виробників автомобілів ці сайти можуть слугувати цінним джерелом інформації про попит, переваги і поведінку покупців, що дозволяє краще планувати маркетингові стратегії і виробництво.

Таким чином, розробка сайтів для підбору автомобілів за заданими критеріями не тільки відповідає потребам сучасних користувачів, але й надає значні переваги як для покупців, так і для продавців автомобілів.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

Метою кваліфікаційної роботи створення вебсистеми для забезпечення підбору автомобілів за заданими критеріями, що дозволить користувачам ефективно і зручно знаходити автомобілі, які відповідають їхнім потребам та уподобанням.

Об'єктом дослідження є процес розробки вебдодатків для підбору автомобілів.

Предметом дослідження є використання вебтехнологій при розробці додатків для підбору автомобілів.

.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧІ

1.1 Постановка задачі

Основним завданням є розробка функціональної вебсистеми для підбору автомобілів за заданими критеріями, яка забезпечить зручний, ефективний та швидкий пошук автомобілів, що відповідають потребам користувачів.

1. Аналіз ринку та існуючих рішень. Провести огляд та аналіз існуючих веб-сайтів та платформ, які надають послуги підбору автомобілів. Визначити основні функціональні можливості та недоліки цих рішень.

2. Визначення вимог до системи. Зібрати та визначити вимоги користувачів до системи. Визначити технічні вимоги до розробки веб-сайту, включаючи вимоги до продуктивності, безпеки та масштабованості.

3. Розробка архітектури системи. Розробити архітектуру веб-сайту, включаючи базу даних, серверну частину та клієнтський інтерфейс. Визначити технології та інструменти, які будуть використовуватись для розробки (наприклад, мови програмування, фреймворки, бази даних).

4. Розробка бази даних. Створити модель даних для зберігання інформації про автомобілі, включаючи їх характеристики та параметри. Реалізувати базу даних відповідно до розробленої моделі.

5. Розробка серверної частини. Реалізувати API для доступу до бази даних та забезпечення функціональності пошуку та фільтрації автомобілів за заданими критеріями. Забезпечити захист даних та безпеку взаємодії з сервером.

6. Розробка клієнтського інтерфейсу. Створити зручний та інтуїтивно зрозумілий інтерфейс користувача для пошуку автомобілів за заданими критеріями. Забезпечити адаптивність інтерфейсу для коректного відображення на різних пристроях (мобільні телефони, планшети, десктопи).

7. Тестування системи. Провести функціональне тестування для перевірки коректності роботи всіх модулів системи. Провести тестування продуктивності для забезпечення відповідності системи заданим вимогам.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

8. Впровадження та оцінка результатів. Впровадити систему на реальному сервері. Оцінити результати розробки на основі зворотного зв'язку від користувачів та ефективності роботи системи.

У рамках кваліфікаційної роботи передбачається створення інформаційної системи забезпечення підбору автомобілів за заданими критеріями. Для досягнення поставленої задачі потрібно виконати такі етапи розробки.

1. Підготовчий етап

a. Провести огляд та аналіз існуючих веб-сайтів та платформ для підбору автомобілів.

b. Визначити їхні основні функціональні можливості та недоліки.

c. Зібрати та визначити вимоги користувачів до системи (функціональні та нефункціональні вимоги).

d. Визначити технічні вимоги до розробки веб-сайту, включаючи вимоги до продуктивності, безпеки та масштабованості.

2. Проектування

a. Розробити архітектуру веб-сайту, що включає базу даних, серверну частину та клієнтський інтерфейс.

b. Вибрати технології та інструменти для розробки (наприклад, мови програмування, фреймворки, бази даних).

3. Розробка

a. Реалізувати API для доступу до бази даних та забезпечення функціональності пошуку та фільтрації автомобілів.

b. Забезпечити безпеку взаємодії з сервером (авторизація, аутентифікація, захист даних).

c. Створити зручний та інтуїтивно зрозумілий інтерфейс користувача для пошуку автомобілів за заданими критеріями.

d. Забезпечити адаптивність інтерфейсу для коректного відображення на різних пристроях (мобільні телефони, планшети, десктопи).

4. Тестування

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

- а. Провести тестування для перевірки коректності роботи всіх модулів системи.
 - б. Виявити та виправити помилки, які можуть вплинути на роботу системи.
 - с. Провести тестування продуктивності для перевірки відповідності системи заданим вимогам (наприклад, швидкість пошуку, обробка великої кількості запитів).
5. Впровадження
- а. Впровадити систему на реальному сервері, налаштувати серверне середовище.
 - б. Забезпечити безперебійну роботу системи та готовність до експлуатації.
6. Оцінка результатів та підтримка
- а. Оцінити результати розробки на основі зворотного зв'язку від користувачів та ефективності роботи системи.
 - б. Виявити можливі покращення та оновлення системи.
 - с. Забезпечити технічну підтримку користувачів.
 - д. Проводити регулярне оновлення системи для покращення функціональності та безпеки.

1.2. Аналіз існуючого програмного забезпечення за тематикою кваліфікаційної роботи.

Для аналізу було обрано декілька аналогічних сайтів зі схожим функціоналом.

«Cars.com» [1] є одним з найбільших і найвідоміших веб-сервісів для купівлі та продажу нових та вживаних автомобілів. Заснований у 1998 році, сайт пропонує широкий спектр інструментів та ресурсів для покупців автомобілів (рис. 1.1).

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

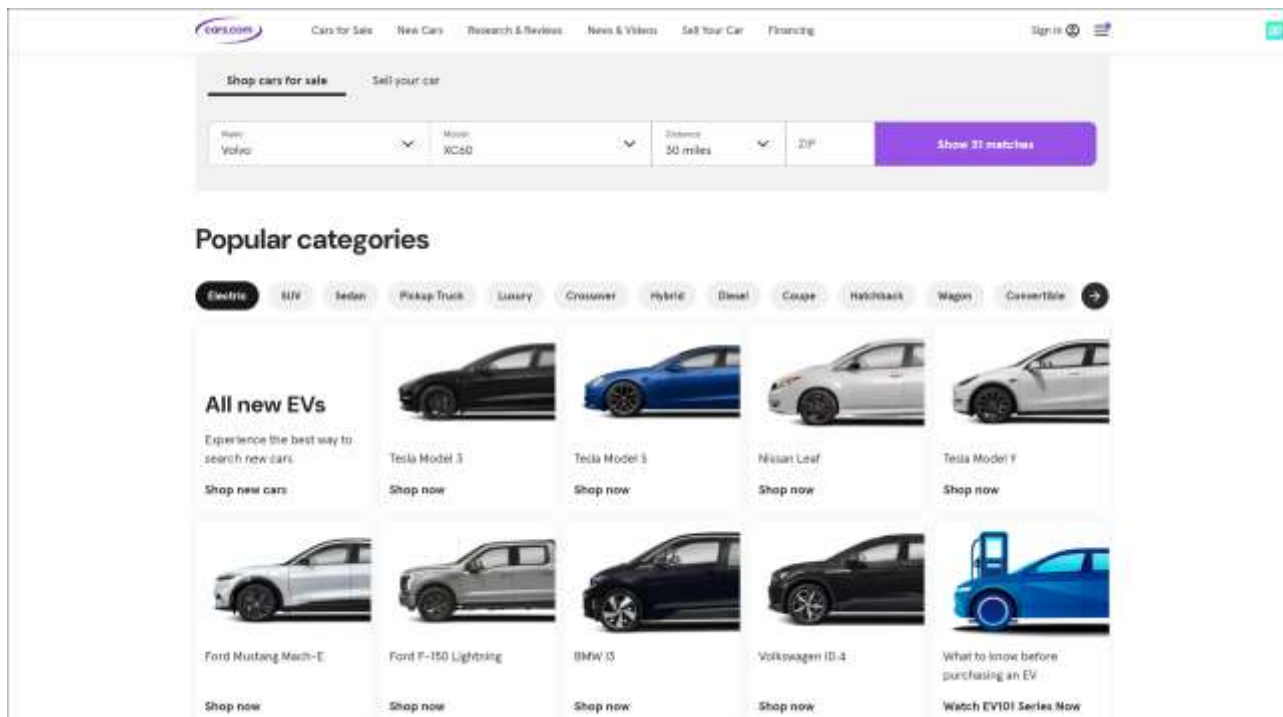


Рис 1.1 – Головна сторінка сайту Cars.com

Основні функції

1. Пошук автомобілів
 - а. Фільтри. Марка, модель, рік випуску, ціна, пробіг, тип кузова, колір, тип пального, трансмісія, привід, кількість дверей, розташування.
 - б. Зручність. Інтерфейс пошуку дозволяє легко налаштувати численні параметри для точного підбору автомобіля.
2. Огляди та рейтинги
 - а. Експертні огляди. Доступні детальні огляди від професійних автомобільних експертів.
 - б. Відгуки користувачів. Рейтинги та відгуки від реальних покупців, що допомагає зробити обґрунтоване рішення.
3. Порівняння автомобілів
 - а. Інструмент порівняння. Можливість порівняння декількох автомобілів за різними характеристиками, такими як ціна, пробіг, технічні дані тощо.
4. Фінансові інструменти
 - а. Калькулятори. Калькулятори вартості кредиту, лізингу, страхування.

b. Інформація. Дані про програми фінансування, доступні ставки кредитів, програми лізингу.

5. Додаткові сервіси

a. Поради. Рекомендації для покупців і продавців, статті про догляд за автомобілем.

b. Оцінка вартості. Інструменти для оцінки вартості вживаних автомобілів.

Переваги

a. Велика база даних автомобілів.

b. Зручний та інтуїтивно зрозумілий інтерфейс.

c. Детальні огляди та рейтинги допомагають користувачам у прийнятті рішень.

Недоліки

a. Іноді користувачі можуть знайти інтерфейс занадто насиченим, особливо новачки, через велику кількість інформації та опцій.

b. Велика кількість реклами може відволікати користувачів і створювати незручності під час пошуку автомобілів.

c. Деякі користувачі скаржаться на неточності у роботі фільтрів, що може призвести до невідповідності результатів пошуку заданим критеріям.

d. Іноді оголошення можуть бути застарілими, що означає, що інформація про наявність автомобіля може бути неактуальною.

«Autotrader.com» [2], заснований у 1997 році, є одним з найбільш відомих і широко використовуваних сайтів для купівлі та продажу нових і вживаних автомобілів. Сайт пропонує численні інструменти для пошуку автомобілів та містить багато корисної інформації для автолюбителів (рис. 1.2).

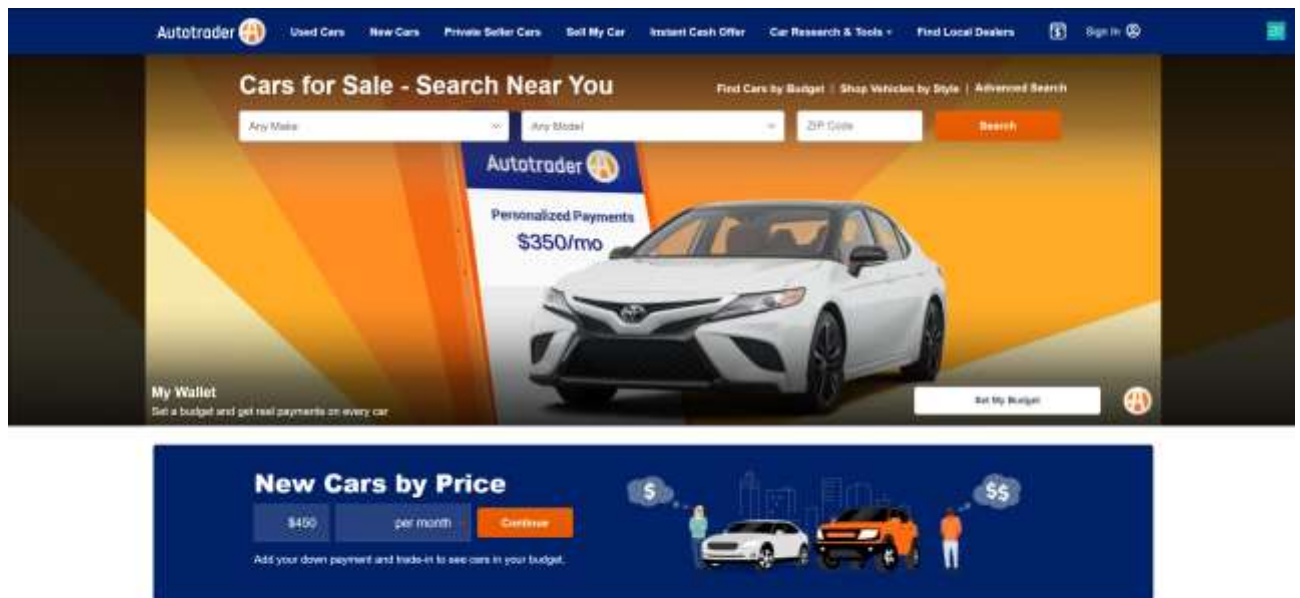


Рис 1.2 – Головна сторінка сайту Autotrader.com

Основні функції

1. Пошук автомобілів
 - а. Фільтри. Марка, модель, рік випуску, ціна, пробіг, тип кузова, колір, тип пального, трансмісія, привід, кількість дверей, розташування.
 - б. Інтерактивний пошук. Карта з розташуванням автомобілів, функція пошуку за радіусом від заданого місця.
2. Огляди та оцінки
 - а. Експертні огляди. Огляди від професійних автомобільних журналістів.
 - б. Відгуки користувачів. Рейтинги та відгуки від реальних користувачів про автомобілі та дилерів.
3. Порівняння автомобілів
 - а. Інструмент порівняння. Можливість порівняння декількох автомобілів за характеристиками та цінами.
4. Фінансові інструменти
 - а. Калькулятори. Калькулятори вартості кредиту та лізингу.
 - б. Програми фінансування. Інформація про доступні програми фінансування та страхування.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						14
Змн.	Арк.	№ докум.	Підпис	Дата		

5. Додаткові функції

a. Оцінка вартості. Інструмент для оцінки вартості вживаних автомобілів.

b. Поради. Рекомендації щодо купівлі та продажу автомобілів, догляду за ними.

Переваги

a. Широкий вибір автомобілів з різних категорій.

b. Детальні огляди та оцінки від експертів і користувачів.

c. Інтерактивний інтерфейс з багатьма фільтрами для точного пошуку.

Недоліки

a. Деякі корисні функції, такі як розширені фільтри або доступ до певної інформації, можуть бути доступні тільки за плату.

b. Іноді сайт може працювати повільно, особливо під час пікових навантажень, що може викликати незручності для користувачів.

c. Велика кількість реклами може бути дратівливою для користувачів і ускладнювати процес пошуку.

d. Іноді інформація про автомобілі може бути неактуальною або неправильною, що може вводити в оману покупців.

«CarGurus» [3], заснований у 2006 році, є популярним веб-сайтом для пошуку нових і вживаних автомобілів. Сайт відомий своїм унікальним алгоритмом оцінки справедливості цін на автомобілі, що допомагає користувачам знаходити найкращі пропозиції (рис. 1.3).

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дата		

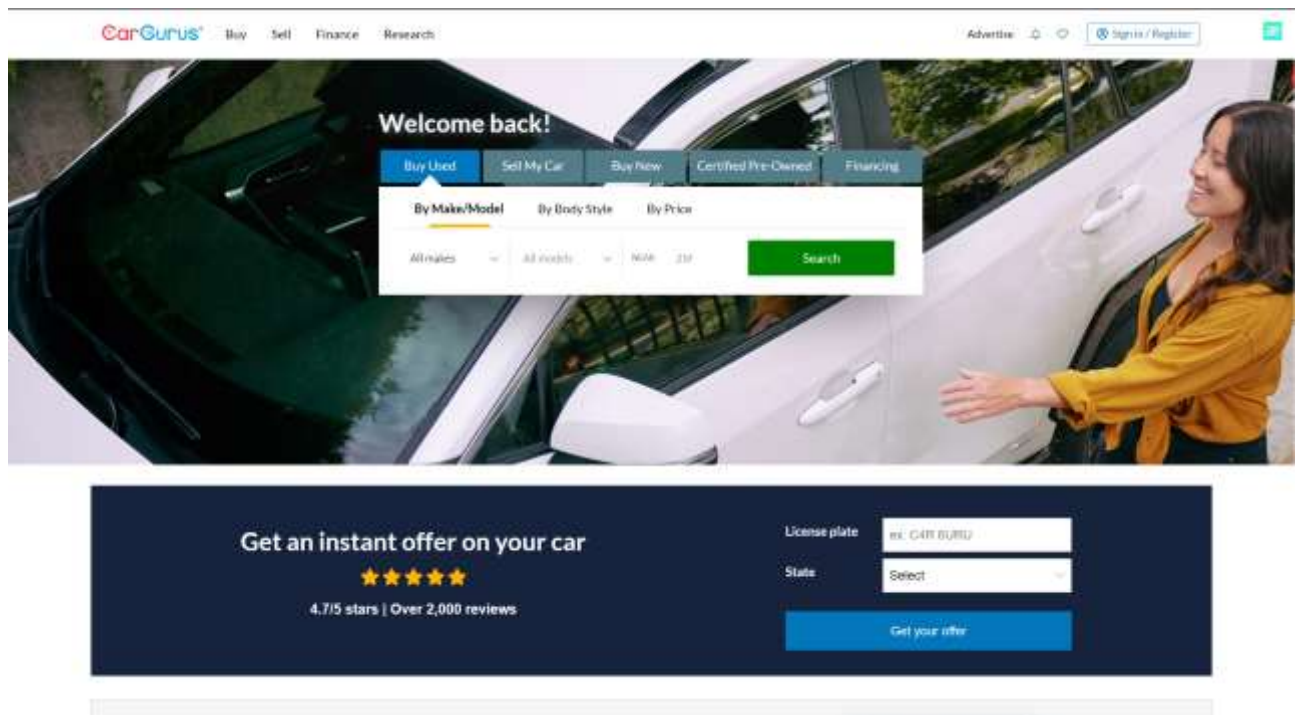


Рис 1.3 – Головна сторінка сайту CarGurus

Основні функції

1. Пошук автомобілів
 - а. Фільтри. Марка, модель, рік випуску, ціна, пробіг, тип кузова, тип пального, трансмісія, привід, колір, розташування.
 - б. Зручність. Інтерфейс з розширеними можливостями фільтрації.
2. Оцінка справедливості ціни
 - а. Алгоритм. Власний алгоритм оцінки вартості автомобілів, який показує, чи є пропозиція вигідною.
 - б. Рейтинг. Автомобілі оцінюються за критеріями "Відмінна угода", "Добра угода", "Звичайна угода" тощо.
3. Огляди та рейтинги
 - а. Експертні огляди. Професійні огляди автомобілів.
 - б. Відгуки користувачів. Рейтинги та відгуки від реальних покупців, оцінка дилерів.
4. Порівняння автомобілів
 - а. Інструмент порівняння. Можливість порівняння декількох автомобілів за різними характеристиками та цінами.
5. Додаткові функції

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дата		

a. Фінансування. Інструменти для пошуку вигідних програм фінансування та страхування.

b. Калькулятори. Калькулятори вартості кредиту та лізингу.

Переваги

a. Унікальний алгоритм оцінки справедливості ціни.

b. Зручний інтерфейс для пошуку автомобілів.

c. Детальні огляди та рейтинги від експертів і користувачів.

Недоліки

a. У деяких регіонах сайт може не пропонувати достатньо оголошень або інформації про дилерів, що обмежує можливості пошуку.

b. Хоча алгоритм оцінки цін є корисним, він іноді може бути неточним, що може вплинути на рішення покупців.

c. Доступ до деяких функцій, таких як історія цін або розширені фільтри, може вимагати платної підписки.

d. Деякі користувачі можуть знайти інтерфейс менш інтуїтивним порівняно з іншими сайтами, що може викликати труднощі під час навігації.

Загальні недоліки для всіх трьох сайтів

a. Незважаючи на їхні зусилля, ці сайти можуть містити оголошення з неактуальною інформацією про наявність або ціни автомобілів.

b. Велика кількість реклами та банерів може негативно впливати на користувацький досвід.

c. Хоча ці сайти намагаються забезпечити безпеку користувацьких даних, завжди існує ризик витоку або зловживання даними.

d. Іноді інформація, надана дилерами, може бути не зовсім точною або завищеною, що створює додаткові складнощі для покупців.

Попри недоліки, ці сайти залишаються популярними і корисними інструментами для пошуку автомобілів. Вони пропонують широкий спектр

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						17
Змн.	Арк.	№ докум.	Підпис	Дата		

функцій, які можуть значно полегшити процес вибору та купівлі автомобіля. Водночас, користувачі повинні бути обізнаними про можливі недоліки і використовувати кілька джерел інформації для прийняття зважених рішень.

Ці три веб-сайти мають багатий набір функцій та інструментів, які роблять процес підбору автомобілів зручним та ефективним для користувачів. Вони надають різноманітні можливості для пошуку, порівняння та оцінки автомобілів, що може стати цінним джерелом натхнення для розробки власного веб-сайту з підбору автомобілів.

1.3. Визначення архітектури сайту

Архітектура системи

1. Клієнтська частина
 - a. Інтерактивний веб-інтерфейс, створений з використанням сучасних технологій фронтенду, таких як HTML[4], CSS[5] а також фреймворків React [6], TypeScript [7], BootStrap [8], Redux[9], React-Hook-Form [16], React-Image-Gallery [17].
 - b. Адаптивний дизайн, який забезпечує коректне відображення на всіх типах пристроїв, включаючи десктопи, планшети та мобільні телефони.
2. Серверна частина
 - a. API для обробки запитів від клієнтської частини, взаємодії з базою даних та іншими сервісами.
 - b. Використання серверних мов програмування, таких як Node.js [10] а також відповідних фреймворків (Express [11], jsonwebtoken [12], Nodemon [13], Mongoose [14], Dotenv [18], npm [19]).
 - c. Реалізація функцій автентифікації та авторизації користувачів, забезпечення безпеки даних.
3. База даних
 - a. Реляційна база даних для зберігання інформації про автомобілі, користувачів, оголошення та історію запитів.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

b. Використання MongoDB [15] для забезпечення високої продуктивності та масштабованості системи.

Дизайн інтерфейсу

1. Користувацький інтерфейс

a. Інтуїтивно зрозумілий та зручний інтерфейс з простими елементами навігації.

b. Використання сучасних дизайнів та принципів UX/UI для забезпечення максимальної зручності користувачів.

2. Адаптивний дизайн

a. Забезпечення коректного відображення на всіх типах пристроїв, включаючи десктопи, планшети та мобільні телефони.

b. Використання адаптивних CSS-фреймворків, таких як Bootstrap або Material-UI.

3. Кольорова схема та стилі

a. Використання нейтральних кольорів та простих шрифтів для полегшення сприйняття інформації.

b. Сучасний та привабливий дизайн з акцентом на зручність використання.

План реалізації

1. Аналіз та планування

a. Збір вимог від потенційних користувачів та зацікавлених сторін.

b. Аналіз ринку та конкурентів для визначення ключових функцій та можливостей.

c. Визначення технічних вимог та створення плану проекту.

2. Проектування

a. Розробка архітектури системи та визначення основних компонентів.

b. Проектування бази даних та визначення структури даних.

c. Створення прототипів інтерфейсу та отримання зворотного зв'язку від користувачів.

3. Розробка

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

a. Розробка серверної частини додатку з використанням обраних технологій.

b. Розробка клієнтського інтерфейсу з використанням сучасних фреймворків та бібліотек.

c. Інтеграція з зовнішніми сервісами для перевірки історії автомобілів та обробки платежів.

4. Тестування

a. Функціональне тестування для перевірки коректної роботи всіх функцій.

b. Тестування продуктивності для забезпечення швидкодії системи.

c. Користувацьке тестування для отримання зворотного зв'язку та внесення необхідних змін.

5. Впровадження

a. Впровадження системи на реальному сервері та налаштування середовища.

b. Проведення навчання користувачів та надання технічної підтримки.

6. Оцінка та підтримка

a. Оцінка результатів розробки та отримання зворотного зв'язку від користувачів.

b. Регулярне оновлення системи для виправлення помилок та додавання нових функцій.

c. Забезпечення технічної підтримки користувачів для вирішення проблем та питань.

Ризики та управління

1. Ризики

a. Технічні складнощі, пов'язані з інтеграцією різних компонентів системи.

b. Зміни в вимогах користувачів, що можуть вплинути на функціональність системи.

2. Управління ризиками

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

- а. Регулярний моніторинг прогресу та виявлення потенційних проблем на ранніх етапах.
- б. Залучення зовнішніх експертів при необхідності для вирішення технічних питань.
- с. Гнучке планування та управління ресурсами для адаптації до змін у вимогах та умовах ринку.

Створення вебсистеми для підбору автомобілів за заданими критеріями є складним, але здійсненним завданням, що потребує ретельного планування, належного ресурсного забезпечення та чіткого управління. Виконання вищенаведених етапів допоможе створити якісний продукт, який задовольнить потреби користувачів і займе гідне місце на ринку. Важливо забезпечити постійний зворотний зв'язок від користувачів та регулярно оновлювати систему для підтримки її актуальності та функціональності.

1.4 Обґрунтування вибору інструментальних засобів

Популярні на даний час інструменти розробки фронтенду для веб сайтів це HTML, CSS, React, TypeScript, BootStrap, Redux. Для бекенд частіше всього використовують Node.js, JavaScript, Express, jsonwebtoken, Nodemon, Mongoose. Для бази даних було використано MongoDB. Було проведено інтеграцію з сервісами AutoRIA та Copart.

Для фронтенду було використано наступні інструменти.

1. HTML (HyperText Markup Language)

- а. Базова мова розмітки для створення веб-сторінок. Вона є фундаментальною технологією для структурування контенту на веб-сторінках.

2. CSS (Cascading Style Sheets)

- а. Використовується для стилізації та візуального оформлення веб-сторінок. З CSS можна створити адаптивний дизайн, що забезпечить коректне відображення сайту на різних пристроях.

3. React

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

- a. Популярний JavaScript-бібліотека для створення користувацьких інтерфейсів. React дозволяє розробляти динамічні та інтерактивні інтерфейси з високою продуктивністю. Компонентний підхід React спрощує підтримку та масштабування коду.

4. TypeScript

- a. Надбудова над JavaScript, що додає статичну типізацію. Це покращує якість коду, зменшує кількість помилок та полегшує масштабування великих проектів.

5. Bootstrap

- a. CSS-фреймворк для швидкої розробки адаптивних і мобільно-орієнтованих веб-сайтів. Включає в себе набір готових стилів та компонентів, що прискорює процес розробки.

6. Redux

- a. Бібліотека для керування станом додатку. Вона допомагає зберігати і керувати станом додатку в передбачуваний спосіб, що є важливим для великих та складних додатків.

Для реалізації бекенду було використано наступні інструменти.

1. Node.js

- a. JavaScript-оточення на стороні сервера, яке забезпечує високу продуктивність та масштабованість. Node.js відомий своєю асинхронною архітектурою, яка дозволяє обробляти велику кількість одночасних запитів.

2. JavaScript

- a. Основна мова програмування для Node.js. Використання однієї мови для фронтенду та бекенду спрощує розробку та підтримку додатку.

3. Express

- a. Легкий та гнучкий фреймворк для Node.js, який спрощує створення веб-додатків та API. Express забезпечує широкий набір функціональності для маршрутизації та обробки HTTP-запитів.

4. Jsonwebtoken

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						22
Змн.	Арк.	№ докум.	Підпис	Дата		

- а. Бібліотека для створення та перевірки JSON Web Tokens (JWT). JWT використовується для аутентифікації та авторизації користувачів, забезпечуючи безпечний доступ до ресурсів.

5. Nodemon

- а. Інструмент для автоматичного перезапуску Node.js-додатків при зміні коду. Це значно підвищує ефективність розробки та зручність для розробників.

6. Mongoose

- а. Бібліотека для роботи з MongoDB в Node.js. Вона надає інтуїтивний API для створення схем, валідації та взаємодії з базою даних, що спрощує процес роботи з MongoDB.

Для реалізації бази даних було використано наступні інструменти.

1. MongoDB

- а. Документно-орієнтована база даних NoSQL, яка забезпечує гнучкість у зберіганні та обробці даних. MongoDB добре підходить для додатків, що потребують масштабованості та високої продуктивності. Її структура дозволяє зберігати складні дані в документі JSON-подібного формату, що полегшує роботу з даними у Node.js.

Також було проведено інтеграцію з іншими сервісами для пошуку автомобілів.

1. AutoRIA

- а. Популярний український сервіс для купівлі та продажу автомобілів. Інтеграція з AutoRIA дозволить отримувати актуальні дані про автомобілі, доступні для покупки, і надавати користувачам додаткову інформацію для прийняття рішень.

2. Copart

- а. Сервіс для купівлі автомобілів на аукціонах, включаючи пошкоджені транспортні засоби. Інтеграція з Copart розширить можливості пошуку, надаючи користувачам доступ до аукціонних пропозицій і спеціальних угод.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

Вибір цих технологій забезпечує сучасний, гнучкий та масштабований підхід до розробки веб-сайту для підбору автомобілів. Комбінація потужних інструментів фронтенду та бекенду, а також інтеграція з провідними сервісами, дозволить створити конкурентоспроможний продукт з високою продуктивністю та зручністю для користувачів.

Висновки до першого розділу

Проаналізовано вимоги до програмного продукту та виходячи з них було створено технічне завдання. Проведено аналіз функціоналу трьох провідних веб-сайтів для підбору автомобілів – Cars.com, Autotrader.com та CarGurus.com. Обрано технології : для Фронтенду HTML, CSS, JavaScript, фреймворк (React, TypeScript, BootStrap, Redux) для створення динамічних та інтерактивних інтерфейсів., для Бекенду Node.js, фреймворк (Express, jsonwebtoken, Nodemon, Mongoose) для забезпечення обробки запитів та логіки додатку., та для бази даних MongoDB для зберігання та управління даними.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНТЕГРАЦІЙНОЇ СИСТЕМИ

2.1 Визначення функціональних вимог системи.

Діаграма варіантів використання (Use Case Diagram) для сайту автомобілів показує взаємодію користувачів та адміністраторів з основними функціями сайту (рис. 2.1).

Взаємодія Користувача

Користувач має доступ до наступних функцій

1. Вхід/Реєстрація
 - а. Користувач може увійти до свого облікового запису або створити новий.
2. Пошук автомобілів
 - а. Користувач може шукати автомобілі за різними критеріями.
3. Перегляд результатів пошуку
 - а. Користувач може переглядати результати пошуку та детальну інформацію про знайдені автомобілі.
4. Вподобання авто
 - а. Користувач може відзначати автомобілі, які йому сподобались.

Взаємодія Адміністратора

Адміністратор має доступ до додаткових функцій для керування оголошеннями

1. Управління оголошеннями
 - а. Адміністратор може переглядати та керувати всіма оголошеннями на сайті.
2. Додавання оголошення
 - а. Адміністратор може додавати нові оголошення про продаж автомобілів.
3. Редагування оголошення
 - а. Адміністратор може редагувати вже існуючі оголошення.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Видалення оголошення

а. Адміністратор може видаляти оголошення, які більше не актуальні або порушують правила сайту.

Ця діаграма надає загальний огляд основних можливостей сайту та розподілу обов'язків між користувачами та адміністраторами.

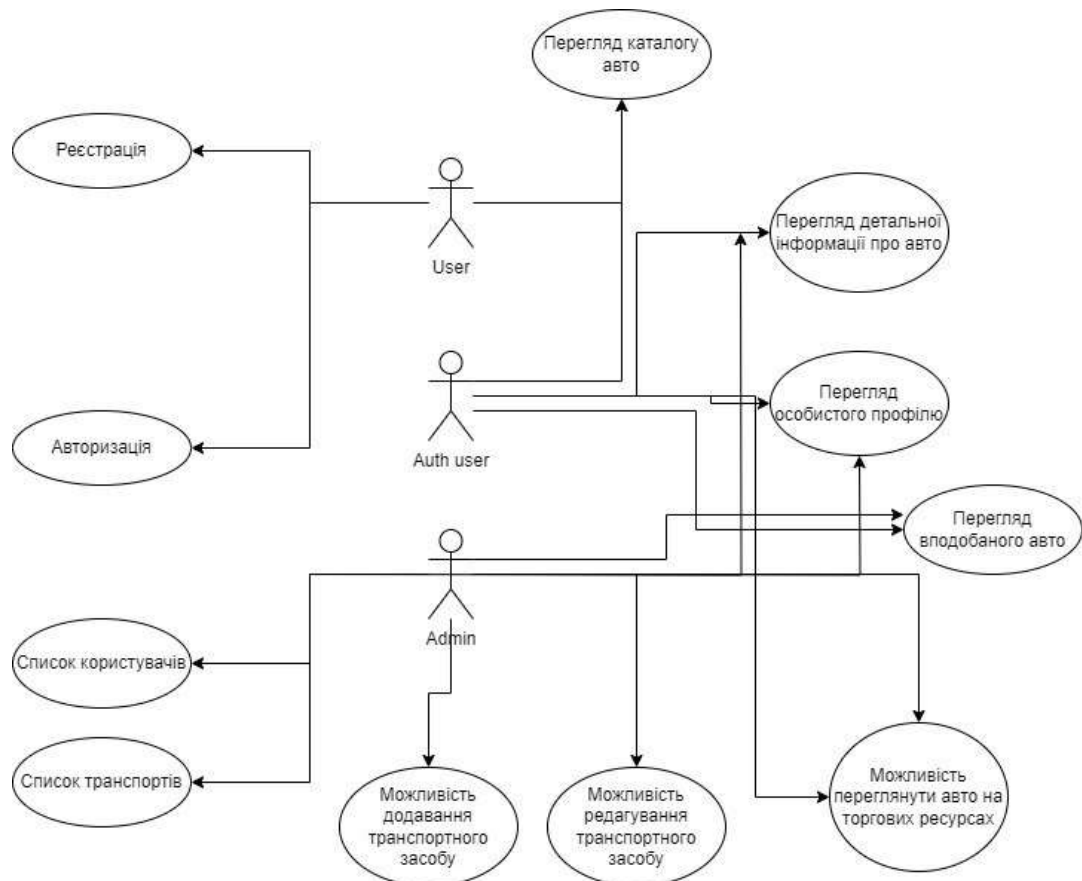


Рисунок 2.1 – Діаграма варіантів використання сайту

Опис діаграми варіантів використання наведено в таблицях 2.1 – 2.5

Таблиця 2.1

Опис прецеденту «Вхід/Реєстрація»

Назва варіанту використання	Вхід/реєстрація
Основні актори	Адміністратор, авторизований та не авторизований користувач
Короткий опис	Користувачі можуть дивитися оголошення або зареєструватися щоб продати авто, авторизація для адміністраторів, щоб керувати оголошеннями.

Таблиця 2.2

Опис прецеденту «Пошук та перегляд оголошень»

Назва варіанту використання	Пошук та перегляд оголошень
Основні актори	Адміністратор, авторизований та не авторизований користувач
Короткий опис	Користувачі можуть шукати та переглядати різні оголошення, які були створені на сайті.

Таблиця 2.3

Опис прецеденту «Особистий кабінет»

Назва варіанту використання	Особистий кабінет
Основні актори	Адміністратор, авторизований користувач
Короткий опис	Користувачі можуть додавати до списку бажаного авто, адміністратори мають можливість міняти ролі користувачів.

Таблиця 2.4

Опис прецеденту «Сторінка редагування»

Назва варіанту використання	Сторінка редагування
Основні актори	Адміністратор
Короткий опис	Адміністратор вибирає автомобіль зі списку та вносить необхідні зміни у форму редагування.

Таблиця 2.5

Опис прецеденту «Сторінка додавання авто»

Назва варіанту використання	Сторінка додавання авто
Основні актори	Адміністратор
Короткий опис	Дозволяє адміністраторам додавати нові автомобілі до бази даних.

Основна ціль розробки сайту для підбору автомобілів за заданими критеріями полягає в створенні ефективного та зручного інструменту, який

допоможе користувачам швидко і легко знайти автомобіль, що відповідає їхнім конкретним вимогам. Цей сайт повинен забезпечувати інтуїтивно зрозумілий інтерфейс, багатофункціональні можливості пошуку та фільтрації, а також високу продуктивність та надійність.

Вимоги користувачів:

Внутрішній користувач – адміністратор:

1. Редагує оголошення авто
2. Міняє ролі користувачів

Внутрішній користувач – не авторизований користувач:

1. Може дивитися оголошення
2. Може шукати авто

Внутрішній користувач – авторизований користувач:

1. Може дивитися оголошення
2. Може шукати авто
3. Додавати до списку бажаного

Продуктивність та швидкодія

а. Сайт повинен завантажувати сторінки швидко, не більше ніж за 3 секунди, щоб забезпечити приємний користувацький досвід і знизити відсоток відмов.

б. Система пошуку має бути ефективною та здатною обробляти запити користувачів навіть під час пікових навантажень.

Надійність та стабільність

а. Сайт повинен бути стабільним і надійним, без системних збоїв і відмов.

б. Передбачити план відновлення системи (disaster recovery plan) у випадку непередбачених ситуацій.

Безпека

а. Забезпечити захист персональних даних користувачів і конфіденційності інформації.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

б. Мати заходи захисту від кібератак, включаючи SQL ін'єкції, XSS атаки і т.д.

с. Використовувати HTTPS для забезпечення безпеки передачі даних між сервером і клієнтом.

Сумісність та доступність

а. Сайт повинен бути доступний з будь-якого типу пристрою (планшет, мобільний телефон, настільний комп'ютер) і будь-якого браузера.

б. Забезпечити сумісність із сучасними версіями популярних браузерів (Chrome, Firefox, Safari, Edge).

Інтерфейс та взаємодія

а. Інтерфейс користувача повинен бути інтуїтивно зрозумілим, легким у навігації і зручним для використання.

б. Забезпечити можливість швидкого доступу до основних функцій, таких як пошук, фільтрація і перегляд автомобілів.

Оновлення та підтримка

а. Забезпечити регулярні оновлення функціоналу та забезпечення високої актуальності інформації на сайті.

б. Надати користувачам можливість зворотного зв'язку і підтримки для вирішення їхніх проблем і питань.

Масштабованість

а. Система повинна бути готовою до масштабування, щоб впоратися зі зростанням кількості користувачів і об'ємом даних.

б. Забезпечити горизонтальне масштабування із збереженням продуктивності.

Ці нефункціональні вимоги допоможуть забезпечити, що розроблений сайт для підбору автомобілів буде не лише функціонально відповідати потребам користувачів, а й мати високу якість, надійність і ефективність у виконанні своїх завдань.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

2.2. Проектування та реалізація алгоритмів роботи системи

Додаток є простим у використанні, має всі основні функції платформи, що дозволяє користувачам легко взаємодіяти з платформою. Алгоритм роботи з платформою зображено на рисунку 2.2, також описано детально його пункти.

Якщо авторизувався адміністратор

- а. Адміністративна панель: Після авторизації адміністратор потрапляє в адміністративну панель.
- б. Перегляд і підтвердження заявок: В панелі адміністратор може переглянути всі заявки і підтвердити їх, якщо вони задовольняють умови користування платформою.
- с. Управління категоріями: Адміністратор має можливість створювати нові категорії, редагувати категорії, та видаляти їх за потреби.
- д. Додавання адміністраторів: Адміністратор має можливість додавати нових адміністраторів шляхом зміни ролі.

Якщо зайшов клієнт

- а. Перегляд активних автомобілів: Клієнт має можливість переглянути список активних автомобілів, доступних для перегляду.
- б. Перегляд статистики: Клієнт має можливість переглянути статистику по окремих автомобілях.
- с. Створення заявки: Клієнт має можливість створити нову заявку на придбання автомобіля.

Якщо авторизувався користувач

- а. Перегляд активних автомобілів: Користувач має можливість переглянути активні автомобілі, доступні для перегляду.
- б. Шукати авто: Користувач може шукати автомобілі.
- с. Публікувати авто: Користувач може публікувати оголошення про продаж автомобіля.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

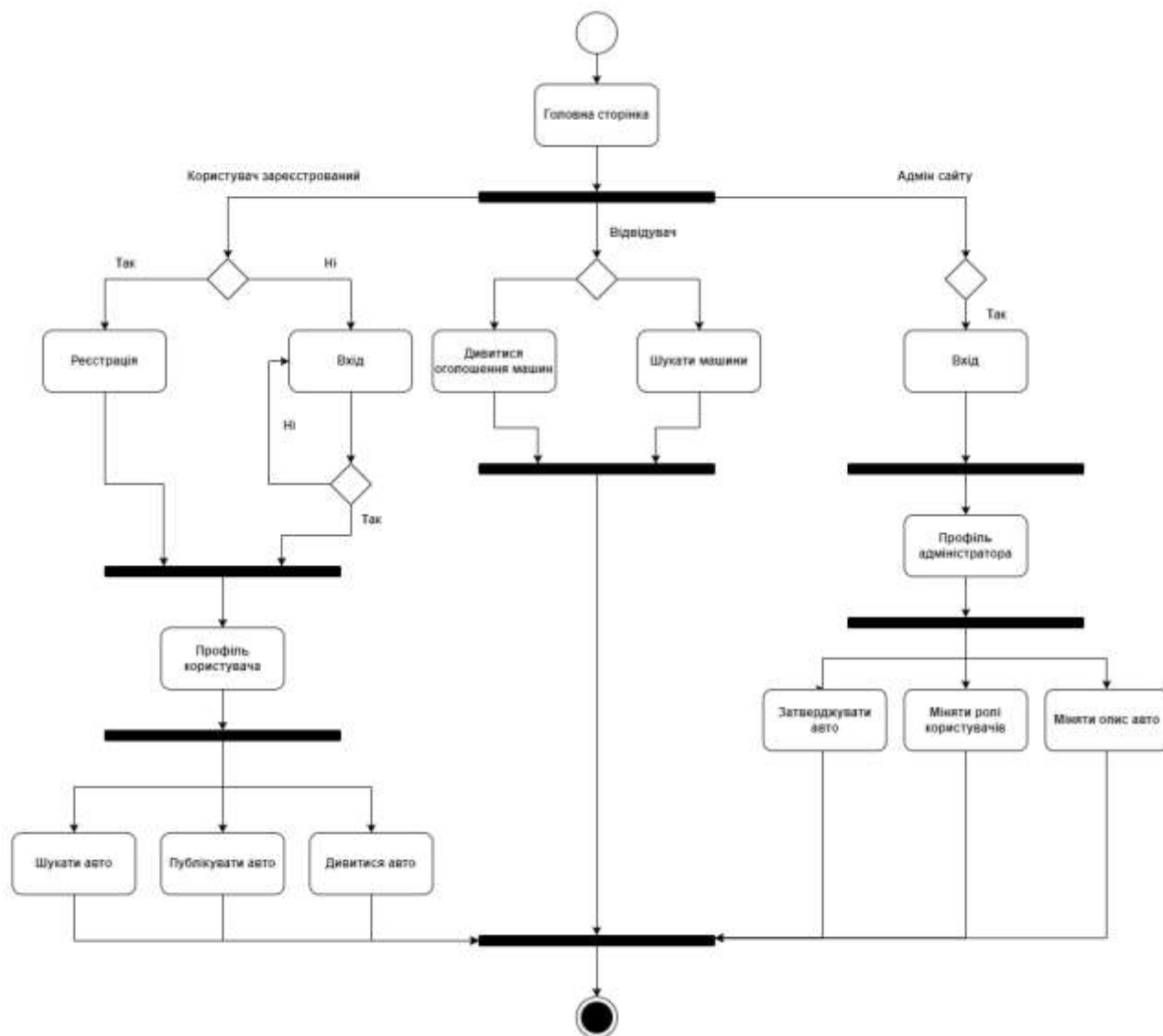


Рис. 2.2. Діаграма активності користувачів на сайті

Діаграма надає чітке уявлення про різні шляхи взаємодії користувачів та адміністратора з системою і про основні функціональні можливості сайту автомобілів.

Діаграма компонентів зображує маршрутизацію та контроль авторизації користувачів у веб-застосунку (рис. 2.3). Вона показує, як різні запити на авторизацію обробляються контролерами, що забезпечують виконання відповідних операцій.

Опис компонентів та їх взаємодії

1. Роутинг для виконання операцій над авторизацією
 - а. Цей компонент відповідає за маршрутизацію HTTP-запитів до відповідних кінцевих точок (endpoint).

Змн.	Арк.	№ докум.	Підпис	Дата

2. Кінцеві точки (endpoints)

- a. `api/me`: Використовується для отримання інформації про залогіненого користувача.
- b. `api/login`: Використовується для авторизації користувача.
- c. `api/signup`: Використовується для реєстрації нового користувача.
- d. `api/logout`: Використовується для виходу з облікового запису.
- e. `api/refresh`: Використовується для оновлення токена авторизації.

3. Контролер

- a. Цей компонент обробляє запити, що надходять через кінцеві точки, і виконує відповідні дії.

4. Операції контролера

- a. Отримання інформації про залогіненого користувача: Контролер обробляє запити до `api/me` для надання інформації про поточного користувача.
- b. Авторизація: Контролер обробляє запити до `api/login` для входу користувача в систему.
- c. Реєстрація: Контролер обробляє запити до `api/signup` для створення нового облікового запису.
- d. Вихід з аккаунту: Контролер обробляє запити до `api/logout` для виходу користувача з облікового запису.
- e. Оновлення токена: Контролер обробляє запити до `api/refresh` для оновлення токена авторизації.

Потік процесів

1. Запит від користувача

- a. Користувач надсилає запит на одну з кінцевих точок (наприклад, `api/login` для входу в систему).

2. Роутинг

- a. Система маршрутизує запит до відповідної кінцевої точки.

3. Обробка контролером

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дата		

а. Контролер отримує запит і виконує відповідну операцію, наприклад, перевірку облікових даних для авторизації або створення нового користувача для реєстрації.

4. Виконання операції

а. Контролер виконує необхідні дії (наприклад, оновлення бази даних, генерацію токена тощо) та формує відповідь.

5. Відповідь користувачу

а. Після обробки запиту контролер повертає відповідь користувачу через кінцеву точку.

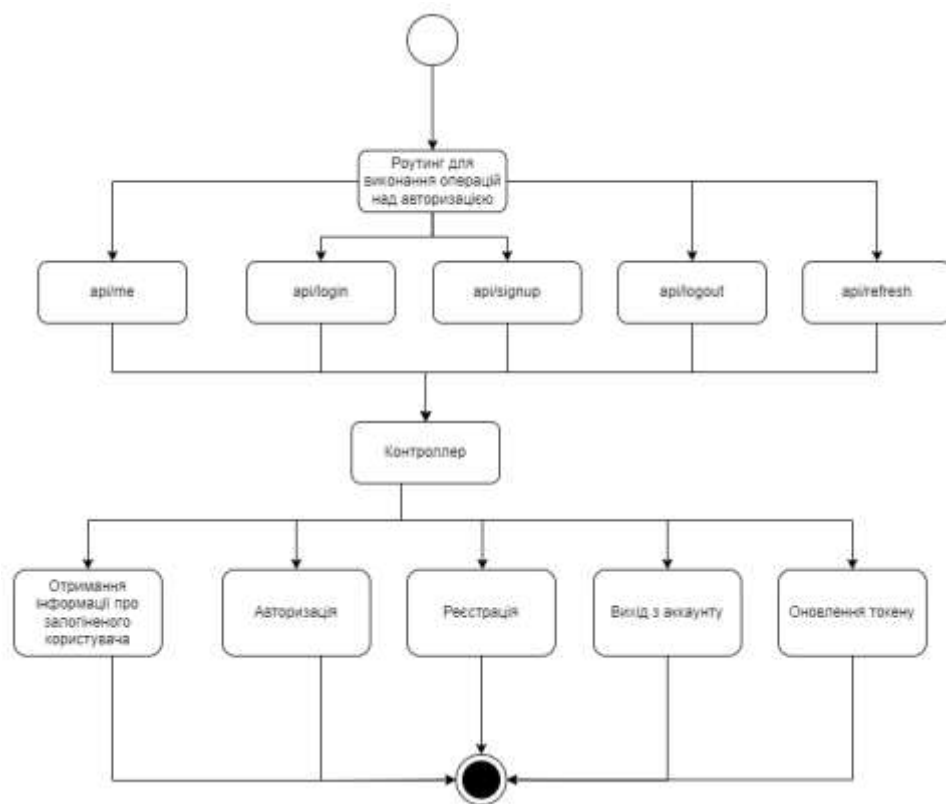


Рис. 2.3. Діаграма роутингу

Діаграма активності реєстрації описує процес реєстрації нового користувача на веб-сайті (рис. 2.4). Вона показує послідовність дій від моменту початку реєстрації до успішного авторизованого входу користувача на сайт.

Опис етапів діаграми

1. Сторінка реєстрації

а. Користувач переходить на сторінку реєстрації.

2. Заповнення полів Email, Password

- а. Користувач заповнює обов'язкові поля для реєстрації, такі як Email і Password.
- 3. Створення нової моделі типу User
 - а. На сервері створюється новий об'єкт користувача з введеними даними (Email і Password).
- 4. Збереження моделі у базі даних
 - а. Новий об'єкт користувача зберігається у базі даних, що завершує процес реєстрації.
- 5. Авторизація
 - а. Після успішної реєстрації здійснюється авторизація користувача, тобто перевірка та підтвердження його облікових даних.
- 6. Користування веб-сайтом
 - а. Після успішної авторизації користувач отримує доступ до всіх функцій веб-сайту і може почати його використовувати.

Потік процесів

- 1. Користувач переходить на сторінку реєстрації
 - а. Цей етап починається, коли користувач відкриває сторінку реєстрації на веб-сайті.
- 2. Заповнення форми реєстрації
 - а. Користувач вводить свою електронну пошту та пароль у відповідні поля форми.
- 3. Створення об'єкта користувача
 - а. На сервері створюється новий об'єкт типу User, який містить введені користувачем дані.
- 4. Збереження в базі даних
 - а. Створений об'єкт користувача зберігається у базі даних. Це означає, що новий користувач успішно зареєстрований у системі.
- 5. Автоматична авторизація
 - а. Після збереження даних новий користувач автоматично авторизується у системі.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						34
Змн.	Арк.	№ докум.	Підпис	Дата		

6. Використання сайту

а. Після успішної авторизації користувач може почати користуватися сайтом з новим обліковим записом.



Рис. 2.4. Діаграма активності реєстрації

Діаграма активності застосування фільтрів пошуку транспортних засобів на веб-сайті за допомогою фільтрів. Вона ілюструє етапи від введення критеріїв пошуку до відображення результатів (рис. 2.5).

Опис етапів діаграми

1. Введення фільтрів пошуку таких як: Рік випуску, Бренд, Модель тощо.

а. Користувач вводить бажані параметри пошуку, такі як рік випуску, бренд, модель та інші характеристики транспортного засобу.

2. Відправлення запиту на отримання транспортних засобів за вказаними фільтрами

а. Після введення фільтрів користувач відправляє запит до серверу на отримання транспортних засобів, що відповідають цим критеріям.

3. Отримання результату

а. Сервер обробляє запит, здійснює пошук у базі даних та повертає результати, які відповідають введеним фільтрам.

4. Відображення результату

а. Отримані результати відображаються на веб-сайті для користувача у зручному для перегляду форматі.

Потік процесів

1. Введення критеріїв пошуку

а. Користувач заповнює поля фільтрів, вказуючи бажані характеристики транспортного засобу.

2. Надсилання запиту

а. Після заповнення критеріїв пошуку користувач ініціює запит до серверу для отримання відповідних транспортних засобів.

3. Обробка запиту сервером

а. Сервер приймає запит, виконує пошук у базі даних відповідно до заданих фільтрів та формує результати.

4. Отримання та відображення результатів

а. Результати пошуку передаються від сервера до клієнта і відображаються на сторінці, де користувач може переглядати їх.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						36
Змн.	Арк.	№ докум.	Підпис	Дата		



Рис. 2.5. Діаграма активності застосування фільтрів пошуку транспортних засобів

Діаграма надає зрозумілу візуалізацію процесу пошуку транспортних засобів з використанням фільтрів, допомагаючи користувачам зрозуміти, як їх введені параметри перетворюються на конкретні результати пошуку.

2.3. Проектування бази даних

Для збереження файлів та інформації в додатку використовується MongoDB, система управління базами даних, яка базується на документній моделі. Після публікації авто зберігається в базі даних, де також фіксуються всі авто на сайті.

База даних системи складається з п'яти таблиць:

1. User model – таблиця, яка зберігає данні користувачів;
2. Vehicle model – таблиця, яка зберігає данні про авто;
3. Token model – таблиця, яка зберігає данні про авторизацію користувача;
4. Favorite model – таблиця, яка зберігає данні про вподобанні авто;

Опис кожної з таблиць бази даних наведений у таблицях 2.1 – 2.4.

Таблиця 2.1

Опис таблиці Users

Поле	Тип	Призначення
email	String	Пошта користувача
password	String	Пароль користувача
role	String	Роль користувача на сайті

Таблиця 2.2

Опис таблиці Vehicle

Поле	Тип	Призначення
brand	String	Марка авто
model	String	Модель авто
year	Number	Рік випуску
body_type	String	Тип кузова
fuel_type	String	Тип палива
engine_volume	String	Об'єм двигуна
power	Date	Тип двигуна
cylinders	Number	Кількість циліндрів
transmission	String	Тип трансмісії

gears	Number	Кількість передач
length	Number	Довжина авто
width	Number	Ширина авто
height	Number	Висота авто
weight	Number	Вага авто
tank_volume	String	Об'єм паливного баку
description	String	Опис авто
max_speed	Number	Швидкість авто
price	Number	Ціна авто

Таблиця 2.3

Опис таблиці Token

Поле	Тип	Призначення
refreshToken	String	Зберігання токена оновлення
user	Schema.Types.ObjectId	Власник токена

Таблиця 2.4

Опис таблиці Favorite

Поле	Тип	Призначення
vehicle	Schema.Types.ObjectId	Вподобане авто
user	Schema.Types.ObjectId	Користувач
time	Date	Дата вподобання

Основні функції серверної частини

1. Авторизація та реєстрація користувачів
 - а. Реалізується через кінцеві точки /login та /signup, де відбувається верифікація користувача та створення нового облікового запису.
2. Управління транспортними засобами

a. Додавання нових транспортних засобів (для адміністратора) через кінцеву точку /vehicle.

b. Отримання списку транспортних засобів через кінцеву точку /vehicles.

c. Оновлення інформації про транспортні засоби через кінцеву точку /vehicle/{id}.

3. Заявки на придбання

a. Створення нових заявок на придбання транспортних засобів.

b. Підтвердження та відхилення заявок (для адміністратора).

4. Управління категоріями

a. Створення, редагування та видалення категорій транспортних засобів.

5. Вподобання транспортних засобів

a. Додавання транспортних засобів до списку вподобаних.

b. Перегляд списку вподобаних транспортних засобів.

Безпека та збереження даних

a. Аутентифікація та авторизація: Використовується токенна система аутентифікації (JWT), де після успішної авторизації користувач отримує токен доступу.

b. Шифрування паролів: Паролі користувачів зберігаються у вигляді хешів.

Для створення інформаційної системи забезпечення підбору автомобілів за заданими критеріями було розроблено кілька таблиць, кожна з яких має своє призначення та специфічні поля. Наведені таблиці забезпечують збереження даних користувачів, інформації про автомобілі, керування токенами для безпеки, а також планування завантаження даних. Нижче надано загальний аналіз кожної таблиці.

Структура бази даних є ключовим елементом для ефективної роботи інформаційної системи забезпечення підбору автомобілів. Таблиця користувачів забезпечує зберігання та управління інформацією про користувачів системи,

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

таблиця автомобілів містить усі необхідні дані про транспортні засоби, таблиця оновлення токенів підтримує безпеку та автентифікацію, а таблиця розкладу завантажень дозволяє планувати та виконувати завдання з завантаження даних. Така організація даних забезпечує високу функціональність та зручність використання системи.

Аналіз функціональних вимог дозволив виділити наступні вимоги до бази даних, що забезпечать програмну реалізацію додатку. На рисунку 2.6. наведено схему таблиць бази даних, яка використовується для роботи веб сайту.

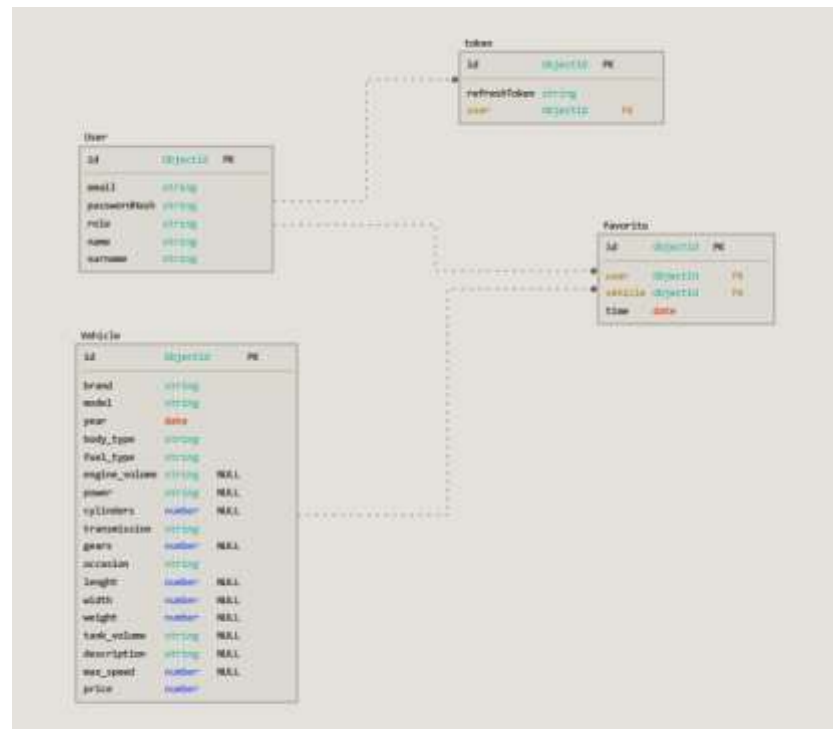


Рис. 2.6. Схема таблиць бази даних

2.4. Розробка функціоналу системи

Цей розділ описує основні функції системи для управління транспортними засобами, включаючи авторизацію, реєстрацію, зміну паролю, створення, отримання, вподобання, редагування та пошук транспортних засобів.

Функція авторизації перевіряє облікові дані користувача і дозволяє йому отримати доступ до системи. Цей код реалізує функцію авторизації користувача з використанням асинхронного запиту до сервера, Redux для управління станом і Axios для HTTP-запитів. Код демонструє використання сучасних технологій

для управління станом додатку і здійснення HTTP-запитів. Він показує, як можна інтегрувати Redux і Axios для реалізації процесу авторизації з асинхронною обробкою, а також обробкою успішних і невдалих результатів.

```
import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export interface ILoginUser {
  email: string;
  password: string;
}

export const login =
  (user: ILoginUser, onSuccess: () => void) =>
  async (dispatch: AppDispatch) => {
    try {
      dispatch(authorizationReducer.actions.LOGIN());
      await $api
        .post('/login', {
          email: user.email,
          password: user.password,
        })
        .then((r) => {
          localStorage.setItem('token', r.data.accessToken);
          dispatch(authorizationReducer.actions.LOGIN_SUCCESS(r.data.user));
          onSuccess();
        });
    } catch (e) {
      console.log(e);
      dispatch(authorizationReducer.actions.LOGIN_ERROR('error'));
    }
  };
};
```

Функція реєстрації створює нового користувача в системі.

```
import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export interface INewUser {
  email: string;
  password: string;
}

export const sign_up =
  (user: INewUser, onSuccess: () => void) => async (dispatch: AppDispatch) => {
    console.log(user);
    try {
      dispatch(authorizationReducer.actions.SIGN_UP());
      await $api
        .post('signup', {
          email: user.email,
```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						42
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        password: user.password,
      })
      .then((r) => {
        localStorage.setItem('token', r.data.accessToken);
        dispatch(authorizationReducer.actions.SIGN_UP_SUCCESS(r.data.user));
        onSuccess();
      });
    } catch (e) {
      console.log(e);
      dispatch(authorizationReducer.actions.SIGN_UP_ERROR('error'));
    }
  }
};

```

Функція зміни паролю дозволяє користувачу змінити свій пароль. Цей код реалізує функцію реєстрації нового користувача з використанням асинхронного запиту до сервера, Redux для управління станом і Axios для HTTP-запитів. Також код демонструє, як використовувати сучасні технології для управління станом додатку і здійснення HTTP-запитів. Він показує, як можна інтегрувати Redux і Axios для реалізації процесу реєстрації нового користувача з асинхронною обробкою, а також обробкою успішних і невдалих результатів.

```

import {AppDispatch} from '../..';
import $api from '../..../axios';
import {authorizationReducer} from '../..../reducers/authorizationReducer';

export interface IChangePassword {
  password: string;
  oldPassword: string;
}

export const changePassword =
  (passwords: IChangePassword, onSuccess: (success: boolean) => void) =>
  async (dispatch: AppDispatch) => {
    try {
      dispatch(authorizationReducer.actions.UPDATE_PASSWORD());
      await $api
        .put('/password', {
          password: passwords.password,
          oldPassword: passwords.oldPassword,
        })
        .then((r) => {
          dispatch(authorizationReducer.actions.UPDATE_PASSWORD_SUCCESS(r.data.user));
          onSuccess(true);
        });
    } catch (e) {
      console.log(e);
      // @ts-ignore
    }
  }

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        dispatch(authorizationReducer.actions.UPDATE_PASSWORD_ERROR(e.response.data.message));
        onSuccess(false);
    }
};

```

Функція створення транспортного засобу додає новий транспортний засіб до системи. Цей код реалізує функцію створення нового транспортного засобу з використанням асинхронного запиту до сервера, Redux для управління станом і Axios для HTTP-запитів. Також він демонструє, як використовувати сучасні технології для управління станом додатку і здійснення HTTP-запитів. Він показує, як можна інтегрувати Redux і Axios для реалізації процесу створення нового транспортного засобу з асинхронною обробкою, а також обробкою успішних і невдалих результатів.

```

import {AppDispatch} from '../..';
import $api from '../..//axios';
import {vehicleReducer} from '../..//reducers/vehicleReducer';

export interface ICreateVehicle {
    brand?: string;
    model?: string;
    year?: string;
    body_type?: string;
    fuel_type?: string;
    engine_volume?: string;
    power?: string;
    cylinders?: string;
    transmission?: string;
    gears?: string;
    occasion?: string;
    length?: string;
    width?: string;
    height?: string;
    weight?: string;
    tank_volume?: string;
    description?: string;
    max_speed?: string;
    price?: string;
    images?: { data: string, fileName: string }[]
}

export const createVehicle =
    (vehicle: ICreateVehicle, onSuccess: (success: boolean) => void) =>
        async (dispatch: AppDispatch) => {
            try {
                dispatch(vehicleReducer.actions.CREATE_VEHICLE());
            }
        }
    );

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						44
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        await $api.post('/vehicle',
            {...vehicle}
        ).then((r) => {
            dispatch(vehicleReducer.actions.CREATE_VEHICLE_SUCCESS());
            onSuccess(true);
        });

    } catch (e) {
        console.log(e);
        // @ts-ignore
        dispatch(vehicleReducer.actions.CREATE_VEHICLE_ERROR(e.response.data.message));
        onSuccess(false);
    }
};

```

Функція отримання транспортних засобів дозволяє користувачеві переглядати список доступних транспортних засобів. Цей код реалізує функцію для отримання списку транспортних засобів з використанням асинхронного запиту до сервера, Redux для управління станом і Axios для HTTP-запитів. Код демонструє, як використовувати сучасні технології для управління станом додатку і здійснення HTTP-запитів. Він показує, як можна інтегрувати Redux і Axios для реалізації процесу отримання списку транспортних засобів з асинхронною обробкою, а також обробкою успішних і невдалих результатів.

```

import {AppDispatch} from '../..';
import $api from '../..../axios';
import {vehicleReducer} from '../..../reducers/vehicleReducer';

export const getVehicles =
    () =>
        async (dispatch: AppDispatch) => {
            try {
                dispatch(vehicleReducer.actions.GET_VEHICLES());
                await $api.post('/vehicles/',
                ).then((r) => {
                    dispatch(vehicleReducer.actions.GET_VEHICLES_SUCCESS(r.data));
                });
            } catch (e) {
                console.log(e);
                dispatch(vehicleReducer.actions.GET_VEHICLES_ERROR('error'));
            }
        };

```

Функція редагування транспортного засобу дозволяє адміністратору змінювати інформацію про транспортний засіб. Цей код демонструє процес

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						45
Змн.	Арк.	№ докум.	Підпис	Дата		

оновлення інформації про транспортний засіб за допомогою Redux для управління станом і Axios для здійснення HTTP-запитів. Він підкреслює важливість обробки асинхронних дій, обробки успішних і невдалих результатів запитів, а також правильного оновлення стану додатку у випадку успішного оновлення транспортного засобу або виникнення помилки.

```
import {AppDispatch} from '../..';
import $api from '../..../axios';
import {vehicleReducer} from '../..../reducers/vehicleReducer';

export interface IUpdateVehicle {
  brand?: string;
  model?: string;
  year?: string;
  body_type?: string;
  fuel_type?: string;
  engine_volume?: string;
  power?: string;
  cylinders?: string;
  transmission?: string;
  gears?: string;
  occasion?: string;
  length?: string;
  width?: string;
  height?: string;
  weight?: string;
  tank_volume?: string;
  description?: string;
  max_speed?: string;
  price?: string;
  images?: { data: string, fileName: string }[]
}

export const updateVehicle =
  (vehicle: IUpdateVehicle, onSuccess: (success: boolean) => void) =>
    async (dispatch: AppDispatch) => {
      try {
        dispatch(vehicleReducer.actions.CREATE_VEHICLE());
        await $api.put('/vehicle/${id}',
          {...vehicle}
        ).then((r) => {
          dispatch(vehicleReducer.actions.CREATE_VEHICLE_SUCCESS());
          onSuccess(true);
        });
      } catch (e) {
        console.log(e);
        // @ts-ignore
      }
    }
  }
```

```

        dispatch(vehicleReducer.actions.CREATE_VEHICLE_ERROR(e.response.data
a.message));
        onSuccess(false);
    }
};

```

Функція пошуку транспортних засобів дозволяє користувачам знаходити транспортні засоби за заданими критеріями. Демонстрація процесу отримання інформації про конкретний транспортний засіб за допомогою Redux для управління станом і Axios для здійснення HTTP-запитів. Він підкреслює важливість обробки асинхронних дій, обробки успішних і невдалих результатів запитів, а також правильного оновлення стану додатку у випадку успішного отримання даних або виникнення помилки.

```

import {AppDispatch} from '../..';
import $api from '../..../axios';
import {vehicleReducer} from '../..../reducers/vehicleReducer';

export const getVehicle =
(id: number) =>
  async (dispatch: AppDispatch) => {
    try {
      dispatch(vehicleReducer.actions.GET_VEHICLES());
      await $api.post(/vehicles/${id},
    ).then((r) => {
        dispatch(vehicleReducer.actions.GET_VEHICLES_SUCCESS(r.data));
      });
    } catch (e) {
      console.log(e);
      dispatch(vehicleReducer.actions.GET_VEHICLES_ERROR('error'));
    }
  };

```

Наведений код забезпечує основний функціонал системи управління транспортними засобами. Користувачі можуть реєструватися, авторизовуватися, змінювати паролі, створювати, отримувати, редагувати та шукати транспортні засоби. Використання асинхронних функцій і Fetch API дозволяє легко взаємодіяти з бекендом і обробляти результати запитів.

Висновки до другого розділу

В даному розділі було розроблено діаграму варіантів використання системи. Проведено розробку загальної схеми роботи програми. Розглянуто та спроектовано алгоритми роботи додатку. Спроектовано та реалізовано базу даних системи. Наведені основні частини коду. Для гарного функціонування була створена деяка кількість компонентів та реалізована схема роботи.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						48
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ОПИС РОБОТИ ТА ТЕСТУВАННЯ СИСТЕМИ

3.1. Встановлення та налаштування системи

Встановлення необхідних інструментів

1. Перед початком, потрібно переконатися, що на комп'ютері встановлені наступні інструменти.

- а. Node.js (включає npm, але ми будемо використовувати Yarn)
- б. Yarn (пакетний менеджер)
- с. MongoDB (встановлений локально або доступ до MongoDB Atlas)

2. Клонування репозиторію

3. Встановлення залежностей

Встановіть необхідні залежності для фронтенду та бекенду, використовуючи Yarn.

Клієнт

- а. Перехід до директорії фронтенду і встановлення залежності (рис. 3.1).

```
PS C:\Work\diplom> cd .\client\  
PS C:\Work\diplom\client> yarn  
yarn install v1.22.22  
[1/4] Resolving packages...  
success Already up-to-date.  
Done in 0.61s.  
PS C:\Work\diplom\client> █
```

Рис. 3.1 – Встановлення залежностей фронтенду

Сервер

- а. Перехід до директорії бекенду і встановлення залежності (рис. 3.2)

```
PS C:\Work\diplom> cd .\server\  
PS C:\Work\diplom\server> yarn  
yarn install v1.22.22  
[1/4] Resolving packages...  
success Already up-to-date.  
Done in 0.15s.  
PS C:\Work\diplom\server> █
```

Рис. 3.2 – Встановлення залежностей бекенду

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

4. Налаштування змінних середовища

а. Створення файлу `.env` у кореневій директорії бекенду і додавання в нього змінні середовища, зокрема `DB_URL` для підключення до MongoDB (рис. 3.3)

```
server > .env
1 PORT=5000
2 DB_URL=mongodb+srv://slumak:tvIwMQk0D8TrNgTD@cluster0.wq4nvjk.mongodb.net/
3 JWT_ACCESS_SECRET=gtp6DZ562D
4 JWT_REFRESH_SECRET=77X9jmTHe7
5 CLIENT_URL=http://localhost:3000
```

Рис. 3.3 – Змінні середовища

5. Запуск MongoDB

а. Використання локальної версії MongoDB, запуск MongoDB серверу. Якщо ж використовувати MongoDB Atlas, потрібно переконаватися, що є доступ до кластеру і що `DB_URL` коректний (рис. 3.4)

```
const start = async () => {
  try {
    await mongoose.connect(process.env.DB_URL, {
      useNewUrlParser: true,
      useUnifiedTopology: true,
    })
    app.listen(PORT, () => {
      console.log(`Server started: http://localhost:${PORT}`)
    })
  } catch (e) {
    console.log(e)
  }
}
start()
```

Рис. 3.4 – Запуск серверу та підключення до бази

6. Запуск бекенду

а. Потрібно переконаватися, в правильності шляху директорії бекенду, і запустити сервер (рис. 3.5)

```
PS C:\Work\diplom\server> yarn start
yarn run v1.22.22
$ node index.js
Server started: http://localhost:5000
```

Рис. 3.5 – Запуск бекенду

- б. Використання Nodemon для автоматичного перезапуску сервера при зміні файлів (рис. 3.6)



```
PS C:\Work\diplom\server> yarn dev
yarn run v1.22.22
$ nodemon index.js
[nodemon] 2.0.22
[nodemon] to restart at any time, enter `rs`
[nodemon] watching path(s): *.*
[nodemon] watching extensions: js,mjs,json
[nodemon] starting `node index.js`
Server started:http://localhost:5000
```

Рис. 3.6 – Запуск бекенду у режимі розробника

7. Запуск фронтенду

- а. Перейдіть до директорії фронтенду і запустіть React-додаток (рис. 3.7)



```
PS C:\Work\diplom\client> yarn dev
yarn run v1.22.22
$ react-scripts start
```

Рис 3.7 – Запуск фронтенду у режимі розробника

3.2. Структура інтерфейсу та опис роботи з системою

1. Сторінка авторизації

Дозволяє користувачам увійти до свого облікового запису (рис. 3.8).

Процедура використання.

- а. Користувач вводить свою електронну пошту та пароль у відповідні поля.
- б. Натискає кнопку "Увійти".
- с. Якщо дані правильні, користувача перенаправляють на головну сторінку або в особистий кабінет.

Рис. 3.8 – Сторінка авторизації користувача

2. Сторінка реєстрації

Дозволяє новим користувачам створити обліковий запис (рис. 3.9).

Процедура використання

- а. Користувач вводить свою електронну пошту, пароль та підтверджує пароль.
- б. Натискає кнопку "Зареєструватися".
- с. Якщо дані правильні і електронна пошта унікальна, користувача перенаправляють на сторінку авторизації.

Рис. 3.9 – Сторінка реєстрації користувача

3. Головна сторінка

Надає основну інформацію про сайт і включає компонент пошуку автомобілів за критеріями (рис. 3.10).

Процедура використання

- а. Користувач бачить поле для пошуку автомобілів за критеріями (марка, модель, рік, тип кузова, тип палива і т.д.).
- б. Вводить бажані параметри у відповідні поля.
- с. Натискає кнопку "Пошук".
- д. Результати пошуку відображаються нижче на тій же сторінці.



Рис. 3.10 – Головна сторінка сайту

4. Сторінка пошуку авто за назвою

Дозволяє користувачам шукати автомобілі за їх назвою або маркою (рис. 3.11).

Процедура використання

- а. Користувач вводить назву автомобіля або його марку у поле пошуку.
- б. Натискає кнопку "Пошук".
- с. Результати пошуку відображаються на сторінці.

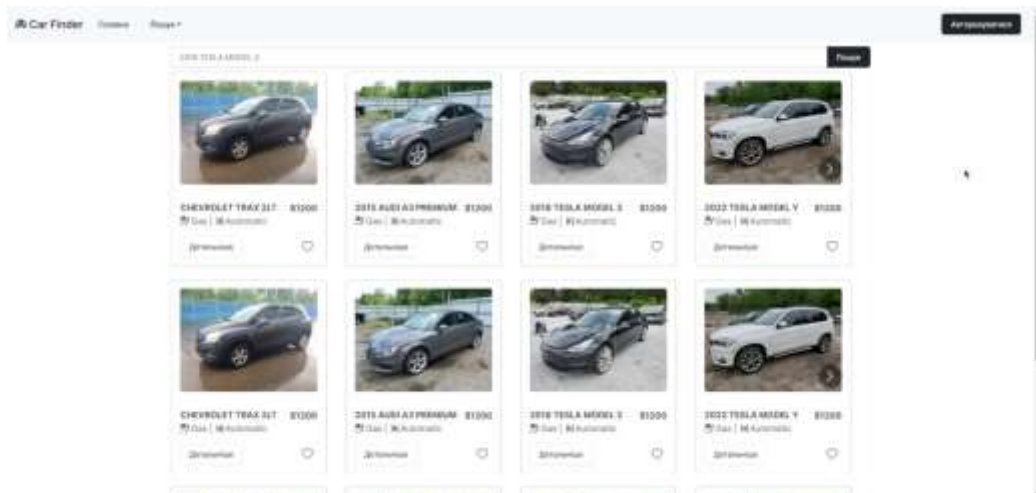


Рис. 3.11 – Пошук авто

5. Особистий кабінет

Дозволяє користувачам переглядати свої вподобані автомобілі, а адміністраторам керувати користувачами та транспортними засобами.

Для звичайних користувачів (рис. 3.12).

- Переглядають список вподобаних автомобілів.
- Можуть видалити автомобіль зі списку вподобань.

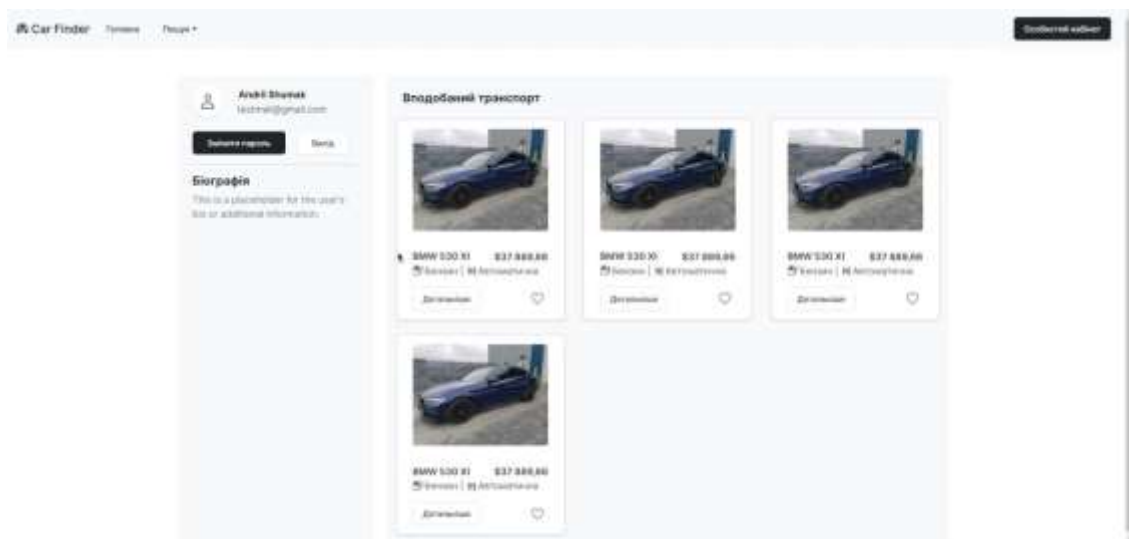


Рис. 3.12 – Особистий кабінет користувача

Для адміністраторів (рис. 3.13).

- Переглядають список користувачів.
- Можуть змінити роль користувача (наприклад, з "користувача" на "адміністратора").
- Переглядають список усіх транспортних засобів.
- Мають можливість видаляти транспортні засоби.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

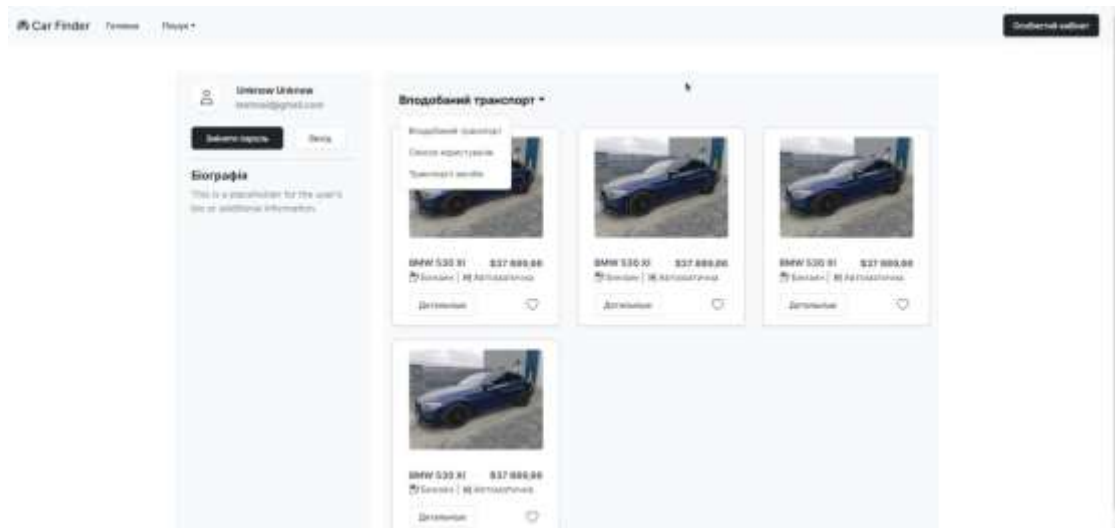


Рис. 3.13 – Особистий кабінет адміністратора

6. Сторінка редагування

Дозволяє адміністраторам редагувати інформацію про автомобілі (рис. 3.14).

Процедура використання

- Адміністратор вибирає автомобіль зі списку.
- Вносить необхідні зміни у форму редагування.
- Натискає кнопку "Зберегти зміни".
- Оновлена інформація зберігається в базі даних.

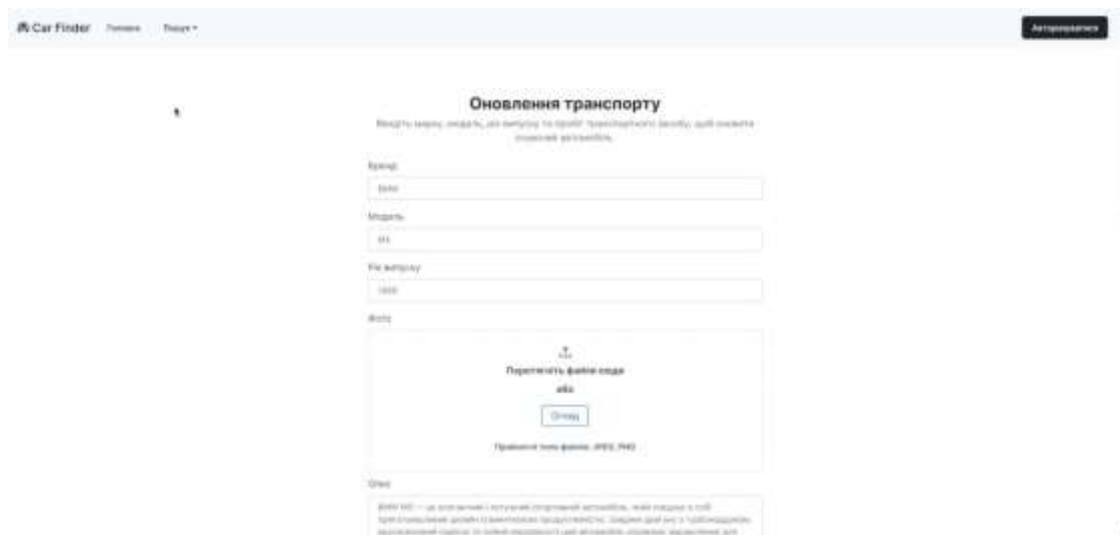


Рис. 3.14 – Форма для редагування авто

7. Сторінка додавання авто

Дозволяє адміністраторам додавати нові автомобілі до бази даних (рис. 3.15).

Процедура використання

- а. Адміністратор заповнює форму з інформацією про новий автомобіль (марка, модель, рік, тип кузова, тип палива, об'єм двигуна, потужність, кількість циліндрів, трансмісія, кількість передач, довжина, ширина, висота, вага, об'єм паливного баку, опис, максимальна швидкість, ціна).
- б. Натискає кнопку "Додати автомобіль".
- с. Новий автомобіль додається до бази даних і стає доступним для пошуку користувачами.

The screenshot shows a web browser window with the title 'Car Finder'. The main heading is 'Додавання транспорту' (Adding transport). Below the heading is a sub-heading: 'Введіть марку, модель, рік випуску та інші транспортні характеристики, щоб додати новий автомобіль.' (Enter the brand, model, year of release and other transport characteristics to add a new car). The form contains several input fields: 'Бренд' (Brand), 'Модель' (Model), 'Рік випуску' (Year of release), and 'Фото' (Photo). The 'Фото' field has a placeholder image and a button 'Вибір' (Select). Below the form is a button 'Додати' (Add). At the bottom, there is a small text block: 'Важко знайти транспортний засіб? Зверніться до нас за допомогою наших спеціалістів.' (Can't find a car? Contact us for help from our specialists.)

Рис. 3.15 – Форма для додавання авто

8. Форма для переглядання повної інформації про авто
- Дозволяє користувачеві переглянути детальну інформацію про транспортний засіб а також фото-матеріали (рис. 3.16).

Процедура використання

- а. Натискає на кнопку детальніше
- б. Відкриття сторінки з детальною інформацією про авто

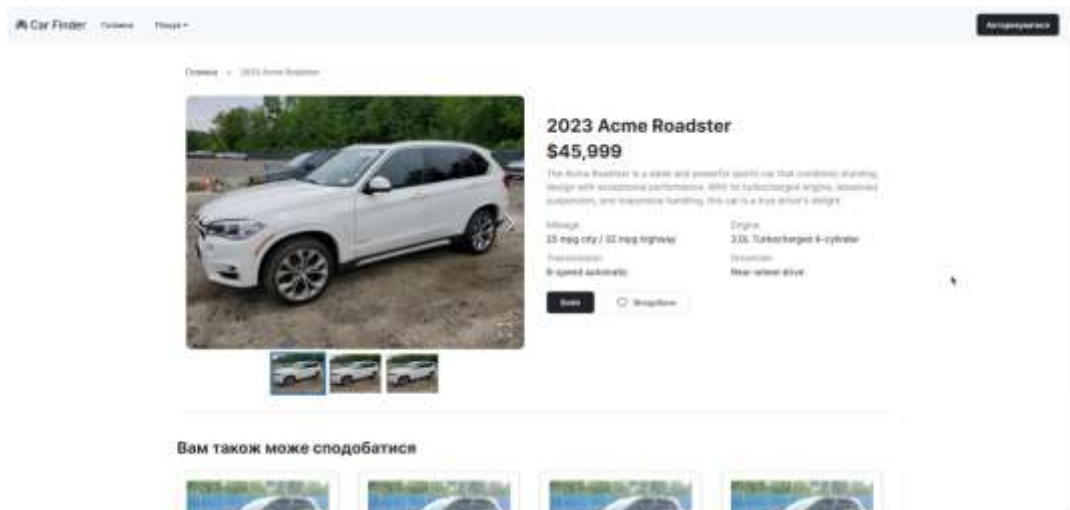


Рис. 3.16 – Форма перегляду авто

Веб-сайт забезпечує зручний та ефективний процес пошуку автомобілів за різними критеріями. Користувачі можуть реєструватися, авторизуватися, здійснювати пошук автомобілів за заданими параметрами та за назвою, а також керувати своїми вподобаними автомобілями через особистий кабінет. Адміністратори мають додаткові можливості для управління користувачами та транспортними засобами, включаючи редагування та додавання нових автомобілів до бази даних.

3.3. Тестування роботи системи

Перевірка валідації форми авторизації. Під час розробки форми авторизації для веб-сайту з підбору автомобілів важливо забезпечити надійність та безпеку введених даних. Нижче розглянуто основні аспекти перевірки валідації для цієї форми:

1. Коректність введення даних
 - а. Введені дані (пошта та пароль) повинні відповідати заданим форматам (рис. 3.17).

Авторизація

Введіть адресу електронної пошти та пароль для доступу до свого облікового запису

Пошта

Не правильний логін або пароль

Пароль

Не правильний логін або пароль

Увійти

Продовжити без авторизації

Не маєте акаунту? [Зареєструйся](#)

Рис. 3.17 – Коректність введення даних

- б. Пароль має відповідати мінімальним вимогам щодо довжини та складності (рис. 3.18).

Реєстрація

Введіть адресу електронної пошти, пароль і підтвердьте пароль, щоб створити обліковий запис

Пошта

Пароль

Пароль має бути мінімум 8 символів

Підтвердьте пароль

Створити акаунт

Продовжити без авторизації

Вже зареєстровані? [Авторизуйся](#)

Рис. 3.18 – Мінімальні вимоги до паролю

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

2. Запобігання SQL-ін'єкціям та іншим атакам

а. Всі введені дані мають бути очищені від потенційно небезпечних символів перед використанням у SQL-запитах або інших операціях з даними.

Перевірка валідації форми реєстрації. При реєстрації нового користувача важливо виконати додаткові перевірки, щоб забезпечити унікальність поштової адреси та ідентичність паролів.

1. Унікальність пошти (рис. 3.19).

а. Перед додаванням нового користувача до бази даних необхідно перевірити, чи не існує вже користувача з такою ж поштовою адресою.

Реєстрація
Введіть адресу електронної пошти, пароль і підтвердьте пароль, щоб створити обліковий запис

Пошта
testmail@gmail.com
Ця електронна пошта вже використовується

Пароль

Підтвердьте пароль

Створити аккаунт

Продовжити без авторизації

Вже зареєстровані? [Авторизуйся](#)

Рис. 3.19 – Реєстрація нового користувача

2. Ідентичність паролів

а. Поля для введення пароля та підтвердження пароля мають бути ідентичними (рис. 3.20).

Реєстрація
Введіть адресу електронної пошти, пароль і підтвердьте пароль, щоб створити обліковий запис

Пошта
testmail@gmail.com

Пароль

Підтвердьте пароль

Паролі не сходяться!

Створити аккаунт

Продовжити без авторизації

Вже зареєстровані? [Авторизуйся](#)

Рис. 3.20 – Ідентичність паролів

3. Заповнена форма реєстрації без помилок (рис. 3.21).

Реєстрація
Введіть адресу електронної пошти, пароль і підтвердьте пароль, щоб створити обліковий запис

Пошта
testmail1@gmail.com

Пароль

Підтвердьте пароль

Створити аккаунт

Продовжити без авторизації

Вже зареєстровані? [Авторизуйся](#)

Рис. 3.21 – Правильно заповнення форма

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						60
Змн.	Арк.	№ докум.	Підпис	Дата		

Перевірка валідації форми додавання транспортних засобів. При додаванні нових транспортних засобів важливо, щоб всі обов'язкові поля були заповнені коректно.

1. Валідація обов'язкових полів

а. Усі поля, які є обов'язковими для заповнення (наприклад, марка автомобіля, модель, рік випуску тощо), повинні бути заповнені користувачем перед надсиланням форми (рис. 3.22).

Додавання транспорту
Введіть марку, модель, рік випуску та пробіг транспортного засобу, щоб додати новий автомобіль.

Бренд
BMW
Це поле обов'язкове!

Модель
M5
Це поле обов'язкове!

Рік випуску
1999
Це поле обов'язкове!

Фото
Перетягніть файли сюди
або

Рис. 3.22 – Валідація додавання транспортного засобу

Перевірка валідації форми при всіх заповнених полях. При додаванні нових транспортних засобів важливо, щоб всі обов'язкові поля були заповнені коректно.

Висновки до третього розділу

В цьому розділі було описано процес встановлення та налаштування системи. Наведено опис роботи з системою, на основі якого зрозумілі можливості користувача. Було проведено тестування вебсистеми. Виправленні незначні помилки. Система працює коректно та виконує поставлені задачі.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						61
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано інформаційну систему забезпечення підбору автомобілів за заданими критеріями. Під час розробки цього вебпроекту було вирішено такі завдання:

1. Поставлене завдання щодо реалізації проекту. Проаналізовано існуючі аналоги веб-додатку та продумано можливий функціонал на основі аналізу аналогів. Було обрано такі засоби розробки як: фронтенд - HTML, CSS, JavaScript, фреймворк (React, TypeScript, Bootstrap, Redux) для створення динамічних та інтерактивних інтерфейсів. Для Бекенду Node.js, фреймворк (Express, jsonwebtoken, Nodemon, Mongoose) для забезпечення обробки запитів та логіки додатку, та MongoDB для зберігання та управління даними.

2. Був визначений аналіз основних вимог до додатку. В нього увійшли: бізнес-вимоги, вимоги-користувачі, системні, функціональні та нефункціональні вимоги, бізнес-правила та антивимоги.

3. За допомогою діаграми варіантів використання було відображено сценарії використання системи та користувачів системи, які використовують її функції.

4. Визначено архітектуру системи, що розробляється, встановлено залежності між програмними компонентами. За допомогою діаграм активностей описано динамічні аспекти поведінки системи у вигляді блок-схеми.

5. Наведено основний код і пояснено, як проходило виконання роботи. Описано інструкцію користування вебсистеми та проведено тестування.

Отже, система є завершеним продуктом, який може нормально функціонувати. Він має простий та зрозумілий інтерфейс та не складний у використанні

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Cars.com [Електронний ресурс]. – 1998. – Режим доступу до ресурсу: <https://www.cars.com>.
2. autotrader.com [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: <https://www.autotrader.com>.
3. CarGurus [Електронний ресурс]. – 2006. – Режим доступу до ресурсу: <https://www.cargurus.com>.
4. HTML [Електронний ресурс] // WHATWG. – 1993. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/HTML>.
5. CSS [Електронний ресурс] // CSS Working Group. – 1996. – Режим доступу до ресурсу: <https://developer.mozilla.org/en-US/docs/Web/CSS>.
6. React [Електронний ресурс] // Meta Platforms. – 2013. – Режим доступу до ресурсу: <https://legacy.reactjs.org/docs/getting-started.html>.
7. TypeScript [Електронний ресурс] // Microsoft. – 2012. – Режим доступу до ресурсу: <https://www.typescriptlang.org/docs/>.
8. BootStrap [Електронний ресурс] // Bootstrap Core Team. – 2011. – Режим доступу до ресурсу: <https://getbootstrap.com/docs/4.1/getting-started/introduction/>.
9. Redux [Електронний ресурс] // Dan Abramovd, Andrew Clarkd, Mark Eriksond і Timothy Dorr. – 2015. – Режим доступу до ресурсу: <https://redux.js.org>.
10. Node.js [Електронний ресурс] // OpenJS Foundation. – 2009. – Режим доступу до ресурсу: <https://nodejs.org/docs/latest/api/>.
11. Express [Електронний ресурс] // TJ Holowaychuk, StrongLoop. – 2010. – Режим доступу до ресурсу: <https://expressjs.com>.
12. jsonwebtoken [Електронний ресурс] // Michael B. Jones Microsoft John Bradley Ping Identity Nat Sakimura NRI. – 2010. – Режим доступу до ресурсу: <https://www.npmjs.com/package/jsonwebtoken>.
13. nodemon [Електронний ресурс] // OpenJS Foundation. – 2011. – Режим доступу до ресурсу: <https://nodemon.io>.

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дата		

14. Mongoose [Електронний ресурс] // OpenJS Foundation. – 2015. – Режим доступу до ресурсу: <https://mongoosejs.com/docs/>.
15. MongoDB [Електронний ресурс] // MongoDB Inc. – 2009. – Режим доступу до ресурсу: <https://www.mongodb.com/docs/>.
16. React-Hook-Form [Електронний ресурс] // Vercel. – 2019. – Режим доступу до ресурсу: <https://react-hook-form.com>.
17. React-Image-Gallery. [Електронний ресурс] // xiaolin. – 2015. – Режим доступу до ресурсу: <https://www.npmjs.com/package/react-image-gallery>.
18. Dotenv [Електронний ресурс] // motdotla. – 2013. – Режим доступу до ресурсу: <https://www.npmjs.com/package/dotenv>.
19. npm [Електронний ресурс] // NodeJS. – 2011. – Режим доступу до ресурсу: <https://www.npmjs.com>.
20. yarn [Електронний ресурс] // yarnpkg. – 2016. – Режим доступу до ресурсу: <https://yarnpkg.com>.

ДОДАТКИ

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						65
Змн.	Арк.	№ докум.	Підпис	Дата		

package.json

```

{
  "name": "client",
  "version": "0.1.0",
  "private": true,
  "dependencies": {
    "@patternfly/react-core": "^5.3.3",
    "@reduxjs/toolkit": "^2.2.5",
    "@testing-library/jest-dom": "^5.17.0",
    "@testing-library/react": "^13.4.0",
    "@testing-library/user-event": "^13.5.0",
    "@types/jest": "^27.5.2",
    "@types/node": "^16.18.98",
    "@types/react": "^18.3.3",
    "@types/react-dom": "^18.3.0",
    "@types/react-image-gallery": "^1.2.4",
    "@types/react-router-bootstrap": "^0.26.6",
    "@types/react-router-dom": "^5.3.3",
    "axios": "^1.7.2",
    "bootstrap": "^5.3.3",
    "eslint-config-prettier": "^9.1.0",
    "eslint-import-resolver-typescript": "^3.6.1",
    "eslint-plugin-prettier": "^5.1.3",
    "jwt-decode": "^4.0.0",
    "prettier": "^3.3.2",
    "react": "^18.3.1",
    "react-bootstrap": "^2.10.2",
    "react-bootstrap-icons": "^1.11.4",
    "react-dom": "^18.3.1",
    "react-hook-form": "^7.52.0",
    "react-image-gallery": "^1.3.0",
    "react-multi-carousel": "^2.8.5",
    "react-redux": "^9.1.2",
    "react-router-bootstrap": "^0.26.2",
    "react-router-dom": "^6.23.1",
    "react-scripts": "5.0.1",
    "typescript": "^4.4.3",
    "web-vitals": "^2.1.4"
  },
  "devDependencies": {
    "@typescript-eslint/eslint-plugin": "^5.0.0",
    "@typescript-eslint/parser": "^5.0.0",
    "eslint": "^8.57.0",
    "eslint-config-airbnb": "^19.0.4",
    "eslint-plugin-import": "^2.29.1",
    "eslint-plugin-jsx-ally": "^6.8.0",
    "eslint-plugin-react": "^7.34.2",
    "eslint-plugin-react-hooks": "^4.6.2"
  }
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    },
    "scripts": {
      "start": "react-scripts start",
      "lint": "eslint \"**/*.ts,tsx\"",
      "format": "eslint \"**/*.ts,tsx\" --fix",
      "prettier": "prettier"
    },
    "eslintConfig": {
      "extends": [
        "react-app",
        "react-app/jest"
      ]
    },
    "browserslist": {
      "production": [
        ">0.2%",
        "not dead",
        "not op_mini all"
      ],
      "development": [
        "last 1 chrome version",
        "last 1 firefox version",
        "last 1 safari version"
      ]
    }
  }
}

```

index.tsx

```

import React from 'react';
import ReactDOM from 'react-dom/client';
import './assets/bootstrap.min.css';
import './assets/styles.css';
import { BrowserRouter } from 'react-router-dom';
import { Provider } from 'react-redux';
import App from './App';
import { setupStore } from './redux/index';

const root = ReactDOM.createRoot(
  document.getElementById('root') as HTMLElement
);

const store = setupStore();

root.render(
  <Provider store={store}>
    <BrowserRouter>
      <App />
    </BrowserRouter>
  </Provider>

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						67
Змн.	Арк.	№ докум.	Підпис	Дата		

);

App.tsx

```
import React, { useEffect } from 'react';
import { Route, Routes } from 'react-router-dom';
import Layout from './pages/Layout';
import Welcome from './pages/shared/Welcome';
import Login from './pages/guest/Login';
import Signup from './pages/guest/Signup';
import AlreadyAuthorized from './pages/shared/errors/AlreadyAuthorized';
import Vehicle from './pages/shared/Vehicle';
import { useAppDispatch, useAppSelector } from './hooks/redux';
import { jwtDecode } from 'jwt-decode';
import { refreshToken } from './redux/actions/authorization/refreshAction';
import Profile from './pages/shared/Profile';
import CreateVehicleForm from './pages/admin/CreateVehicleForm';

function App() {
  const dispatch = useAppDispatch();
  const { isAuthenticated } = useAppSelector(
    (state) => state.authorizationReducer
  );
  useEffect(() => {
    const token = localStorage.getItem('token');
    if (token) {
      const decodedToken = jwtDecode(token);
      const currentTime = Date.now() / 1000;
      if (decodedToken.exp! < currentTime) {
        dispatch(refreshToken());
      }
    }
  }, []);

  return (
    <Routes>
      <Route element={<Layout />}>
        <Route path="/" element={<Welcome />} />
        <Route path="/vehicle" element={<Vehicle />} />
        <Route path="/profile" element={<Profile />} />
        <Route path="/vehicles/new" element={<CreateVehicleForm />} />
        {isAuthenticated ? (
          <>
            <Route path="/login" element={<AlreadyAuthorized />} />
            <Route path="/signup" element={<AlreadyAuthorized />} />
          </>
        ) : (
          <>
            <Route path="/login" element={<Login />} />
            <Route path="/signup" element={<Signup />} />
          </>
        )}
      </Routes>
    )
  );
}
```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </Route>
      </Routes>
    );
  }

```

```

export default App;

```

setTitle.ts

```

export default function (title: string) {
  document.title = title;
}

```

config.ts

```

export const responsive1 = {
  megalarge: {
    breakpoint: { max: 4000, min: 1200 },
    items: 4,
  },
  large: {
    breakpoint: { max: 1200, min: 991 },
    items: 3,
  },
  medium: {
    breakpoint: { max: 991, min: 767 },
    items: 2,
  },
  mobile: {
    breakpoint: { max: 767, min: 0 },
    items: 1,
  },
};

```

Icon.ts

```

export interface IIconProps {
  width?: number;
  height?: number;
  className?: string;
}

```

redux/index.ts

```

import { combineReducers, configureStore } from '@reduxjs/toolkit';
import authorizationReducer from './reducers/authorizationReducer';
import usersReducer from './reducers/usersReducer';
// import wareReducer from './reducers/wareReducer';
// import NotificationReducer from './reducers/NotificationReducer';
// import UsersReducer from './reducers/usersReducer';

const rootReducer = combineReducers({
  authorizationReducer,
  usersReducer,
  // wareReducer,

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    // NotificationReducer,
  });
  export const setupStore = () => {
    return configureStore({
      reducer: rootReducer,
    });
  };

  export type RootState = ReturnType<typeof rootReducer>;
  export type AppStore = ReturnType<typeof setupStore>;
  export type AppDispatch = AppStore['dispatch'];
  usersReducer.ts

import { PayloadAction, createSlice } from '@reduxjs/toolkit';
import { IUser } from '../types/User';

interface IAuthorizationReducerState {
  users: IUser[];
  isLoading: boolean;
  error: any;
}

const initialState: IAuthorizationReducerState = {
  users: [],
  error: '',
  isLoading: false,
};

export const usersReducer = createSlice({
  name: 'usersReducer',
  initialState,
  reducers: {
    GET_ALL_USERS(state) {
      state.error = '';
      state.isLoading = true;
    },
    GET_ALL_USERS_SUCCESS(state, action: PayloadAction<IUser[]>) {
      state.error = '';
      state.isLoading = false;
      state.users = action.payload;
    },
    GET_ALL_USERS_ERROR(state, action: PayloadAction<any>) {
      state.error = action.payload;
      state.isLoading = false;
    },
  },
});

export default usersReducer.reducer;
authReducer.ts

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { PayloadAction, createSlice } from '@reduxjs/toolkit';
import { IUser } from '../../types/User';

interface IAuthorizationReducerState {
  user: IUser | null;
  isLoading: boolean;
  error: any;
  isAuthorized: boolean;
}

const initialState: IAuthorizationReducerState = {
  user: null,
  error: '',
  isLoading: false,
  isAuthorized: false,
};

export const authorizationReducer = createSlice({
  name: 'authorizationReducer',
  initialState,
  reducers: {
    LOGIN(state) {
      state.error = '';
      state.isLoading = true;
    },
    LOGIN_SUCCESS(state, action: PayloadAction<IUser>) {
      state.error = '';
      state.isLoading = false;
      state.isAuthorized = true;
      state.user = action.payload;
    },
    LOGIN_ERROR(state, action: PayloadAction<any>) {
      state.error = action.payload;
      state.isLoading = false;
      state.isAuthorized = false;
      state.user = null;
    },
    LOGOUT(state) {
      state.error = '';
      state.isLoading = true;
    },
    LOGOUT_SUCCESS(state, action: PayloadAction<IUser>) {
      state.error = '';
      state.isLoading = false;
      state.isAuthorized = false;
      state.user = null;
    },
    LOGOUT_ERROR(state, action: PayloadAction<any>) {
      state.error = action.payload;
      state.isLoading = false;
    },
  },
});

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

```

SIGN_UP(state) {
  state.isLoading = true;
},
SIGN_UP_SUCCESS(state, action: PayloadAction<IUser>) {
  state.isLoading = false;
  state.isAuthorized = true;
  state.user = action.payload;
  state.error = '';
},
SIGN_UP_ERROR(state, action: PayloadAction<any>) {
  state.isLoading = false;
  state.error = action.payload;
},
REFRESH_TOKEN(state) {
  state.error = '';
  state.isLoading = true;
},
REFRESH_TOKEN_SUCCESS(state, action: PayloadAction<IUser>) {
  state.error = '';
  state.isLoading = false;
  state.isAuthorized = true;
  state.user = action.payload;
},
REFRESH_TOKEN_ERROR(state) {
  state.isLoading = false;
  state.isAuthorized = false;
  state.user = null;
},
SIGN_OUT(state) {
  state.isLoading = true;
},
SIGN_OUT_SUCCESS(state) {
  state.isLoading = false;
  state.isAuthorized = false;
  state.user = null;
  state.error = '';
},
SIGN_OUT_ERROR(state, action: PayloadAction<any>) {
  state.isLoading = false;
  state.error = action.payload;
},
},
});

```

```

export default authorizationReducer.reducer;
getUsersAction.ts

```

```

import { AppDispatch } from '../..';
import $api from '../..../axios';
import { usersReducer } from '../..../reducers/usersReducer';

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

```
export const getUsers = () => async (dispatch: AppDispatch) => {
  try {
    dispatch(usersReducer.actions.GET_ALL_USERS());
    await $api.get('/users').then((r) => {
      dispatch(usersReducer.actions.GET_ALL_USERS_SUCCESS(r.data));
    });
  } catch (e) {
    console.log(e);
    dispatch(usersReducer.actions.GET_ALL_USERS_ERROR('error'));
  }
};
```

signupAction.ts

```
import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export interface INewUser {
  email: string;
  password: string;
}

export const sign_up =
  (user: INewUser, onSuccess: () => void) => async (dispatch: AppDispatch) => {
    console.log(user);
    try {
      dispatch(authorizationReducer.actions.SIGN_UP());
      await $api
        .post('signup', {
          email: user.email,
          password: user.password,
        })
        .then((r) => {
          localStorage.setItem('token', r.data.accessToken);
          dispatch(authorizationReducer.actions.SIGN_UP_SUCCESS(r.data.user));
          onSuccess();
        });
    } catch (e) {
      console.log(e);
      dispatch(authorizationReducer.actions.SIGN_UP_ERROR('error'));
    }
  };
};
```

refreshAction.ts

```
import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export const refreshToken = () => async (dispatch: AppDispatch) => {
  try {
    dispatch(authorizationReducer.actions.REFRESH_TOKEN());
    await $api.get('/refresh').then((r) => {
```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						73
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        localStorage.setItem('token', r.data.accessToken);
        dispatch(authorizationReducer.actions.REFRESH_TOKEN_SUCCESS(r.data.user));
    });
} catch (e) {
    console.log(e);
    dispatch(authorizationReducer.actions.REFRESH_TOKEN_ERROR());
}
};

```

logout.ts

```

import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export const logout = () => async (dispatch: AppDispatch) => {
    try {
        dispatch(authorizationReducer.actions.LOGOUT());
        await $api.post('/logout').then((r) => {
            localStorage.removeItem('token');
            dispatch(authorizationReducer.actions.LOGOUT_SUCCESS(r.data.user));
        });
    } catch (e) {
        console.log(e);
        dispatch(authorizationReducer.actions.LOGOUT_ERROR('error'));
    }
};

```

login.ts

```

import { AppDispatch } from '../..';
import $api from '../..../axios';
import { authorizationReducer } from '../..../reducers/authorizationReducer';

export interface ILoginUser {
    email: string;
    password: string;
}

export const login =
    (user: ILoginUser, onSuccess: () => void) =>
    async (dispatch: AppDispatch) => {
        try {
            dispatch(authorizationReducer.actions.LOGIN());
            await $api
                .post('/login', {
                    email: user.email,
                    password: user.password,
                })
                .then((r) => {
                    localStorage.setItem('token', r.data.accessToken);
                    dispatch(authorizationReducer.actions.LOGIN_SUCCESS(r.data.user));
                    onSuccess();
                });
        }
    };

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						74
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    } catch (e) {
      console.log(e);
      dispatch(authorizationReducer.actions.LOGIN_ERROR('error'));
    }
  };
};

```

Laout.tsx

```

import React from 'react';
import { Nav, Navbar, NavDropdown } from 'react-bootstrap';
import { Outlet, useNavigate } from 'react-router-dom';
import { LinkContainer } from 'react-router-bootstrap';
import { useDispatch, useSelector } from 'react-redux';
import { useAppSelector } from '../hooks/redux';
import { Link } from 'react-router-dom';
import {
  Binoculars,
  BinocularsFill,
  CarFront,
  DoorOpen,
  PeopleFill,
  PersonCircle,
  Search,
} from 'react-bootstrap-icons';
import { useLogout } from '../hooks/useLogout';

export default function Layout() {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const { logoutHandler } = useLogout();
  const { user, isAuthorized } = useAppSelector(
    (state) => state.authorizationReducer
  );

  const goOnLogin = () => navigate('/login');
  return (
    <>
      <Navbar bg="light" expand="lg">
        <Navbar.Brand className={ 'ms-3' } href="/">
          <CarFront className="mb-1" height={20} width={20} /> Car Finder
        </Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <LinkContainer className="bg-light me-2" to={"/"}>
              <Nav.Link>Головна</Nav.Link>
            </LinkContainer>

            <NavDropdown className={ 'text-primary' } title={ 'Пошук' }>
              <LinkContainer to={"/catalog"}>
                <NavDropdown.Item>
                  Звичайний пошук{' '}
                </NavDropdown.Item>
              </LinkContainer>
            </NavDropdown>
          </Nav>
        </Navbar.Collapse>
      </Navbar>
      <Outlet />
    </>
  );
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <Search className={'mb-1'} height={'18'} width={'18'} />
      </NavDropdown.Item>
    </LinkContainer>
    <LinkContainer to={'/catalog'}>
      <NavDropdown.Item>
        Розширений пошук{' '}
        <BinocularsFill
          className={'mb-1'}
          height={'18'}
          width={'18'}
        />
      </NavDropdown.Item>
    </LinkContainer>
  </NavDropdown>
  <LinkContainer className="bg-light me-2" to={'/'}>
    <Nav.Link>Про нас</Nav.Link>
  </LinkContainer>
  <LinkContainer className="bg-light me-2" to={'/'}>
    <Nav.Link>Контакти</Nav.Link>
  </LinkContainer>
</Nav>
<div className={'me-4'}>
  {!isAuthorized ? (
    <button className="btn btn-dark" onClick={goOnLogin}>
      Авторизуватися
    </button>
  ) : (
    <Link
      to={'/profile'}
      className="btn btn-dark"
      onClick={goOnLogin}
    >
      Особистий кабінет
    </Link>
  )}
</div>
</Navbar.Collapse>
</Navbar>
<main>
  <Outlet />
</main>
</>
);
}

```

Welcome.tsx

```

import React from 'react';
import { Nav, Navbar, NavDropdown } from 'react-bootstrap';
import { Outlet, useNavigate } from 'react-router-dom';
import { LinkContainer } from 'react-router-bootstrap';
import { useDispatch, useSelector } from 'react-redux';

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						76
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { useAppSelector } from '../hooks/redux';
import { Link } from 'react-router-dom';
import {
  Binoculars,
  BinocularsFill,
  CarFront,
  DoorOpen,
  PeopleFill,
  PersonCircle,
  Search,
} from 'react-bootstrap-icons';
import { useLogout } from '../hooks/useLogout';

export default function Layout() {
  const dispatch = useDispatch();
  const navigate = useNavigate();
  const { logoutHandler } = useLogout();
  const { user, isAuthorized } = useAppSelector(
    (state) => state.authorizationReducer
  );

  const goOnLogin = () => navigate('/login');
  return (
    <>
      <Navbar bg="light" expand="lg">
        <Navbar.Brand className={ 'ms-3' } href="/">
          <CarFront className="mb-1" height={20} width={20} /> Car Finder
        </Navbar.Brand>
        <Navbar.Toggle aria-controls="basic-navbar-nav" />
        <Navbar.Collapse id="basic-navbar-nav">
          <Nav className="me-auto">
            <LinkContainer className="bg-light me-2" to={"/"}>
              <Nav.Link>Головна</Nav.Link>
            </LinkContainer>

            <NavDropdown className={ 'text-primary' } title={ 'Пошук' }>
              <LinkContainer to={"/catalog"}>
                <NavDropdown.Item>
                  Звичайний пошук{ ' ' }
                  <Search className={ 'mb-1' } height={ '18' } width={ '18' } />
                </NavDropdown.Item>
              </LinkContainer>
              <LinkContainer to={"/catalog"}>
                <NavDropdown.Item>
                  Розширений пошук{ ' ' }
                  <BinocularsFill
                    className={ 'mb-1' }
                    height={ '18' }
                    width={ '18' }
                  />
                </NavDropdown.Item>
              </LinkContainer>
            </Nav>
          </Navbar.Collapse>
        </Navbar>
    </>
  );
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </LinkContainer>
      </NavDropdown>
      <LinkContainer className="bg-light me-2" to={'/'}>
        <Nav.Link>Про нас</Nav.Link>
      </LinkContainer>
      <LinkContainer className="bg-light me-2" to={'/'}>
        <Nav.Link>Контакти</Nav.Link>
      </LinkContainer>
    </Nav>
    <div className={ 'me-4' }>
      {!isAuthorized ? (
        <button className="btn btn-dark" onClick={goOnLogin}>
          Авторизуватися
        </button>
      ) : (
        <Link
          to={'/profile'}
          className="btn btn-dark"
          onClick={goOnLogin}
        >
          Особистий кабінет
        </Link>
      )}
    </div>
  </Navbar.Collapse>
</Navbar>
<main>
  <Outlet />
</main>
</>
);
import React, { useRef } from 'react';
import { Button, Card, Container } from 'react-bootstrap';
import Carousel from 'react-multi-carousel';
import 'react-multi-carousel/lib/styles.css';
import ItemCard from '../components/Cards/ItemCard';
import { responsive1 } from '../utils/config';
import Search from '../components/Search/Search';
import setTitle from '../utils/setTitle';

export default function Welcome() {
  setTitle('Головна');

  const elementRef = useRef(null);

  const handleScroll = () => {
    if (elementRef.current) {
      // @ts-ignore
      elementRef.current.scrollToView({ behavior: 'smooth' });
    }
  };
};

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return (
  <Container fluid>
    <Container className="mt-5">
      <h4 className="fw-bold">Ласкаво просимо на наш автомобільний ринок</h4>
      <p className="text-muted">
        Відкрийте для себе автомобіль своєї мрії за допомогою нашого простого
        у використанні інструмента пошуку. Перегляньте широкий вибір
        транспортних засобів, порівняйте ціни та замовте тест-драйв. Почніть
        свою подорож, щоб знайти ідеальний автомобіль сьогодні!
      </p>
      <p className="text-muted">
        Наш автомобільний ринок пропонує різноманітний асортимент автомобілів
        провідних брендів, включаючи седани, позашляховики, хетчбеки та
        вантажівки. Незалежно від того, чи шукаєте ви економічний автомобіль
        для приміських поїздок чи потужний спортивний автомобіль, у нас є щось
        для кожного.
      </p>

      <p className="text-muted">
        За допомогою наших зручних фільтрів пошуку ви можете легко звузити
        вибір на основі ваших уподобань, таких як тип кузова, марка, модель,
        ціновий діапазон, тип палива та пробіг. Наші докладні списки
        автомобілів нададуть вам всю інформацію, необхідну для прийняття
        зваженого рішення.
      </p>

      <button
        onClick={handleScroll}
        className="btn btn-dark mt-3 py-3 py-sm-2"
      >
        Почати пошук
      </button>
    </Container>
    <Container ref={elementRef} className="mt-5">
      <Search showResults />
    </Container>
    <Container className="mt-4">
      <div className="row gap-4 ms-4"></div>
    </Container>
  </Container>
);
}
}

```

Vehicle.tsx

```

import React, { useEffect, useRef, useState } from 'react';
import { Breadcrumb, Container } from 'react-bootstrap';
import { ArrowLeftSquare, Heart } from 'react-bootstrap-icons';
import ReactImageGallery from 'react-image-gallery';
import { Link } from 'react-router-dom';
import ItemCard from '../components/Cards/ItemCard';

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						79
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import Carousel from 'react-multi-carousel';
import { responsive1 } from '../../utils/config';
import CardsCarousel from '../../components/Cards/CardsCarousel';
import setTitle from '../../utils/setTitle';

export default function Vehicle() {
  setTitle('NAME');

  const handleDragStart = (e: any) => e.preventDefault();

  const [nav1, setNav1] = useState(null);
  const [nav2, setNav2] = useState(null);
  let sliderRef1 = useRef(null);
  let sliderRef2 = useRef(null);

  useEffect(() => {
    // @ts-ignore
    setNav1(sliderRef1);
    // @ts-ignore
    setNav2(sliderRef2);
  }, []);
  const images = [
    {
      original:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
      thumbnail:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
    },
    {
      original:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
      thumbnail:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
    },
    {
      original:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
      thumbnail:
        'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/8176e1c1b3914b7e8a9f8ad
d1b83c5a8_ful.jpg',
    },
  ],
  // Додайте більше изображений по необходимости
];
return (
  <Container fluid>
    <Container>

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						80
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<div className="row align-items-center">
  <div className="col-6">
    <nav className="mt-4">
      <ol className="breadcrumb border-0 shadow-none">
        <li className="breadcrumb-item">
          <Link className="text-dark text-decoration-none" to="/">
            Головна
          </Link>
        </li>
        <li className="breadcrumb-item active">2023 Acme Roadster</li>
      </ol>
    </nav>
    <div className="px-3">
      <ReactImageGallery showPlayButton={false} items={images} />
    </div>
  </div>
  <div className="col-6 mb-4">
    <h2 className="fw-bold">2023 Acme Roadster</h2>
    <h2 className="fw-bold">$45,999</h2>
    <p className="text-muted">
      The Acme Roadster is a sleek and powerful sports car that combines
      stunning design with exceptional performance. With its
      turbocharged engine, advanced suspension, and responsive handling,
      this car is a true driver's delight.
    </p>
    <div className="row mb-3">
      <div className="col-6">
        <p className="text-muted fw-light my-1">Mileage</p>
        <h6 className="">25 mpg city / 32 mpg highway</h6>
      </div>
      <div className="col-6">
        <p className="text-muted fw-light my-1">Engine</p>
        <h6>2.0L Turbocharged 4-cylinder</h6>
      </div>
      <div className="col-6">
        <p className="text-muted fw-light my-1">Transmission</p>
        <h6>8-speed automatic</h6>
      </div>
      <div className="col-6">
        <p className="text-muted fw-light my-1">Drivetrain</p>
        <h6>Rear-wheel drive</h6>
      </div>
    </div>
    <div className="d-flex gap-3">
      <button className="btn btn-dark px-4">Smth</button>
      <button className="btn btn-secondary px-4">
        <Heart className="me-2" height={18} width={18} /> Вподобати
      </button>
    </div>
  </div>
</div>

```

```

<div className="px-3 my-4">
  <hr />
</div>
<div className="mb-5">
  <h3 className="fw-bold mt-5 mb-4">Вам також може сподобатися</h3>
  <CardsCarousel
    vehicles=[
      {
        id: 1,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
          'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1e4
66d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
      },
      {
        id: 2,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
          'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1e4
66d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
      },
      {
        id: 3,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
          'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
      },
      {
        id: 4,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
          'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
      },
      {
        id: 5,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						82
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        fuelType: 'Gas',
        imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
            e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
    },
    ]]
    />
</div>
</Container>
</Container>
);
}

```

Profile.tsx

```

import React, { useState } from 'react';
import { Container, Dropdown, Nav } from 'react-bootstrap';
import { Link, useNavigate } from 'react-router-dom';
import { useAppSelector } from '../hooks/redux';
import ItemCard from '../components/Cards/ItemCard';
import { Calendar, Heart, HeartFill } from 'react-bootstrap-icons';
import Favorite from '../components/Profile/Favorite';
import Users from '../components/Profile/Users';
import Vehicles from '../components/Profile/Vehicles';
import { useLogout } from '../hooks/useLogout';

export default function Profile() {
    const { user, isLoading } = useAppSelector(
        (state) => state.authorizationReducer
    );
    const { logoutHandler } = useLogout();
    const [activeTab, setActiveTab] = useState('favorites');
    const navigate = useNavigate();
    if (isLoading) return null;
    if (user === null) {
        navigate('/login');
        return null;
    }

    return (
        <Container fluid>
            <Container>
                <div className="row justify-content-between mt-5">
                    <div className="col-3 bg-light shadow-lg rounded px-4 py-3">
                        <div className="row align-items-center">
                            <div className="col-3 h-100 d-flex justify-content-center align-items-center">
                                

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						83
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        width={64}
        height={64}
        alt=""
      />
    </div>
    <div className="col-7">
      <input
        type="text"
        defaultValue={` ${user.name} || 'Unknow'} ${user.surname} ??
'Unknow'}` ` }
        className="border-0 bg-light fw-bold w-100"
      />
      <p className="my-0">{user.email}</p>
    </div>
  </div>
  <div className="d-flex justify-content-between mt-4 pe-3">
    <button className="btn btn-dark px-4">Змінити пароль</button>
    <button
      onClick={logoutHandler}
      className="btn btn-secondary px-4"
    >
      Вихід
    </button>
  </div>
  <hr />
  <h5 className="fw-bold mt-3">Біографія</h5>
  <textarea
    defaultValue={
      "This is a placeholder for the user's bio or additional
information."
    }
    onChange={(e) => {
      e.currentTarget.style.height = 'auto';
      e.currentTarget.style.height = ${e.currentTarget.scrollHeight}px;
    }}
    style={{ resize: 'none' }}
    className="border-0 bg-light fw-light w-100 text-muted"
  />
</div>
<div className="col-9">
  <div className="ms-3 bg-light shadow-lg rounded px-4 py-4">
    {user!.role === 'Admin' ? (
      <>
        <Dropdown>
          <Dropdown.Toggle
            className="bg-light text-black fw-bold fs-5 border-0 shadow-
none px-0"
            variant="success"
            id="dropdown-basic"
          >
            {activeTab === 'favorites' ? (

```

```

        'Вподобаний транспорт'
      ) : (
        <></>
      )}
      {activeTab === 'users' ? 'Список користувачів' : <></>}
      {activeTab === 'vehicles' ? 'Транспортні засоби' : <></>}
    </Dropdown.Toggle>
  <Dropdown.Menu>
    <Dropdown.Item
      onClick={() => setActiveTab('favorites')}
      active={activeTab === 'favorite'}
    >
      Вподобаний транспорт
    </Dropdown.Item>
    <Dropdown.Item
      onClick={() => setActiveTab('users')}
      active={activeTab === 'users'}
    >
      Список користувачів
    </Dropdown.Item>
    <Dropdown.Item
      onClick={() => setActiveTab('vehicles')}
      active={activeTab === 'vehicles'}
    >
      Транспортні засоби
    </Dropdown.Item>
  </Dropdown.Menu>
</Dropdown>
</>
) : (
  <>
    <h5 className="fw-bold ">Вподобаний транспорт</h5>
  </>
)
{activeTab === 'favorites' ? <Favorite /> : <></>}
{activeTab === 'users' ? <Users /> : <></>}
{activeTab === 'vehicles' ? <Vehicles /> : <></>}
</div>
</div>
</div>
</Container>
</Container>
);
}

```

Login.tsx

```

import React from 'react';
import { Container } from 'react-bootstrap';
import { useForm } from 'react-hook-form';
import { Link, useNavigate } from 'react-router-dom';
import setTitle from '../utils/setTitle';

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { useAppDispatch } from '../../../hooks/redux';
import { login } from '../../../redux/actions/authorization/loginAction';

interface ILoginForm {
  email: string;
  password: string;
}

export default function Login() {
  setTitle('Авторизація');

  const navigate = useNavigate();
  const dispatch = useAppDispatch();
  const skipAuth = () => navigate('/');

  const {
    register,
    formState: { errors },
    handleSubmit,
  } = useForm<ILoginForm>();

  const loginHandler = (data: ILoginForm) => {
    dispatch(login(data, () => navigate('/')));
  };

  return (
    <div className={ 'container-fluid' }>
      <div className="row mt-5">
        <div className="col-md-5 col-lg-4 m-auto">
          <div className="my-5 ">
            <div className="w-100 py-3 px-5">
              <h3 className="text-center align-items-center mb-1 fw-bold">
                Авторизація
              </h3>
              <p className="text-center align-items-center my-0 text-muted">
                Введіть адресу електронної пошти та пароль для доступу до свого
                облікового запису
              </p>
            </div>
            <form onSubmit={handleSubmit(loginHandler)}>
              <div className="mb-3 px-5 mt-2">
                <label htmlFor="username" className="form-label fs-6 fw-light">
                  Пошта
                </label>
                <input
                  type="text"
                  className="form-control"
                  placeholder={ 'm@example.com' }
                  {...register('email', {
                    required: true,
                  })}
                >
              </div>
            </form>
          </div>
        </div>
      </div>
    </div>
  );
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    />
    <span className={'text-danger fw-light mt-2 opacity-100'}>
      {errors?.email && `Не правильний логін або пароль`}
    </span>
  </div>
  <div className="mb-3 px-5">
    <label htmlFor="password" className="form-label fs-6 fw-light">
      Пароль
    </label>
    <input
      type="password"
      className="form-control"
      {...register('password', {
        required: true,
      })}
    />
    <span className={'text-danger fw-light mt-2 opacity-100'}>
      {errors?.password && `Не правильний логін або пароль`}
    </span>
  </div>
  <div className="px-5 mb-3 padding-bottom text-center text-sm-start">
    <button className={'btn btn-dark w-100 mb-3'} type={'submit'}>
      Увійти
    </button>
    <button
      onClick={skipAuth}
      className={'btn btn-secondary w-100'}
    >
      Продовжити без авторизації
    </button>
  </div>
  <p className={'px-5 text-center'}>
    Не маєте аккаунту?{' '}
    <Link className={'text-dark fw-normal'} to={'/signup'}>
      Зареєструйся
    </Link>
  </p>
</form>
</div>
</div>
</div>
);
}

```

CreateVehicleForm.tsx

```

import React from 'react';
import setTitle from '../utils/setTitle';
import { useAppDispatch } from '../hooks/redux';
import { useNavigate } from 'react-router-dom';
import { useForm } from 'react-hook-form';

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import '@patternfly/react-core/dist/styles/base.css';

import {
  DropEvent,
  HelperText,
  HelperTextItem,
  MultipleFileUpload,
  MultipleFileUploadMain,
  MultipleFileUploadStatus,
  MultipleFileUploadStatusItem,
} from '@patternfly/react-core';
import { Upload } from 'react-bootstrap-icons';

interface readFile {
  fileName: string;
  data?: string;
  loadResult?: 'danger' | 'success';
  loadError?: DOMException;
}

interface IFormInput {
  brand: string;
  model: string;
  year: string;
  body_type: string;
  fuel_type: string;
  engine_volume: string;
  power?: string;
  cylinders: string;
  transmission: string;
  gears: string;
  occasion: string;
  lenght: string;
  width: string;
  height: string;
  weight: string;
  tank_volume: string;
  description: string;
  max_speed: string;
  price: string;
}

export default function CreateVehicleForm() {
  setTitle('Новий транспорт');
  const dispatch = useAppDispatch();
  const navigate = useNavigate();
  const [currentFiles, setCurrentFiles] = React.useState<File[]>([]);
  const [readFileData, setReadFileData] = React.useState<readFile[]>([]);
  const [showStatus, setShowStatus] = React.useState(false);
  const [statusIcon, setStatusIcon] = React.useState('inProgress');

  const {

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    register,
    watch,
    formState: { errors },
    handleSubmit,
  } = useForm<IFormInput>();

  const convertFileToBase64 = (file: File) => {
    return new Promise<string>((resolve, reject) => {
      const reader = new FileReader();
      reader.onload = () => resolve(reader.result as string);
      reader.onerror = (error) => reject(error);
      reader.readAsDataURL(file);
    });
  };

  const addVehicle = async (data?: IFormInput) => {
    try {
      const filesBase64Promises = currentFiles.map((file) =>
        convertFileToBase64(file)
      );

      const filesBase64Data = await Promise.all(filesBase64Promises);

      const vehicleData = {
        ...data,
        images: filesBase64Data.map((base64, index) => ({
          fileName: currentFiles[index].name,
          data: base64,
        })),
      };

      console.log(vehicleData); // You can dispatch the vehicleData to your Redux
store or API here
      // dispatch(sign_up(vehicleData, () => navigate('/')));
    } catch (error) {
      console.error('Error converting files to Base64:', error);
    }
  };

  if (!showStatus && currentFiles.length > 0) {
    setShowStatus(true);
  }

  React.useEffect(() => {
    if (readFileData.length < currentFiles.length) {
      setStatusIcon('inProgress');
    } else if (readFileData.every((file) => file.loadResult === 'success')) {
      setStatusIcon('success');
    } else {
      setStatusIcon('danger');
    }
  });

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						89
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }, [readFileData, currentFiles]));

const removeFiles = (namesOfFilesToRemove: string[]) => {
    const newCurrentFiles = currentFiles.filter(
        (currentFile) =>
            !namesOfFilesToRemove.some((fileName) => fileName === currentFile.name)
    );

    setCurrentFiles(newCurrentFiles);

    const newReadFiles = readFileData.filter(
        (readFile) =>
            !namesOfFilesToRemove.some((fileName) => fileName === readFile.fileName)
    );

    setReadFileData(newReadFiles);
};

const updateCurrentFiles = (files: File[]) => {
    setCurrentFiles((prevFiles) => [...prevFiles, ...files]);
};

const handleFileDrop = (_event: DropEvent, droppedFiles: File[]) => {
    const currentFileNames = currentFiles.map((file) => file.name);
    const reUploads = droppedFiles.filter((droppedFile) =>
        currentFileNames.includes(droppedFile.name)
    );

    Promise.resolve()
        .then(() => removeFiles(reUploads.map((file) => file.name)))
        .then(() => updateCurrentFiles(droppedFiles));
};

const handleReadSuccess = (data: string, file: File) => {
    setReadFileData((prevReadFiles) => [
        ...prevReadFiles,
        { data, fileName: file.name, loadResult: 'success' },
    ]);
};

const handleReadFail = (error: DOMException, file: File) => {
    setReadFileData((prevReadFiles) => [
        ...prevReadFiles,
        { loadError: error, fileName: file.name, loadResult: 'danger' },
    ]);
};

const createHelperText = (file: File) => {
    const fileResult = readFileData.find(
        (readFile) => readFile.fileName === file.name
    );
    if (fileResult?.loadError) {

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

```

return (
  <HelperText isLiveRegion>
    <HelperTextItem variant={'error'}>
      {fileResult.loadError.toString()}
    </HelperTextItem>
  </HelperText>
);
}
};

const successfullyReadFileCount = readFileData.filter(
  (fileData) => fileData.loadResult === 'success'
).length;

return (
  <div className={'container-fluid'}>
    <div className="row mt-3">
      <div className="col-md-5 col-lg-5 m-auto">
        <div className="my-5 ">
          <div className="w-100 py-3 px-5">
            <h3 className="text-center align-items-center mb-1 fw-bold">
              Додавання транспорту
            </h3>
            <p className="text-center align-items-center my-0 text-muted">
              Введіть марку, модель, рік випуску та пробіг транспортного
              засобу, щоб додати новий автомобіль.
            </p>
          </div>
        </div>
        <form onSubmit={handleSubmit(addVehicle)}>
          <div className="mb-3 px-5 mt-2">
            <label htmlFor="string" className="form-label fs-6 fw-light">
              Бренд
            </label>
            <input
              type="text"
              className="form-control"
              placeholder="BMW"
              {...register('brand', {
                required: "Це поле обов'язкове!",
              })}
            />
            <span className="text-danger mt-1 opacity-100">
              {errors?.brand && errors.brand.message}
            </span>
          </div>
          <div className="mb-3 px-5 mt-2">
            <label htmlFor="string" className="form-label fs-6 fw-light">
              Модель
            </label>
            <input
              type="text"

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        className="form-control"
        placeholder="M5"
        {...register('model', {
            required: "Це поле обов'язкове!",
        })}
    />
    <span className="text-danger mt-1 opacity-100">
        {errors?.model && errors.model.message}
    </span>
</div>
<div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
        Рік випуску
    </label>
    <input
        type="text"
        className="form-control"
        placeholder="1999"
        {...register('year', {
            required: "Це поле обов'язкове!",
        })}
    />
    <span className="text-danger mt-1 opacity-100">
        {errors?.year && errors.year.message}
    </span>
</div>
<div className="px-5 mt-2">
    <label className="form-label fs-6 fw-light">Фото</label>
    <MultipleFileUpload
        onFileDrop={handleFileDrop}
        dropzoneProps={{
            accept: {
                'image/jpeg': ['.jpg', '.jpeg'],
                'image/png': ['.png'],
            },
        }}
    >
    <MultipleFileUploadMain
        className="fw-bold"
        browseButtonText="Огляд"
        titleIcon={<Upload height={20} width={20} />}
        titleText="Перетягніть файли сюди"
        titleTextSeparator="або"
        infoText="Прийнятні типи файлів: JPEG, PNG"
    />
    {showStatus && (
        <MultipleFileUploadStatus
            statusToggleText={` ${successfullyReadFileCount} з
${currentFiles.length} фото завантажено`}
            statusToggleIcon={statusIcon}
        >

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        {currentFiles.map((file) => (
            <MultipleFileUploadStatusItem
                file={file}
                key={file.name}
                onClearClick={() => removeFiles([file.name])}
                onReadSuccess={handleReadSuccess}
                onReadFail={handleReadFail}
                progressHelperText={createHelperText(file)}
            />
        ))}
    </MultipleFileUploadStatus>
  ))
</MultipleFileUpload>
</div>
<div className="mb-3 px-5 mt-2">
  <label htmlFor="string" className="form-label fs-6 fw-light">
    Опис
  </label>
  <textarea
    className="form-control"
    style={{ resize: 'none', minHeight: '150px' }}
    placeholder={
      'BMW M5 – це елегантний і потужний спортивний автомобіль, який
      поєднує в собі приголомшливий дизайн із винятковою продуктивністю. Завдяки двигуну
      з турбонаддувом, вдосконаленій підвісці та чуйній керованості цей автомобіль
      справжнє задоволення для водія.'
    }
  >
    {...register('description', {
      required: "Це поле обов'язкове!",
      onChange: (e) => {
        e.currentTarget.style.height = 'auto';
        e.currentTarget.style.height =
        ${e.currentTarget.scrollHeight}px;
      },
    })}
  </textarea>
  <span className="text-danger mt-1 opacity-100">
    {errors?.description && errors.description.message}
  </span>
</div><div className="mb-3 px-5 mt-2">
  <label htmlFor="string" className="form-label fs-6 fw-light">
    Тип кузова
  </label>
  <select
    {...register('body_type', {
      required: "Це поле обов'язкове!",
    })}
    className="form-select"
  >
    <option value="sedan">Седан</option>
    <option value="hatchback">Хетчбек</option>
  </select>
</div>

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		93

```

<option value="universal">Універсал</option>
<option value="crossover">Кросовер</option>
<option value="suv">Позашляховик (SUV)</option>
<option value="coupe">Купе</option>
<option value="convertible">Кабріолет</option>
<option value="roadster">Родстер</option>
<option value="minivan">Мінівен</option>
<option value="van">Фургон (Вен)</option>
<option value="pickup">Пікап</option>
<option value="liftback">Ліфтбек</option>
<option value="targa">Тарга</option>
<option value="shooting-brake">Шутинг-брейк</option>
<option value="fastback">Фастбек</option>
<option value="coupe-crossover">Купе-кросовер</option>
<option value="armored">Броньований автомобіль</option>
</select>

<span className="text-danger mt-1 opacity-100">
  {errors?.body_type && errors.body_type.message}
</span>
</div>

<div className="mb-3 px-5 mt-2">
  <label htmlFor="string" className="form-label fs-6 fw-light">
    Тип палива
  </label>
  <input
    type="text"
    className="form-control"
    placeholder="Бензин"
    {...register('fuel_type', {
      required: "Це поле обов'язкове!",
    })}
  />
  <span className="text-danger mt-1 opacity-100">
    {errors?.fuel_type && errors.fuel_type.message}
  </span>
</div>

<div className="mb-3 px-5 mt-2">
  <label htmlFor="string" className="form-label fs-6 fw-light">
    Об'єм двигуна
  </label>
  <input
    type="text"
    className="form-control"
    placeholder="3000 см³"
    {...register('engine_volume')}
  />
  <span className="text-danger mt-1 opacity-100">
    {errors?.engine_volume && errors.engine_volume.message}
  </span>
</div>

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    </span>
  </div>

  <div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
      Потужність двигуна
    </label>
    <input
      type="text"
      className="form-control"
      placeholder="250 к.с."
      {...register('power')}
    />
  </div>

  <div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
      Кількість циліндрів
    </label>
    <input
      type="text"
      className="form-control"
      placeholder="6"
      {...register('cylinders')}
    />
    <span className="text-danger mt-1 opacity-100">
      {errors?.cylinders && errors.cylinders.message}
    </span>
  </div><div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
      Трансмісія
    </label>
    <input
      type="text"
      className="form-control"
      placeholder="Автоматична"
      {...register('transmission', {
        required: "Це поле обов'язкове!",
      })}
    />
    <span className="text-danger mt-1 opacity-100">
      {errors?.transmission && errors.transmission.message}
    </span>
  </div>

  <div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
      Кількість передач
    </label>
    <input
      type="text"

```

```

        className="form-control"
        placeholder="8"
        {...register('gears')}}
    />
    <span className="text-danger mt-1 opacity-100">
        {errors?.gears && errors.gears.message}
    </span>
</div>

<div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
        Пробір
    </label>
    <input
        type="text"
        className="form-control"
        placeholder="100000 км"
        {...register('occasion', {
            required: "Це поле обов'язкове!",
        })}
    />
    <span className="text-danger mt-1 opacity-100">
        {errors?.occasion && errors.occasion.message}
    </span>
</div>

<div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
        Довжина
    </label>
    <input
        type="text"
        className="form-control"
        placeholder="5000 мм"
        {...register('length')}
    />
    <span className="text-danger mt-1 opacity-100">
        {errors?.length && errors.length.message}
    </span>
</div>

<div className="mb-3 px-5 mt-2">
    <label htmlFor="string" className="form-label fs-6 fw-light">
        Ширина
    </label>
    <input
        type="text"
        className="form-control"
        placeholder="2000 мм"
        {...register('width')}
    />

```

```

        <span className="text-danger mt-1 opacity-100">
            {errors?.width && errors.width.message}
        </span>
    </div>

    <div className="mb-3 px-5 mt-2">
        <label htmlFor="string" className="form-label fs-6 fw-light">
            Висота
        </label>
        <input
            type="text"
            className="form-control"
            placeholder="1500 мм"
            {...register('height')}
        />
        <span className="text-danger mt-1 opacity-100">
            {errors?.height && errors.height.message}
        </span>
    </div>

    <div className="mb-3 px-5 mt-2">
        <label htmlFor="string" className="form-label fs-6 fw-light">
            Бара
        </label>
        <input
            type="text"
            className="form-control"
            placeholder="1800 кг"
            {...register('weight')}
        />
        <span className="text-danger mt-1 opacity-100">
            {errors?.weight && errors.weight.message}
        </span>
    </div><div className="mb-3 px-5 mt-2">
        <label htmlFor="string" className="form-label fs-6 fw-light">
            Об'єм паливного бака
        </label>
        <input
            type="text"
            className="form-control"
            placeholder="60 л"
            {...register('tank_volume')}
        />
        <span className="text-danger mt-1 opacity-100">
            {errors?.tank_volume && errors.tank_volume.message}
        </span>
    </div>

    <div className="mb-3 px-5 mt-2">
        <label htmlFor="string" className="form-label fs-6 fw-light">
            Максимальна швидкість

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						97
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </label>
        <input
          type="text"
          className="form-control"
          placeholder="250 км/год"
          {...register('max_speed')}
        />
      </div>

      <div className="mb-3 px-5 mt-2">
        <label htmlFor="string" className="form-label fs-6 fw-light">
          Ціна
        </label>
        <input
          type="text"
          className="form-control"
          placeholder="1 000 000 грн"
          {...register('price', {
            required: "Це поле обов'язкове!",
          })}
        />
        <span className="text-danger mt-1 opacity-100">
          {errors?.price && errors.price.message}
        </span>
      </div>

      <div className="px-5 mt-3 mb-3 padding-bottom text-center text-sm-
start">
        <button
          onClick={() => addVehicle()}
          className={'btn btn-dark w-100 mb-3'}
          type={'submit'}
        >
          Додати
        </button>
      </div>
    </form>
  </div>
</div>
</div>
</div>
</div>
);
}

```

redux.ts

```

import { TypedUseSelectorHook, useDispatch, useSelector } from 'react-redux';
import { AppDispatch, RootState } from '../redux/index';

export const useAppDispatch = () => useDispatch<AppDispatch>();
export const useAppSelector: TypedUseSelectorHook<RootState> = useSelector;

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						98
Змн.	Арк.	№ докум.	Підпис	Дата		

useLogout.tsx

```
import React from 'react';
import { useDispatch } from '../redux';
import { logout } from '../redux/actions/authorization/logoutAction';

export const useLogout = () => {
  const dispatch = useDispatch();
  const logoutHandler = () => {
    dispatch(logout());
  };
  return { logoutHandler };
};
```

Search.tsx

```
import React from 'react';
import CardsCarousel from '../Cards/CardsCarousel';
interface ISearch {
  showResults?: boolean;
}

export default function Search({ showResults = false }: ISearch) {
  return (
    <div className="bg-light pt-2 pb-4 px-4 rounded border border-1 shadow">
      <h4 className="fw-bold mt-2 mb-4">Знайдіть своє ідеальне авто</h4>
      <div className="row">
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Тип кузова
          </label>
          <select className="form-select" defaultValue="">
            <option></option>
            <option>Седан</option>
          </select>
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Бренд
          </label>
          <select className="form-select" defaultValue="">
            <option></option>
            <option>Седан</option>
          </select>
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Модель
          </label>
          <input
            placeholder="Введіть модель"
            className="form-control"
            type="text"
          />
        </div>
      </div>
    </div>
```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						99
Змн.	Арк.	№ докум.	Підпис	Дата		

```

</div>
<div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
  <label htmlFor="username" className="form-label fs-6 fw-light">
    Ціна
  </label>
  <div className="row">
    <div className="col">
      <input className="form-control" defaultValue={0} type="text" />
    </div>
    <div className="col">
      <input
        className="form-control"
        defaultValue={100000}
        type="text"
      />
    </div>
  </div>
</div>
<div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
  <label htmlFor="username" className="form-label fs-6 fw-light">
    Тип палива
  </label>
  <select className="form-select" defaultValue={''}>
    <option></option>
    <option>Бензин</option>
    <option>Дизель</option>
    <option>Електро</option>
    <option>Гибрид</option>
  </select>
</div>
</div>
<div className="w-100 mb-4">
  <button className="btn btn-dark w-100 py-3 py-sm-2">Пошук</button>
</div>
{showResults ? (
  <>
    <h4 className="fw-bold mb-4">Результати пошуку</h4>
    <CardsCarousel
      vehicles=[
        {
          id: 1,
          title: 'CHEVROLET TRAX 2LT',
          price: '1200',
          fuelType: 'Gas',
          imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
            e466d85816c0722b70dfd_ful.jpg',
          transmission: 'Automatic',
        },
        {
          id: 2,

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						100
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
    },
    {
        id: 3,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
    },
    {
        id: 4,
        title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
    },
    {
        id: 5, title: 'CHEVROLET TRAX 2LT',
        price: '1200',
        fuelType: 'Gas',
        imgUrl:
            'https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0624/046d4c87ad1
e466d85816c0722b70dfd_ful.jpg',
        transmission: 'Automatic',
    },
    ]}
    />
</>
) : (
    <></>
)}
</div>
);
}

```

Vehicles.tsx

```

import React from 'react';
import { Eye, Plus } from 'react-bootstrap-icons';
import { Link } from 'react-router-dom';

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						101
Змн.	Арк.	№ докум.	Підпис	Дата		

```

export default function Vehicles() {
  return (
    <>
      <div className="row mt-2">
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Тип кузова
          </label>
          <select className="form-select" defaultValue={''}>
            <option></option>
            <option>Седан</option>
          </select>
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Бренд
          </label>
          <select className="form-select" defaultValue={''}>
            <option></option>
            <option>Седан</option>
          </select>
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Модель
          </label>
          <input
            placeholder="Введіть модель"
            className="form-control"
            type="text"
          />
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Тип палива
          </label>
          <select className="form-select" defaultValue={''}>
            <option></option>
            <option>Бензин</option>
            <option>Дизель</option>
            <option>Електро</option>
            <option>Гибрид</option>
          </select>
        </div>
        <div className="col-12 col-sm-6 col-md-4 col-lg-3 mb-3 mt-2">
          <label htmlFor="username" className="form-label fs-6 fw-light">
            Ціна
          </label>
          <div className="row">
            <div className="col">

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <input className="form-control" defaultValue={0} type="text" />
      </div>
      <div className="col">
        <input
          className="form-control"
          defaultValue={100000}
          type="text"
        />
      </div>
    </div>
  </div>
  <div className="d-flex justify-content-end">
    <Link to={"/vehicles/new"} className="btn btn-dark">
      Додати транспорт <Plus width={21} height={21} />
    </Link>
  </div>
  <table className="table shadow-sm mt-3">
    <thead>
      <tr>
        <th>#</th>
        <th>Тип кузова</th>
        <th>Бренд</th>
        <th>Модель</th>
        <th>Ціна</th>
        <th>Тип палива</th>
        <th>Управління</th>
      </tr>
    </thead>
    <tbody>
      { /* {users.map((item, index) => (
        <tr className="align-middle">
          <td>{index + 1}</td>
          <td>{item.name}</td>
          <td>{item.surname}</td>
          <td>{item.email}</td>
          <td>{item.role}</td>
          <td>
            <button className="btn btn-secondary px-2 py-1">
              <Eye className="mb-1" />
            </button>
          </td>
        </tr>
      )} */ }
    </tbody>
  </table>
</>
);
}

```

Users.tsx

```

import React, { useEffect } from 'react';
import { useAppDispatch, useAppSelector } from '../../hooks/redux';
import { getUsers } from '../../redux/actions/users/getUsersAction';
import { Eye, Search } from 'react-bootstrap-icons';

export default function Users() {
  const dispatch = useAppDispatch();
  const { isLoading, users } = useAppSelector((state) => state.usersReducer);

  useEffect(() => {
    dispatch(getUsers());
  }, []);
  if (isLoading) return null;
  return (
    <>
      <div className="row mt-2">
        <div className="col-4 d-flex">
          <div className="input-group">
            <input
              placeholder="m@example.com"
              className="form-control"
              type="text"
            />
            <button className="btn btn-secondary">
              <Search className="mb-1" />
            </button>
          </div>
        </div>
        <div className="col-4 d-flex align-items-center">
          <select className="form-select">
            <option value="">Всі</option>
            <option value="">Користувач</option>
            <option value="">Адмін</option>
          </select>
        </div>
      </div>
      <table className="table shadow-sm mt-3">
        <thead>
          <tr>
            <th>#</th>
            <th>Ім'я</th>
            <th>Прізвище</th>
            <th>Пошта</th>
            <th>Роль</th>
            <th>Управління</th>
          </tr>
        </thead>
        <tbody>
          {users.map((item, index) => (
            <tr className="align-middle">
              <td>{index + 1}</td>

```

```

        <td>{item.name}</td>
        <td>{item.surname}</td>
        <td>{item.email}</td>
        <td>{item.role}</td>
        <td>
            <button className="btn btn-secondary px-2 py-1">
                <Eye className="mb-1" />
            </button>
        </td>
    </tr>
    )})
</tbody>
</table>
</>
);
}

```

Favorite.tsx

```

import React from 'react';
import ItemCard from '../Cards/ItemCard';

export default function Favorite() {
    return (
        <div className="row justify-content-between mt-4 gap-4">
            <ItemCard
                className="col-6"
                fuel_type="Бензин"
                price="37 889,66"
                imageUrl="https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/ab5726ea058b4753b244310392150b57_ful.jpg"
                title="BMW 530 XI"
                transmission="Автоматична"
            />
            <ItemCard
                className="col-6"
                fuel_type="Бензин"
                price="37 889,66"
                imageUrl="https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/ab5726ea058b4753b244310392150b57_ful.jpg"
                title="BMW 530 XI"
                transmission="Автоматична"
            />
            <ItemCard
                className="col-6"
                fuel_type="Бензин"
                price="37 889,66"
                imageUrl="https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/ab5726ea058b4753b244310392150b57_ful.jpg"
                title="BMW 530 XI"
                transmission="Автоматична"
            />
        </div>
    );
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<ItemCard
  className="col-6"
  fuel_type="Бензин"
  price="37 889,66"
  imageUrl="https://cs.copart.com/v1/AUTH_svc.pdoc00001/lpp/0524/ab5726ea058b
4753b244310392150b57_ful.jpg"
  title="BMW 530 XI"
  transmission="Автоматична"
/>
</div>
);
}

```

Transmission.tsx

```

import React from 'react';
import { IIIconProps } from '../types/Icon';

export default function Transmission({
  width = 18,
  height = 18,
  className,
}: IIIconProps) {
  return (
    <svg
      version="1.1"
      xmlns="http://www.w3.org/2000/svg"
      width={width}
      height={height}
      className={className}
      viewBox="0 0 32 32"
    >
      <title>transmission-2</title>
      <path
        fill="#444446"
        d="M7.704 5.333c-1.309 0-2.37 1.061-2.37 2.37 0.001 0.846 0.452 1.628 1.185
2.051v4.861c-0.021 0.128-0.021 0.259 0 0.387v4.875c-0.732 0.423-1.184 1.203-1.185
2.049 0 1.309 1.061 2.37 2.37 2.37s2.37-1.061 2.37-2.37c-0.001-0.846-0.452-1.628-
1.185-2.051v-3.875h5.926v3.875c-0.733 0.423-1.185 1.205-1.185 2.051 0 1.309 1.061
2.37 2.37 2.37s2.37-1.061 2.37-2.37c-0.002-0.845-0.453-1.626-1.185-2.049v-
3.877h5.926v6.248c-0.732 0.423-1.184 1.203-1.185 2.049 0 1.309 1.061 2.37 2.37
2.37s2.37-1.061 2.37-2.37c-0.001-0.846-0.452-1.628-1.185-2.051v-7.231c0.021-0.128
0.021-0.259 0-0.387v-4.875c0.732-0.423 1.184-1.203 1.185-2.049 0-1.309-1.061-2.371-
2.37-2.371s-2.37 1.061-2.37c0.001 0.846 0.452 1.628 1.185 2.051v3.875h-5.926v-
3.877c0.732-0.423 1.184-1.203 1.185-2.049 0-1.309-1.061-2.37-2.37-2.37s-2.37 1.061-
2.37 2.37c0.001 0.846 0.452 1.628 1.185 2.051v3.875h-5.926v-3.877c0.732-0.423
1.184-1.203 1.185-2.049 0-1.309-1.061-2.37-2.37-2.37z"
      ></path>
    </svg>
  );
}

```

ItemCard.tsx

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						106
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import React from 'react';
import { Button, Card } from 'react-bootstrap';
import { FuelPump, Heart } from 'react-bootstrap-icons';
import Transmission from '../Icons/Transmission';
import { Link } from 'react-router-dom';

interface IItemCardProps {
  className?: string;
  imageUrl?: string;
  title?: string;
  price?: string;
  fuel_type?: string;
  transmission?: string;
}

export default function ItemCard(props: IItemCardProps) {
  return (
    <Card
      className={`rounded-3 border-1 ${props.className}`}
      style={{ width: '18rem' }}
    >
      <div className="p-3">
        <img className="card-img rounded" src={props.imageUrl} alt="" />
      </div>
      <Card.Body className="px-3">
        <div className="d-flex flex-row justify-content-between align-items-center flex-grow-1">
          <h6 className="fw-bold my-0">{props.title}</h6>
          <p className="fw-bold my-0">${props.price}</p>
        </div>
        <div>
          <p>
            <FuelPump className="mb-1" /> {props.fuel_type} |{' '}
            <Transmission className="" width={20} height={20} />
            <span>{props.transmission}</span>
          </p>
        </div>
        <div className="d-flex flex-row justify-content-between flex-grow-1">
          <Link to={`/vehicle`} className="btn btn-secondary">
            Детальніше
          </Link>
          <button className="btn btn-sm shadow-none border-0">
            <Heart height={20} width={20} />
          </button>
        </div>
      </Card.Body>
    </Card>
  );
}

```

CardCarousel.tsx

```
import React from 'react';
```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

import { responsive1 } from '../utils/config';
import ItemCard from './ItemCard';
import Carousel from 'react-multi-carousel';
import { IVehicle } from '../types/Vehicle';

interface ICardsCarousel {
  vehicles: IVehicle[];
}

export default function CardsCarousel({ vehicles }: ICardsCarousel) {
  return (
    <Carousel responsive={responsive1}>
      {vehicles.map((item) => (
        <ItemCard
          fuel_type={item.fuelType}
          imageUrl={item.imageUrl}
          key={item.id}
          price={item.price}
          transmission={item.transmission}
          title={item.title}
          className="col-12 col-sm-6 col-md-4"
        />
      ))}
    </Carousel>
  );
}

```

axios/index.ts

```

import axios from 'axios';

const $api = axios.create({
  withCredentials: true,
  baseURL: process.env.REACT_APP_API_URL,
});

$api.interceptors.request.use((config: any) => {
  config.headers.Authorization = `Bearer ${localStorage.getItem('token')}`;
  return config;
});

export default $api;

```

styles.css

```

@import '~react-image-gallery/styles/css/image-gallery.css';

.image-gallery-right-nav svg {
  width: 32px !important;
}

.image-gallery-left-nav svg {
  width: 32px !important;
}

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						108
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.image-gallery-image {
  border-radius: 2%;
}

.parrent:hover button {
  display: inline !important;
}

.hoverpointer:hover {
  cursor: pointer;
}

.headerr {
  background-color: #fff;
  border-bottom: 1px solid #eaeaea;
  position: fixed;
  width: 100%;
  top: 0;
  z-index: 1000;
  display: flex;
  justify-content: space-between;
  align-items: center;
  padding: 10px 20px;
}

.header-logo {
  font-size: 24px;
  font-weight: bold;
}

.main-nav {
  list-style: none;
  margin: 0;
  padding: 0;
  display: flex;
}

.main-nav li {
  margin-right: 20px;
}

.main-nav a {
  font-size: 16px;
  color: #333;
}

.filter-panel {
  background-color: #fff;
  padding: 20px;
  border: 1px solid #eaeaea;
  border-radius: 5px;
}

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						109
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    margin-bottom: 20px;
}

.filter-panel h2 {
    font-size: 18px;
    margin-bottom: 10px;
}

.filter-panel label {
    display: block;
    margin-bottom: 5px;
}

.filter-panel input,
.filter-panel select {
    width: 100%;
    padding: 10px;
    margin-bottom: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}

.car-card {
    background-color: #fff;
    border: 1px solid #eaeaea;
    border-radius: 5px;
    padding: 20px;
    margin-bottom: 20px;
    display: flex;
    flex-direction: column;
    align-items: center;
}

.car-card img {
    max-width: 100%;
    border-bottom: 1px solid #eaeaea;
    margin-bottom: 15px;
}

.car-card h3 {
    font-size: 20px;
    margin-bottom: 10px;
}

.car-card p {
    font-size: 16px;
    color: #666;
}

.car-card .price {
    font-size: 18px;

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

font-weight: bold;
color: #000;
margin-top: 10px;
}

@media (max-width: 768px) {
  .main-nav {
    flex-direction: column;
  }

  .main-nav li {
    margin-right: 0;
    margin-bottom: 10px;
  }

  .filter-panel {
    margin-bottom: 20px;
  }

  .car-card {
    flex-direction: column;
  }
}

```

index.html

```

<!doctype html>
<html lang="en">
  <head>
    <meta charset="utf-8" />
    <meta name="viewport" content="width=device-width, initial-scale=1" />
    <title>Car Finder</title>
    <link rel="icon" href="./car.ico" />
  </head>
  <body>
    <noscript>You need to enable JavaScript to run this app.</noscript>
    <div id="root"></div>
  </body>
</html>

```

Додаток Б

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						111
Змн.	Арк.	№ докум.	Підпис	Дата		

server/package.json

```
{
  "name": "server",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "dev": "nodemon index.js"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "dependencies": {
    "bcrypt": "^5.1.0",
    "cookie-parser": "^1.4.6",
    "cors": "^2.8.5",
    "dotenv": "^16.0.3",
    "express": "^4.18.2",
    "express-validator": "^7.0.1",
    "jsonwebtoken": "^9.0.0",
    "mongoose": "^5.5.0",
    "mongoose": "^7.2.0",
    "uuid": "^9.0.0"
  },
  "devDependencies": {
    "nodemon": "^2.0.22"
  }
}
```

server/index.js

```
require('dotenv').config()
const express = require("express")
const cors = require("cors")
const cookieParser = require("cookie-parser")
const app = express()
const mongoose = require('mongoose');
const PORT = process.env.PORT || 5000
const errorMiddleware = require("./middlewares/error-middleware")
const initService = require("./service/init-service")
const indexRouter = require('./routers')

// const init = async () => {
//   await initService.Init()
// }
// init()
app.use(express.json())
app.use(cookieParser())
app.use(cors({
  credentials: true,
  origin: process.env.CLIENT_URL
```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						112
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    )))
    app.use("/api", indexRouter)
    app.use(errorMiddleware)

    const start = async () => {
      try {
        await mongoose.connect(process.env.DB_URL, {
          useNewUrlParser: true,
          useUnifiedTopology: true,
        })
        app.listen(PORT, () => {
          console.log(`Server started:http://localhost:${PORT}`)
        })
      } catch (e) {
        console.log(e)
      }
    }
    start()

```

auth-services.js

```

const UserModel = require("../models/user-model");
const ApiError = require("../exceptions/api-errors");
const bcrypt = require("bcrypt");
const UserDto = require("../dtos/user-dto");
const tokenService = require("./token-service");

class AuthService{
  async login(email, password) {
    const user = await UserModel.findOne({email})
    console.log(user)
    if (!user)
      throw ApiError.BadRequest("Не правильний логін або пароль")

    const isPasswordEqual = await bcrypt.compare(password, user.password)
    if (!isPasswordEqual)
      throw ApiError.BadRequest("Не правильний логін або пароль")

    const userDto = new UserDto(user)
    const tokens = tokenService.generateTokens({...userDto})
    await tokenService.saveToken(userDto.id, tokens.refreshToken)

    return {
      ...tokens,
      user: userDto
    }
  }

  async logout(refreshToken) {
    const token = await tokenService.removeToken(refreshToken)
    return token
  }
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						113
Змн.	Арк.	№ докум.	Підпис	Дата		

```

async refresh(refreshToken){
    if(!refreshToken)
        throw ApiError.UnauthorizedError()

    const userData = tokenService.validateRefreshToken(refreshToken)
    const tokenFromDb = await tokenService.findToken(refreshToken)
    if (!userData || !tokenFromDb)
        throw ApiError.UnauthorizedError()

    const user = await UserModel.findById(userData.id)
    const userDto = new UserDto(user)
    const tokens = tokenService.generateTokens({...userDto})
    await tokenService.saveToken(userDto.id, tokens.refreshToken)
    return {
        ...tokens,
        user: userDto
    }
}

async updateSelf(req){
    const id = req.user.id
    const user = await UserModel.findOne({_id: id})
    const fields = ['email', 'phone', 'role']
    const reqData = req.body.data
    if(reqData["password"] && reqData["password2"] && reqData["oldPassword"]){
        console.log("Паролі є")
        if (reqData["password"] !== reqData["password2"]){
            console.log("Паролі не сходяться")
            throw ApiError.BadRequest("Паролі не сходяться")
        }else{
            console.log("Паролі сходяться")
            if (reqData['oldPassword']) {
                const isPasswordEqual = await
bcrypt.compare(reqData['oldPassword'], user.password)
                if (!isPasswordEqual) {
                    throw ApiError.BadRequest("Не правильний пароль")
                } else {
                    const hashPassword = await bcrypt.hash(reqData["password"],
3)
                    user['password'] = hashPassword
                }
            }
        }
    }
    fields.forEach(item => {
        if (reqData[item])
            user[item] = reqData[item]
    })
    await user.save()
    return user
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

module.exports = new AuthService()
                                auth-router.js

const Router = require('express').Router
const authController = require("../controllers/auth-controller");
const authMiddleware = require("../middlewares/auth-middleware")
const router = new Router()

router.get('/me',authMiddleware, authController.getSelfInfo)
router.put('/me',authMiddleware, authController.updateSelfInfo)
router.post('/login', authController.login)
router.post("/signup", authController.signup);
router.post('/logout', authController.logout)
router.get('/refresh',authMiddleware, authController.refresh)

module.exports = router

```

vehicle-model.js

```

const {Schema, model} = require("mongoose")

const vehicleSchema = new Schema({

  brand: {type: String, require: true},
  model: {type: String, require: true},
  year: {type: Number, require: true},
  body_type: {type: String, require: true},
  fuel_type: {type: String, require: true},
  engine_volume: {type: String, require: false},
  power: {type: Date, require: false},
  cylinders: {type: Number, require: false},
  transmission: {type: String, require: true},
  gears: {type: Number, require: false},
  occasion: {type: String, require: true},
  lenght: {type: Number, require: false},
  width: {type: Number, require: false},
  height: {type: Number, require: false},
  weight: {type: Number, require: false},
  tank_volume: {type: String, require: false},
  description: {type: String, require: false},
  max_speed: {type: Number, require: false},
  price: {type: Number, require: true},
})

module.exports = model('Vehicle', vehicleSchema)
                                user-model.js

```

```

const {Schema, model} = require("mongoose")

```

```

const userSchema = new Schema({
  email: {type: String, require: true},
  password: {type: String, require: true},
  role: {type: String, require: true, default: "User"},
})

module.exports = model('User', userSchema)
                                auth-middleware.js

const ApiError = require("../exceptions/api-errors")
const tokenService = require("../service/token-service")
module.exports = function (req, res, next) {
  try {
    const authHeader = req.headers.authorization
    if (!authHeader)
      return next(ApiError.UnauthorizedError())

    const accessToken = authHeader.split(' ')[1]
    if (!accessToken)
      return next(ApiError.UnauthorizedError())

    const userData = tokenService.validateAccessToken(accessToken)
    if(!userData)
      return next(ApiError.UnauthorizedError())

    req.user = userData
    next()
  }catch (e) {
    next(ApiError.UnauthorizedError())
  }
}

```

error-middleware.js

```

const apiError = require('../exceptions/api-errors')
module.exports = function (err, req, res, next){
  console.log(err)
  if (err instanceof apiError){
    return res.status(err.status).json({message: err.message, errors:
err.errors})
  }
  return res.status(500).json({message: "Непередбаченна помилка"})
}\

```

api-error.js

```

module.exports = class ApiError extends Error {
  status
  errors

  constructor(status, message, errors = []) {
    super(message);
    this.status = status
  }
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						116
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        this.errors = errors
    }

    static UnauthorizedError(){
        return new ApiError(401, 'Користувач не авторизований')
    }

    static BadRequest(message, errors = []){
        return new ApiError(400, message, errors)
    }

    static Forbidden(){
        return new ApiError(403)
    }

    static MethodNotAllowed(){
        return new ApiError(405)
    }
}

```

user-dto.js

```

module.exports = class UserDto {
    id
    role
    email
    constructor(model) {
        this.id = model._id
        this.role = model.role
        this.email = model.email
    }
}

```

auth-controller.js

```

const authService = require("../service/auth-service");
const userService = require("../service/user-service");

class AuthController {
    async login(req, res, next) {
        try {
            const {email, password} = req.body
            const userData = await authService.login(email, password)
            res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 * 24 * 60
* 60 * 1000, httpOnly: true})
            return res.json(userData)
        } catch (e) {
            next(e)
        }
    }

    async signup(req, res, next) {
        try {

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const userData = await userService.createUser(req.body);
console.log(userData)
res.cookie("refreshToken", userData.refreshToken, {
  maxAge: 30 * 24 * 60 * 60 * 1000,
  httpOnly: true,
});

return res.json(userData);
} catch (error) {
  next(error);
}
}

async logout(req, res, next) {
  try {
    const {refreshToken} = req.cookies
    await authService.logout(refreshToken);
    res.clearCookie('refreshToken')
    return res.status(200).send()
  } catch (e) {
    next(e)
  }
}

async refresh(req, res, next) {
  try {
    const {refreshToken} = req.cookies
    console.log(req.user.email, refreshToken)
    const userData = await authService.refresh(refreshToken)
    res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 * 24 * 60
* 60 * 1000, httpOnly: true})
    return res.json(userData)
  } catch (e) {
    next(e)
  }
}

async getSelfInfo(req, res, next){
  try {
    const id = req.user.id
    const userData = await userService.getInfoById(id)
    return res.json(userData)
  } catch (e) {
    next()
  }
}

async updateSelfInfo(req, res, next){
  try {
    const userData = await authService.updateSelf(req)
    return res.json(userData)
  }
}

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        } catch (e) {
            next(e)
        }
    }
}

module.exports = new AuthController()
                                user-controller.js

const userService = require('../service/user-service')
const {validationResult} = require('express-validator')
const ApiError = require("../exceptions/api-errors")

class UserController {
    async createUser(req, res, next) {
        try {
            if (req.user.role !== 2)
                throw ApiError.Forbidden()

            const errors = validationResult(req)
            if (!errors.isEmpty()) {
                return next(ApiError.BadRequest("Помилка при створенні користувача", errors.array()))
            }
            const {login, password, role, email, phone, coach} = req.body
            const userData = await userService.createUser(login, password, role, email, phone, coach)
            res.cookie('refreshToken', userData.refreshToken, {maxAge: 30 * 24 * 60 * 60 * 1000, httpOnly: true})
            return res.json(userData)
        } catch (e) {
            next(e)
        }
    }

    async getAllUsers(req, res, next) {
        try {
            if (req.user.role !== "Admin")
                throw ApiError.Forbidden()

            const users = await userService.getAllUsers()
            return res.json(users)
        } catch (e) {
            next(e)
        }
    }

    async getUserById(req, res, next){
        try {
            const user = await userService.getInfoById(req.params.id)
            return res.json(user)
        }
    }
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						119
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }catch (e) {
        next(e)
    }
}

async updateUser(req, res, next) {
    try {
        if (req.user.role !== 2)
            throw ApiError.Forbidden()

        if (req.body.id === req.user.id)
            throw ApiError.MethodNotAllowed()

        const userData = await userService.updateUser(req)
        return res.json(userData)
    } catch (e) {
        next(e)
    }
}

async checkEmail(req, res, next){
    try {
        console.log("check email")
        const email = req.params.email

        const userData = await userService.checkEmailUnique(email)
        return res.json(userData)
    } catch (e) {
        next(e)
    }
}

async deleteUserById(req, res, next){
    try {
        if (req.user.role !== 2)
            throw ApiError.Forbidden()

        if (req.body.id === req.user.id)
            throw ApiError.MethodNotAllowed()

        const userData = await userService.deleteUserById(req.params.id)
    }catch (e) {
        next(e)
    }
}

async findByLogin(req, res, next){
    try {
        const candidate = req.params.login
        if (!candidate)
            throw ApiError.BadRequest("Ви не вказали логін")
    }
}

```

					ІСТ.КР.Б – 126 – 24 – ПЗ	Арк.
						120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        const userData = await userService.findByLogin(candidate)

        return res.json(userData)
    } catch (e) {
        next(e)
    }
}

module.exports = new UserController()

```

					ICT.KP.Б – 126 – 24 – ПЗ	Арк.
						121
Змн.	Арк.	№ докум.	Підпис	Дата		