

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Застосунок для продажу автомобілів»

Виконав студент групи ПЗ-21-3
ФЕДОРОВИЧ Дмитро Миколайович

Керівник роботи:
КУШНІР Надія Олександрівна

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення
Тетяна ВАКАЛЮК
«17» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ФЕДОРОВИЧ Дмитро Миколайович**

Керівник роботи: **КУШНІР Надія Олександрівна**

Тема роботи: **«Застосунок для продажу автомобілів»**,

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: об'єктом дослідження є процес створення зручного та простого застосунку для продажу автомобілів, який охоплює усі етапи від подачі оголошення до технічних основ застосунку: реєстрація, база даних, інтерфейс, перевірка оголошень, спілкування з адміністратором, а предметом дослідження є методи та інструменти розробки, що використовуються для побудови платформи, яка повинна бути не лише ефективною з технічної точки зору, але й зручною, безпечною та гнучкою у використанні.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Кушнір Н.О.	17.02.2025р.	17.02.2025р.
2	Кушнір Н.О.	17.02.2025р.	17.02.2025р.
3	Кушнір Н.О.	17.02.2025р.	17.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з застосунку для продажу автомобілів та пояснювальної записки. Пояснювальна записка до випускної роботи містить 78 сторінок, 21 ілюстрацію та 18 таблиць.

Виконано розробку застосунку для продажу автомобілів, який забезпечує користувачам зручну платформу для вибору, замовлення та придбання як нових, так і вживаних авто. Основна ідея полягає у створенні простого та інтуїтивно зрозумілого інтерфейсу, що дозволяє клієнтам переглядати актуальні пропозиції, фільтрувати автомобілі за різними параметрами (марка, модель, рік, ціна тощо).

У рамках проєкту було визначено ключові завдання розробки, проведено аналіз функціональних вимог до системи, а також розглянуто подібні сервіси, що вже існують на ринку. Було здійснено проектування бази даних, яка має зберігати інформацію про автомобілі та користувачів. Для реалізації застосунку обрано сучасний технологічний стек, включно з React.js для фронтенду та надійною базою даних MongoDB, що забезпечує високу швидкість і масштабованість сервісу.

КЛЮЧОВІ СЛОВА: ЗАСТОСУНОК, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, ПРОДАЖ АВТОМОБІЛІВ, REACT.JS, MONGODB.

					ІПЗ.КР.Б-121-25-ПЗ					
Змн.	Арк.	№ докум.	Підпис	Дата	Застосунок для продажу автомобілів. Пояснювальна записка			Літ.	Арк.	Аркушів
Розроб.		Федорович Д.М.								
Керівник		Кушнір Н.О.							3	78
								Житомирська політехніка, група ІПЗ-21-3		
Н. контр.		Тичина О.П.								
Зав. каф.		Вакалюк Т.А.								

ABSTRACT

The final bachelor's thesis consists of an application for car sales and an explanatory note. The explanatory note to the final thesis contains 78 pages, 21 illustrations and 18 tables.

An application for selling cars has been developed. An application for selling cars has been developed, providing users with a convenient platform for selecting, ordering and purchasing both new and used cars. The main idea is to create a simple and intuitive interface that allows customers to view current offers and filter cars by various parameters (make, model, year, price, etc.).

As part of the project, key development tasks were identified, functional requirements for the system were analysed, and similar services already available on the market were reviewed. A database was designed to store information about cars and users. A modern technology stack was selected for the implementation of the application, including React.js for the front end and a reliable MongoDB database, which ensures high speed and scalability of the service.

KEYWORDS: APPLICATION, CLIENT-SERVER ARCHITECTURE, CAR SALES, REACT.JS, MONGODB.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ.....	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	6
ВСТУП.....	7
РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту	11
1.3 Вибір архітектури застосунку для продажу автомобілів.....	18
1.4 Обґрунтування вибору інструментальних засобів	19
Висновки до першого розділу	21
РОЗДІЛ 2 ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ	22
2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи.....	22
2.2 Розробка бази даних системи.....	28
2.3 Проектування та реалізація алгоритмів роботи системи	32
2.4 Реалізація функціоналу застосунку для продажу автомобілів.....	34
Висновки до другого розділу.....	41
РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ ЗАСТОСУНКОМ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ	42
3.1 Порядок встановлення та налаштування параметрів системи	42
3.2 Структура інтерфейсу застосунку для продажу автомобілів	43
3.3 Тестування роботи застосунку	50
Висновки до третього розділу	53
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	56
ДОДАТКИ.....	58

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

ПЗ – програмне забезччення

БД – база даних

СУБД – система управління базами даних

CI – Continuous Integration

CD – Continuous Delivery

HTTP – Hypertext Transfer Protocol

TCP – Transmission Control Protocol

IP – Internet Protocol

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми пов'язана із тим що, у сучасному світі, де цифровізація охоплює практично всі сфери життя, ринок купівлі-продажу автомобілів також зазнає суттєвих змін. Все більше людей надають перевагу онлайн-сервісам, які дозволяють швидко знайти потрібне авто, порівняти пропозиції та оформити угоду – не виходячи з дому. Традиційні автосалони та оголошення в газетах поступово втрачають актуальність, поступаючись місцем інноваційним цифровим платформам.

Розробка застосунку для продажу автомобілів є особливо актуальною, зважаючи на зростаючий попит на дистанційне обслуговування, а також потребу у зручних інструментах, які спрощують як пошук авто, так і процес його продажу. Такий застосунок дозволяє не лише економити час користувача, а й забезпечує ширший вибір, прозорість інформації та можливість миттєвої комунікації між покупцем і продавцем.

Крім того, за допомогою інтеграції з базами даних, системами оцінювання та перевірки авто, можна значно підвищити рівень довіри користувачів до сервісу. Таким чином, розробка подібного застосунку не лише відповідає потребам ринку, а й сприяє розвитку цифрової економіки в цілому.

Для реалізації такого функціонального та зручного застосунку доцільно використовувати сучасні технології, які вже довели свою ефективність у веброзробці. React, як популярна JavaScript-бібліотека, дозволяє створювати динамічний інтерфейс користувача, що швидко реагує на дії та забезпечує плавний користувацький досвід. Завдяки компонентному підходу React значно спрощує процес підтримки та масштабування застосунку в майбутньому.

MongoDB, у свою чергу, забезпечує гнучку та ефективну роботу з даними. Ця документно-орієнтована база даних найкраще підходить для зберігання великої кількості оголошень, даних про користувачів та

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				7
Змн.	Арк.	№ докум.	Підпис	Дата		

фотографій. Такий підхід гарантує надійність, гнучкість та масштабованість системи, що є критично важливим при зростанні кількості користувачів.

Метою випускної кваліфікаційної роботи є побудова та проєктування застосунку для продажу автомобілів, що має забезпечити користувачам можливість зручно підбирати, переглядати та купувати як нові, так і вживані авто онлайн, без потреби фізично відвідувати автосалони. Головна ідея полягає у створенні сучасного, зручного та надійного цифрового середовища, де кожен охочий зможе швидко знайти автомобіль, що відповідає його вимогам і бюджету.

Для досягнення поставленої мети необхідно вирішити такі **завдання**:

1. Провести огляд і аналіз аналогів застосунку, дослідити наявні рішення, їхні переваги й недоліки.
2. Здійснити вибір інструментальних засобів реалізації застосунку.
3. Спроєктувати архітектуру системи та базу даних для зберігання інформації про автомобілі, користувачів, замовлення та інші елементи.
4. Реалізувати зручні механізми оформлення замовлень, оплати та можливості для покупців відстежувати статус свого придбання.
5. Створити інтерфейс, який буде інтуїтивно зрозумілим та простим у користуванні.
6. Провести тестування застосунку для виявлення та усунення помилок.

Об'єктом дослідження є процес створення зручного та простого застосунку для продажу автомобілів, який охоплює усі етапи від подачі оголошення до технічних основ застосунку: реєстрація, база даних, інтерфейс, перевірка оголошень, спілкування з адміністратором.

Предметом дослідження є методи та інструменти розробки, що використовуються для побудови платформи, яка повинна бути не лише ефективною з технічної точки зору, але й зручною, безпечною та гнучкою у використанні.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				8
Змн.	Арк.	№ докум.	Підпис	Дата		

За темою випускної кваліфікаційної роботи бакалавра подані та прийняті до опублікування тези: Федорович Д.М., Кушнір Н.О., Лисогор Ю.І. Застосунок для продажу автомобілів. Всеукраїнська наукова-практична on-line конференція здобувачів вищої освіти і молодих учених, присвячена Дню науки, 12 травня 2025 року.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ

1.1 Постановка задачі

У сучасному цифровому середовищі значна частина торгівлі перемістилася в онлайн-простір, що відкриває нові можливості для бізнесу та споживачів. Сфера автомобільної торгівлі не є винятком – дедалі більше людей обирають купівлю чи продаж авто через Інтернет, оминаючи традиційні автосалони та ринки. У зв'язку з цим виникає потреба у створенні зручних, надійних і функціональних платформ, які б забезпечували ефективну взаємодію між продавцями та покупцями транспортних засобів.

Метою роботи є розробка сучасного застосунку для продажу автомобілів, який дозволить користувачам легко розміщувати оголошення, знаходити потрібні машини за різними параметрами, а також здійснювати комунікацію в межах платформи. Також потрібно створити інтуїтивно зрозумілий інтерфейс, який буде однаково зручним як для досвідчених користувачів, так і для тих, хто вперше користується подібними сервісами.

Під час роботи над проектом планується дослідити особливості ринку онлайн-продажу авто, проаналізувати популярні сервіси в цій сфері, визначити їхні переваги й недоліки, а також врахувати сучасні тенденції у дизайні й функціональності застосунків. Крім цього, важливим етапом стане реалізація системи безпеки, що захищатиме дані користувачів і мінімізуватиме ризики шахрайства.

Результатом роботи має стати повнофункціональний застосунок, який дозволить не лише ефективно продавати та купувати автомобілі, а й підвищуватиме загальний рівень зручності та довіри в процесі таких операцій. Цей проект зможе зробити внесок у розвиток українського цифрового авторинку та запропонувати користувачам новий рівень сервісу.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Під час проектування особлива увага приділена зручності користувача. Сучасний застосунок повинен бути максимально простим у використанні: від реєстрації до публікації оголошення має пройти мінімум часу, а сам процес не повинен викликати труднощів навіть у людей без технічної підготовки. Саме тому був продуманий кожен крок взаємодії з інтерфейсом, щоб він був логічним, зрозумілим і відповідав очікуванням користувачів.

Ще один важливий момент – це зручність використання платформи для кожного. Сьогодні пошук авто став справжньою необхідністю, тож важливо, щоб додаток був простим і зрозумілим. Люди мають змогу швидко знаходити потрібну машину, обираючи за маркою, моделлю, роком випуску, ціною, типом кузова та іншими параметрами. Це дозволяє зробити пошук максимально точним і не витратити зайвий час.

1.2 Аналіз аналогів програмного продукту

У ході огляду й аналізу ринку онлайн-платформ для продажу автомобілів в Україні, увагу було зосереджено на двох найбільш впізнаваних сервісах – AUTO.RIA та RST. Ці сайти добре знайомі користувачам, які хоча б раз замислювались про купівлю чи продаж авто, і саме тому вони стали основою для порівняння та розуміння загальних тенденцій у цій сфері.

AUTO.RIA давно закріпив за собою статус провідної онлайн-платформи для купівлі та продажу автомобілів в Україні [1]. Його впізнаваність серед користувачів – результат багаторічної присутності на ринку, постійного вдосконалення функціональності та активної роботи з довірою клієнтів. Одразу при вході на сайт створюється враження, що платформа дійсно орієнтована на людей – структура зрозуміла, інтерфейс охайний, а сам процес пошуку авто або розміщення оголошення не викликає жодних труднощів. Це показано на рис. 1.1.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		11

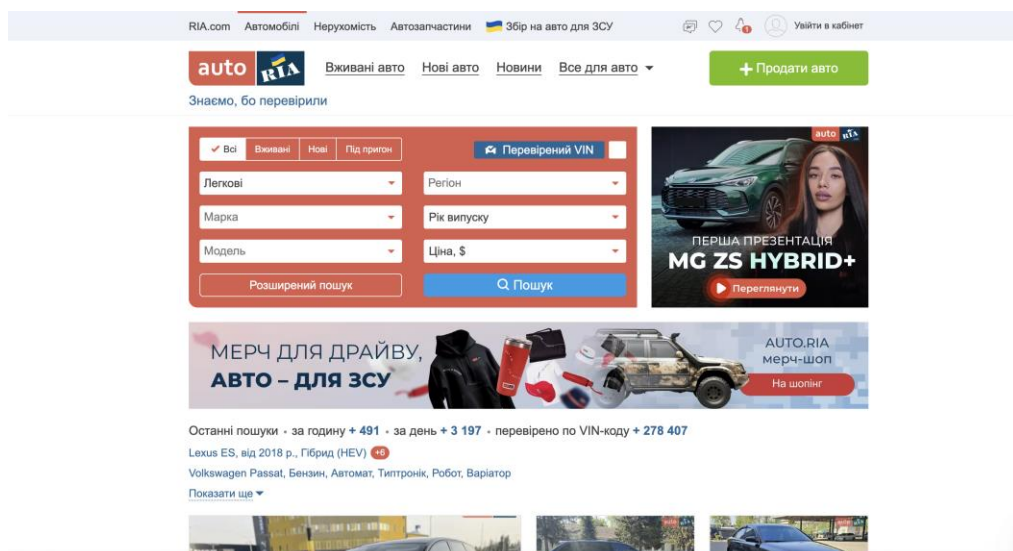


Рисунок 1.1 – Головна сторінка у застосунку AUTO.RIA

Однією з найважливіших переваг AUTO.RIA є можливість перевірки автомобіля за VIN-кодом. Це представлено на рис. 1.2. Перевірка дозволяє побачити історію транспортного засобу, включно з інформацією про пробіг, аварії, сервісне обслуговування тощо. Ця функція суттєво підвищує рівень довіри між покупцем і продавцем. Крім того, платформа активно співпрацює з автосалонами та офіційними дилерами, що дозволяє користувачам переглядати нові авто або авто з пробігом, які були ретельно перевірені.

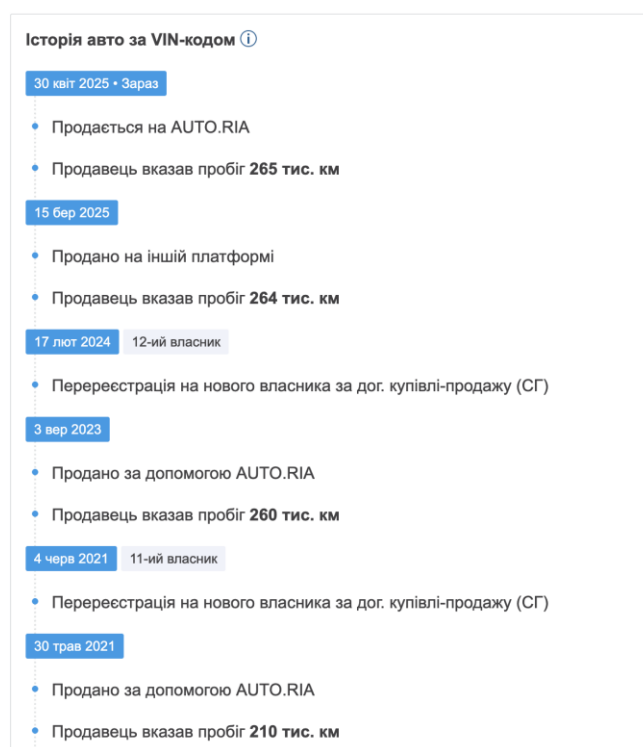


Рисунок 1.2 – Перевірка за VIN-кодом

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Розширений пошук вживаних авто

New! ☒ Всі авто ☐ Вживані ☐ Нові ☐ Під пригон

☐ Перевірений VIN-код

☐ Держ. номер

☐ Вперше на AUTO.RIA

☐ Електрокари

Тип транспорту:

Країна виробник:

Марка, модель, рік:

Рік, від: Рік, до:

[+ Ще марка](#)

[▼ Включити авто](#)

Вартість: \$

☐ Ціна з ПДВ

☐ Можливий торг

☐ Можливий обмін на автомобіль

☐ Можлива оплата частинами

Регіон:

Північ	Захід	Схід
☐ Київська	☐ Львівська	☐ Дніпропетровська
☐ Житомирська	☐ Волинська	☐ Харківська
☐ Сумська	☐ Закарпатська	☐ Донецька
☐ Чернігівська	☐ Івано-Франківська	☐ Запорізька
	☐ Хмельницька	☐ Луганська
Центр	☐ Рівненська	Південь
☐ Вінницька	☐ Тернопільська	☐ Херсонська
☐ Кіровоградська	☐ Чернівецька	☐ Миколаївська
☐ Полтавська		☐ Одеська
☐ Черкаська		

Стан

Участь в ДТП:

Технічний стан:

Лакофарбове покриття:

Авто не в Україні (під замовлення):

Нерозмитнені:

Знайдено 360 112 авто

Рисунок 1.3 – Розширений пошук авто

На рис. 1.3 відображено ще одну перевагу AUTO.RIA – це розгалужена система фільтрів для пошуку. Користувачі можуть надзвичайно точно задати свої параметри: марка, модель, рік випуску, тип палива, об'єм двигуна, кількість власників, і навіть колір чи тип коробки передач. Такий рівень деталізації не тільки економить час, а й дозволяє швидко знайти автомобіль, який максимально відповідає потребам користувача.

Усе це робить AUTO.RIA не просто дошкою оголошень, а справжнім інструментом для безпечної, зручної та прозорої купівлі-продажу авто. Саме тому при розробці власного додатку доцільно орієнтуватися на ключові ідеї цього сервісу, адаптуючи їх під власну цільову аудиторію та можливості.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		13

Іншим українським аналогом є RST – це ще одна добре відома українська платформа для купівлі та продажу автомобілів, яка тривалий час залишається популярною серед автолюбителів, незважаючи на менший масштаб у порівнянні з лідерами ринку [2]. Сайт виглядає дещо скромніше, ніж AUTO.RIA, але водночас має свою аудиторію і виконує основну функцію – дає змогу швидко знайти або продати авто. Головне, що користувач відчуває: усе працює просто, без зайвих кроків, і навіть недосвідчений відвідувач розбереться, що до чого.

Зображений на рис. 1.4. дизайн інтерфейсу RST приваблює перш за все своєю лаконічністю та відсутністю нав'язливого функціоналу. Це платформа, де головне – авто, а не інтерфейс. Користувач заходить – і зразу бачить, що шукає. Оголошення подаються у звичному форматі, з фото, основною інформацією та контактами продавця. Тут менше аналітики, немає складної перевірки VIN чи великих оглядів, але саме це подобається частині користувачів: усе чесно, без зайвих фільтрів.

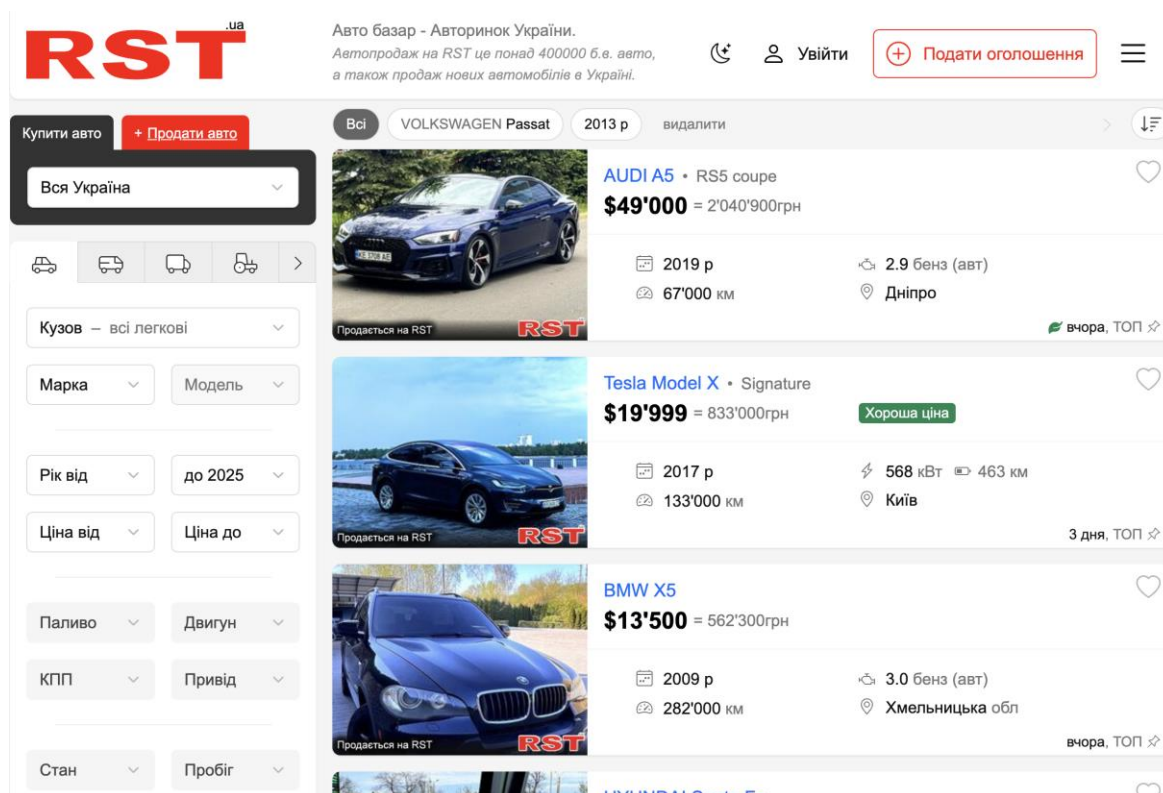


Рисунок 1.4 – Головна сторінка у застосунку RST

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

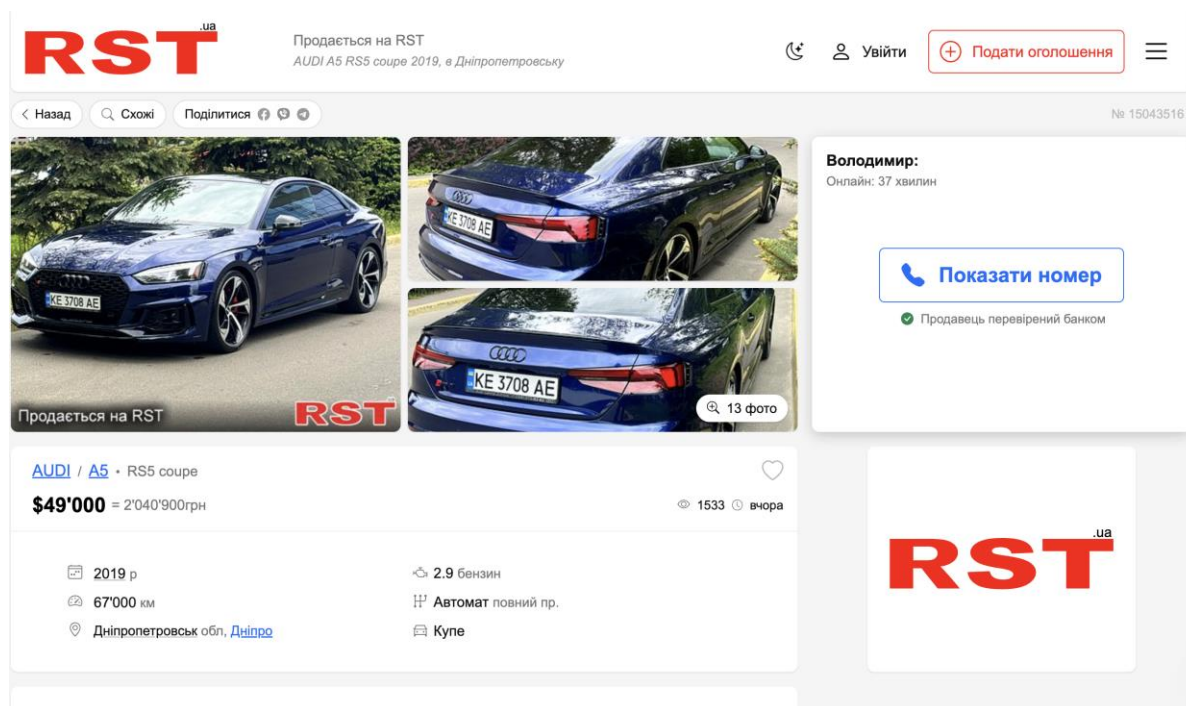


Рисунок 1.5 – Вигляд оголошення у застосунку RST

Рисунок 1.6 – Вигляд додавання оголошення у застосунку RST

Також варто зазначити, що на RST багато оголошень дублюються з інших платформ, зокрема з AUTO.RIA, що іноді створює ефект додаткової

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

вибірки. Проте, незважаючи на меншу кількість фільтрів чи інструментів, сайт має стабільну аудиторію, яка цінує простоту і прямоту.

Загалом, RST – це приклад того, як навіть у мінімалістичному форматі можна утримувати увагу тисяч користувачів. Його досвід показує, що не завжди потрібні складні технології, щоб створити дієвий сервіс. У контексті розробки власного застосунку для продажу автомобілів, RST може стати прикладом того, як важливо зберігати простоту і орієнтацію на потреби людей.

Для порівняння вище описаних застосунків було створено порівняльну таблицю (табл. 1.1).

Таблиця 1.1

Порівняння аналогів застосунку

Характеристика	AUTO.RIA	RST
Адаптивність інтерфейсу	+	+
Дизайн	Сучасний, деталізований	Простий та лаконічний
Навігаційне меню	+	+
Інформаційне наповнення	Багате: VIN-історія, поради, статистика	Мінімальне, лише необхідне
Пошукові фільтри	Розширені, гнучкі	Базові, зрозумілі
Охоплення ринку	Високе, охоплює всі регіони	Стабільна аудиторія
Простота подачі оголошення	Помірна (багато полів)	Дуже проста і швидка
Додаткові сервіси	Автоперевірка, оцінка авто, кредитування	Відсутні або мінімальні

Аналізуючи табл. 1.1, можна зробити висновок, що AUTO.RIA однозначно можна назвати лідером серед подібних сервісів в Україні. Його інтерфейс зручний, функціональність доволі широка – від звичайного розміщення оголошень до перевірки автомобіля за VIN-кодом. Видно, що розробники працювали над довірою користувачів, адже платформа пропонує і експертні огляди, і можливість перевірки історії авто. Окрім того, тут завжди оновлюється база актуальних пропозицій, що дуже важливо, коли мова йде про такий серйозний товар, як автомобіль.

RST, у свою чергу, справляє враження більш простої та приземленої платформи. Тут менше надбудов у вигляді додаткових сервісів, зате навігація зрозуміла одразу. Платформа особливо корисна тим, хто продає або шукає бюджетні автомобілі – велика частина пропозицій тут саме з нижчого цінового сегмента. І хоч сайт виглядає дещо скромніше на фоні AUTO.RIA, він виконує свою функцію – зв'язує покупців із продавцями – цілком впевнено.

Цей аналіз дав змогу зрозуміти, що навіть серед таких, на перший погляд, схожих сайтів є суттєва різниця в підході до користувача. Один орієнтується на комплексність і довіру, інший – на простоту й доступність. Це важливі спостереження, які допоможуть при створенні власного веб-додатку: варто знайти баланс між функціональністю та легкістю використання, щоб охопити якомога ширше коло користувачів.

На основі аналізу аналогів було сформовано вимоги до застосунку. Передусім, застосунок повинен дозволяти користувачам швидко знаходити бажані автомобілі та публікувати власні. Усі дії мають бути інтуїтивно зрозумілими. Оголошення мають мати достатньо інформації: опис, ціна, контактні дані, а також заповнення частини полів за VIN-кодом. Крім того, важливо забезпечити зручні фільтри для пошуку авто за параметрами, такими як марка, модель, рік випуску, тип та ціна.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		17

1.3 Вибір архітектури застосунку для продажу автомобілів

Застосунок буде побудований на клієнт-серверній архітектурі. Це забезпечить стабільність у роботі та коректну обробку інформації. Така архітектура передбачає розподіл застосунку на дві основні частини: клієнтську (те, що бачить і з чим взаємодіє користувач – інтерфейс) та серверну (яка відповідає за обробку даних, зберігання оголошень, керування базою тощо). Користувач через браузер надсилає запит (наприклад, шукає авто або додає нове оголошення), а сервер обробляє цей запит і повертає відповідь – уже готову інформацію, яку можна побачити на екрані.

Такий підхід має кілька важливих переваг. По-перше, він дозволяє гнучко розвивати систему: якщо згодом буде потрібно додати нові функції, то це можна буде зробити без повного переписування застосунку. По-друге, клієнт-серверна модель легко та просто масштабується – тобто система зможе витримати більше користувачів, якщо проєкт стане популярнішим.

На клієнтській частині працює інтерфейс, створений за допомогою бібліотеки React [3]. Вона дозволяє динамічно оновлювати вміст сторінки без перезавантаження, що суттєво підвищує зручність користування. Завдяки компонентній структурі React-код залишається гнучким, масштабованим і легко підтримуваним.

Для серверної логіки використовується середовище Node.js [4], яке забезпечує асинхронну обробку запитів і гарну масштабованість. Це особливо важливо для вебзастосунків, які можуть одночасно обслуговувати велику кількість користувачів. Зберігання та обробку даних реалізовано на базі MongoDB [5], що дозволяє ефективно працювати з неструктурованими даними і легко адаптувати схему БД до потреб, які з часом можуть змінюватися.

Крім того, така архітектура дозволяє легко інтегрувати сторонні сервіси – наприклад, платіжні системи, сервіси перевірки VIN-кодів або системи

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				18
Змн.	Арк.	№ докум.	Підпис	Дата		

push-сповіщень. У майбутньому це відкриває широкі можливості для збагачення функціоналу застосунку та покращення користувацького досвіду.

Крім того, завдяки REST API архітектурі серверна частина може легко масштабуватись або бути використаною для створення мобільного застосунку в майбутньому.

Отже, обрана архітектура дозволяє забезпечити надійну взаємодію між усіма складовими системи, що робить застосунок зручним, стабільним, сучасним і готовим до подальшого розвитку. Це рішення повністю відповідає завданням проєкту та очікуванням користувачів. А також, дозволить легко масштабувати і розвивати застосунок для потреб ринку та очікуванням користувачів.

1.4 Обґрунтування вибору інструментальних засобів

Під час вибору інструментальних засобів для розробки застосунку для продажу автомобілів основним критерієм була можливість створення зручного, швидкого та масштабованого вебпродукту з привабливим інтерфейсом. Технології, які використовуються у проєкті, мали не лише відповідати сучасним вимогам, але й забезпечувати легкість у підтримці, гнучкість структури та можливість розширення функціоналу в майбутньому. З цієї причини було обрано React для клієнтської частини та MongoDB як основну базу даних.

React став основою фронтенд-частини, оскільки він дозволяє реалізувати багатий функціонал без втрати продуктивності. Вибір саме цієї бібліотеки був зумовлений її широкою популярністю, наявністю великої спільноти, документації та безлічі готових рішень, що пришвидшують розробку. Крім того, React забезпечує високу гнучкість у створенні адаптивного дизайну, що важливо для застосунку, який повинен добре працювати на комп'ютерах.

MongoDB була обрана як основна система керування базами даних через свою здатність працювати з неструктурованими або слабо структурованими

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				19
Змн.	Арк.	№ докум.	Підпис	Дата		

даними. У контексті оголошень про автомобілі це дозволяє зберігати різні характеристики, які можуть відрізнятися від оголошення до оголошення, без необхідності постійного оновлення структури бази. Завдяки високій швидкості обробки запитів, MongoDB дозволяє реалізувати зручну та ефективну систему пошуку, що є ключовим елементом у будь-якому комерційному застосунку.

Для розробки застосунку було обрано середовище WebStorm [6] – потужну інтегровану середу розробки, яка забезпечує комфортну роботу з JavaScript [7] та React, а також має вбудовану підтримку роботи з системами контролю версій. WebStorm має розширені можливості для автодоповнення коду, перевірки синтаксису, навігації по проєкту та зручної роботи з модулями, що значно пришвидшує процес розробки. Завдяки своїй гнучкості та багатofункціональності, ця IDE стала надійним інструментом на всіх етапах створення застосунку – від проектування архітектури до реалізації та тестування.

Для зберігання коду та керування версіями було використано GitLab [8]. Це рішення дозволило забезпечити безперервну інтеграцію (CI) та безперервне розгортання (CD), що особливо важливо при частих оновленнях і змінах у проєкті. GitLab також слугує як засіб резервного копіювання коду та управління завданнями в команді, а його приватні репозиторії гарантують безпеку збережених даних. Наявність гнучкої системи гілок дозволила ефективно впроваджувати нові функції, не заважаючи стабільній роботі основної версії застосунку.

Таким чином, використання WebStorm у поєднанні з GitLab забезпечило стабільний і безпечний процес розробки, дозволяючи підтримувати високу якість коду та ефективно впроваджувати нові рішення в рамках проєкту.

Вибір React та MongoDB як основних технологій для створення застосунку дозволив поєднати ефективну взаємодію з користувачем на фронтенді з надійним та гнучким зберіганням даних на бекенді. Обрані

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		20

інструменти повністю відповідають вимогам сучасної розробки та дозволяють створити конкурентно спроможний продукт на ринку онлайн-продажу автомобілів.

Висновки до першого розділу

У першому розділі було закладено основу для розробки застосунку, покликаного спростити процес купівлі та продажу автомобілів в онлайн-середовищі. Під час дослідження проаналізовано актуальні потреби ринку, оцінено досвід існуючих рішень та сформовано вимоги до того, яким має бути сучасний, зручний і надійний цифровий продукт у цій сфері. Все це дозволяє створити стабільне середовище для розробки, тестування та подальшого масштабування продукту. Завдяки цьому вся подальша розробка виконується за чітко визначеними технічними й концептуальними вимогами, що дозволило успішно реалізувати застосунок.

Також були обрані інструментальні засоби реалізації програмного продукту, а саме React, MongoDB, WebStorm та GitLab. Використано сучасні технології, що відповідають вимогам адаптивності, продуктивності та зручності використання.

На основі відповідних вимог та критеріїв був сформований набір основних технологій і засобів реалізації для розробки застосунку.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		21

РОЗДІЛ 2 ПРОЕКТУВАННЯ ЗАСТОСУНКУ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ

2.1 Визначення варіантів використання та об'єктно-орієнтованої структури системи

Застосунок для продажу автомобілів надає зручний інтерфейс, через який користувачі можуть створювати оголошення, переглядати наявні пропозиції, а також застосовувати фільтри для пошуку потрібного авто. Ефективність роботи системи значною мірою залежить від активності її користувачів, адже саме вони наповнюють платформу змістом і підтримують її життєздатність.

У рамках платформи користувачів умовно можна поділити на дві основні категорії: продавці, які розміщують оголошення про продаж автомобілів, та покупці, які шукають авто, переглядають пропозиції та зв'язуються з продавцями для уточнення деталей або домовленості про угоду. Оголошення включають ключову інформацію про автомобіль: марку, модель, опис, фото, ціну, що допомагають потенційним покупцям швидко оцінити релевантність пропозиції.

Для кращого розуміння взаємодії між користувачами та основними компонентами системи було створено діаграму варіантів використання (рис. 2.1), яка наочно демонструє ключові сценарії поведінки користувачів у межах застосунку.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		22

Таблиця 2.2

Опис прецеденту “Перегляд автомобілей”

Назва варіанту використання	Перегляд автомобілей
Основні актори	Користувачі
Короткий опис	Користувачі мають змогу переглядати, шукати та сортувати оголошення, які були додані на продаж.

Таблиця 2.3

Опис прецеденту “Перегляд новин”

Назва варіанту використання	Перегляд новин
Основні актори	Користувачі
Короткий опис	Користувачі мають змогу переглядати новини зі світу автомобілів.

Таблиця 2.4

Опис прецеденту “Створення оголошень”

Назва варіанту використання	Створення оголошень
Основні актори	Користувачі
Короткий опис	Авторизовані користувачі мають змогу створення оголошень.

Таблиця 2.5

Опис прецеденту “Чат з адміністрацією”

Назва варіанту використання	Чат з адміністрацією
Основні актори	Адміністратор, користувач
Короткий опис	Можливість контакту між користувачем та адміністратором для вирішення питань, які можуть з’явитися у користувача.

Таблиця 2.6

Опис прецеденту “Верифікація оголошень”

Назва варіанту використання	Верифікація оголошень
Основні актори	Адміністратор, користувач
Короткий опис	Після створення оголошення користувачем, воно потрапляє на верифікацію до адміністратора, а вже після цього з’являється у пошуку.

Таблиця 2.7

Опис прецеденту “Редагування оголошень”

Назва варіанту використання	Редагування оголошень
Основні актори	Адміністратор
Короткий опис	Адміністратор має змогу редагувати оголошення, якщо воно не відповідає вимогам.

Основана мета розробки застосунку для продажу автомобілів – забезпечити користувачам зручний, швидкий і зрозумілий інструмент для взаємодії з автомобільним ринком. Основна увага приділена простоті користування та доступності ключової інформації – від перегляду оголошень до комунікації з продавцем. Застосунок дозволяє ефективно шукати автомобілі за заданими параметрами, переглядати їх технічні характеристики, порівнювати варіанти та приймати зважені рішення.

Інтерфейс розроблено таким чином, щоб навіть користувачі без технічної підготовки могли інтуїтивно орієнтуватися в системі. Кожен елемент сторінки логічно розташований, що дозволяє швидко знаходити потрібні розділи та оголошення. Це значно спрощує процес пошуку та купівлі автомобіля, зменшуючи кількість кроків, які користувач повинен виконати для досягнення своєї мети.

Для забезпечення неперервного виконання необхідних функцій і відповідності встановленим стандартам та специфікаціям, застосунок для продажу автомобілів повинен задовольняти наступні високорівневі вимоги.

Бізнес вимоги можна сформулювати таким чином.

Проект створюється для комфортного та ефективного продажу та купівлі автомобілів користувачами. Метою є оптимізація процесів купівлі та продажу автомобілів та створення безпечних умов для користувачів.

Проект буде реалізовано у вигляді веб-застосунку з простим та комфортним дизайном, що зробить комфортний і швидкий пошук автомобілів, а також продаж з швидким створенням оголошенням.

Вимоги користувачів є такими: можливість перегляду актуальних оголошень, можливість фільтрації оголошень за різними параметрами (ціна, рік випуску і т.д.), можливість спілкування з адміністрацією застосунку, можливість просувати оголошення на початок списку, можливість автоматичного заповнення полів за VIN-кодом авто.

В свою чергу, **вимогами персоналу застосунку** є можливість верифікації автомобілів з адмін профілю, можливість спілкування з користувачами, можливість переглядати всі оголошення користувачів (верифіковані та неверифіковані).

Функціональні вимоги є такими: при реєстрації та вході до системи користувач повинен мати можливість створити обліковий запис, увійти в систему та отримати доступ до персонального кабінету; при додаванні оголошень про продаж авто платформа повинна дозволяти користувачу створювати нові оголошення, вказуючи технічні характеристики автомобіля, фото, опис, ціну та контактні дані; при перегляді і фільтрації оголошень користувач повинен мати змогу переглядати всі актуальні оголошення, застосовувати фільтри (ціна, рік, пробіг, марка тощо) та знаходити потрібні варіанти; при комунікації з адміністратором реалізується можливість надіслати повідомлення або звернення безпосередньо через застосунок; для

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

підняття оголошення вгору списку користувач може активувати функцію просування свого оголошення для більшої видимості; для автоматичного заповнення оголошення за VIN-кодом система може автоматично підтягувати характеристики авто після введення VIN; для реалізації адміністративних функцій адміністратор має доступ до всіх оголошень, можливість їх перевірки, верифікації та спілкування з користувачами.

В свою чергу, **нефункціональні вимоги** є такими.

Інтерфейс має бути інтуїтивно зрозумілим – нові користувачі повинні мати змогу освоїти його приблизно за 10 хвилин, тоді як досвідченим користувачам достатньо 3–5 хвилин. Система повинна забезпечувати високу швидкодію: звичайні операції виконуються не довше ніж за 5 секунд, а складні запити – до 10 секунд. Надійність системи є критично важливою – обслуговування не повинно перевищувати 5% часу її роботи, середній час безвідмовної роботи повинен складати не менше 90 днів, а допустимий рівень помилок – не більше 3 на кожні 10 000 рядків коду. Крім того, система має бути продуктивною та здатною стабільно функціонувати при одночасному використанні не менше 100 користувачами.

Системні вимоги

Вимоги до середовища виконання передбачають мінімальну апаратну конфігурацію пристроїв користувачів для забезпечення стабільної та коректної роботи застосунку. Зокрема, необхідно мати не менше 2 ГБ оперативної пам'яті, двоядерний процесор із тактовою частотою не менше ніж 1.0 GHz, а також стабільне під'єднання до Інтернету, що забезпечує безперебійну передачу даних і коректне функціонування всіх сервісів застосунку. Рекомендовано використовувати сучасні веб-браузери з актуальними оновленнями.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		27

2.2 Розробка бази даних системи

СУБД MongoDB є основною системою зберігання даних завдяки своїй гнучкості у роботі з напівструктурованими даними, можливості горизонтального масштабування та високій продуктивності при операціях читання й запису. Це особливо важливо для систем, де структура об'єктів (зокрема, автомобілів) може змінюватися або доповнюватися новими параметрами. MongoDB дозволяє легко модифікувати систему без порушення вже збережених даних, що критично для швидко розвиваного застосунку.

Крім того, MongoDB проста у використанні, має інтуїтивно зрозумілий синтаксис для запитів і без проблем інтегрується з сучасними технологіями, такими як React. Це значно прискорює розробку та полегшує обслуговування системи. Широка спільнота користувачів і якісна офіційна документація забезпечують підтримку на всіх етапах від розгортання до масштабування.

У базі даних застосунку для продажу автомобілів повинна міститися інформація як про користувачів, так і про автомобілі, оскільки обидва типи даних є ключовими для ефективної роботи системи та забезпечення зручної взаємодії з користувачами.

У майбутньому система може бути доповнена іншими колекціями, наприклад, для зберігання відгуків, історії переглядів, аналітики або інтеграції з сервісами перевірки авто. Гнучкість MongoDB дозволяє реалізовувати ці розширення без порушення вже існуючої структури даних, що робить її ідеальним рішенням для цього типу застосунку.

У результаті сформованих функціональних та нефункціональних вимог було створено діаграму колекцій, які використовуються для роботи додатку. Діаграму наведено на рис. 2.2.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		28

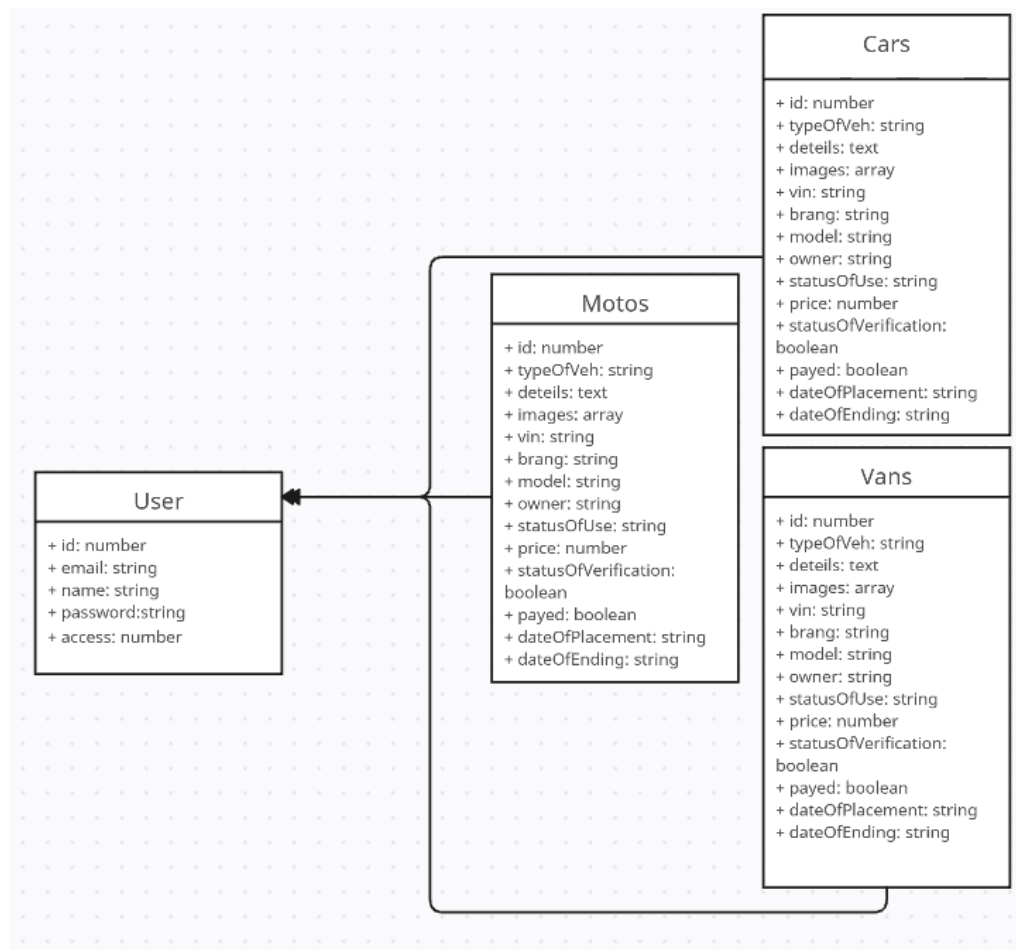


Рисунок 2.2 – Діаграма колекцій застосунку для продажу автомобілів

Опишемо діаграму колекцій, наведену на рис. 2.2.

Колекція “User” зберігає інформацію про користувачів: електронну пошту, ім’я, пароль та рівень доступу (наприклад, звичайний користувач або адміністратор). Головне призначення цієї колекції – забезпечення автентифікації, реєстрації користувачів та управління їхніми правами доступу.

Колекція “Cars” описує автомобілі, які користувачі можуть додавати до системи. Тут зберігаються тип транспорту, назва, деталі, фото, VIN-код, бренд, модель, статус використання (наприклад, новий або вживаний), перевірений стан (верифіковане оголошення або ні), ціна, дата розміщення й закінчення оголошення.

Колекції “Motos” та “Vans” повторюють поля колекції “Cars” для зручності створення оголошень.

Опис кожної колекції наведений у табл. 2.8 – 2.11.

Таблиця 2.8

Опис колекції User

Поле	Тип даних	Опис
_id	ObjectID	Ідентифікатор користувача
email	string	Е-mail користувача
name	string	Ім'я користувача
password	string	Пароль
access	number	Рівень доступу

Таблиця 2.9

Опис колекції Cars

Поле	Тип даних	Опис
_id	ObjectID	Ідентифікатор транспорту
typeOfVeh	string	Тип транспорту
deteils	text	Опис транспорту
images	array	Масив посилань на фото
vin	string	VIN-код транспорту
brang	string	Марка транспорту
model	string	Модель транспорту
owner	string	Власник транспорту (User)
statusOfUse	string	Новий чи вживаний
price	number	Ціна транспорту
statusOfVerification	boolean	Верифіковане або ні
payed	boolean	Проплачене оголошення або ні
dateOfPlacement	string	Дата розміщення
dateOfEnding	string	Дата закінчення

Таблиця 2.10

Опис колекції Vans

Поле	Тип даних	Опис
_id	ObjectID	Ідентифікатор транспорту
typeOfVeh	string	Тип транспорту
deteils	text	Опис транспорту
images	array	Масив посилань на фото
vin	string	VIN-код транспорту
brang	string	Марка транспорту
model	string	Модель транспорту
owner	string	Власник транспорту (User)
statusOfUse	string	Новий чи вживаний
price	number	Ціна транспорту
statusOfVerification	boolean	Верифіковане або ні
payed	boolean	Проплачене оголошення або ні
dateOfPlacement	string	Дата розміщення
dateOfEnding	string	Дата закінчення

Таблиця 2.11

Опис колекції Motos

Поле	Тип даних	Опис
_id	ObjectID	Ідентифікатор транспорту
typeOfVeh	string	Тип транспорту
deteils	text	Опис транспорту
images	array	Масив посилань на фото
vin	string	VIN-код транспорту
brang	string	Марка транспорту
model	string	Модель транспорту
owner	string	Власник транспорту (User)
statusOfUse	string	Новий чи вживаний
price	number	Ціна транспорту
statusOfVerification	boolean	Верифіковане або ні

Продовження табл. 2.11

payed	boolean	Проплачене оголошення або ні
dateOfPlacement	string	Дата розміщення
dateOfEnding	string	Дата закінчення

Після створення оголошення дані, які зберігаються в текстовому виді, зберігаються на MongoDB, а фото оголошення зберігаються у вигляді посилань на сервіс Cloudinary. Пароль користувача хешується для безпеки даних користувача.

2.3 Проєктування та реалізація алгоритмів роботи системи

Розроблений застосунок є простим та інтуїтивно зрозумілим у користуванні, що дозволяє кожному користувачу – незалежно від досвіду – легко взаємодіяти з платформою. Основні функції реалізовані таким чином, щоб максимально спростити процес купівлі та продажу автомобілів. Загальна логіка роботи з платформою показана на рис. 2.3, розглянемо її детально.

Якщо авторизувався адміністратор:

- після авторизації має змогу відкрити чат з авторизованими користувачами;
- у меню має змогу верифікувати оголошення;
- має змогу перегляду всіх оголошень.

Якщо авторизувався користувач:

- користувач має змогу додати нове оголошення;
- користувач має змогу написати адміністратору через чат;
- користувач має змогу переглядати свої оголошення;
- користувач має змогу переглядати свої неверифіковані оголошення.

Якщо користувач неавторизувався, то він має змогу тільки переглядати оголошення інших та новини на головній сторінці.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

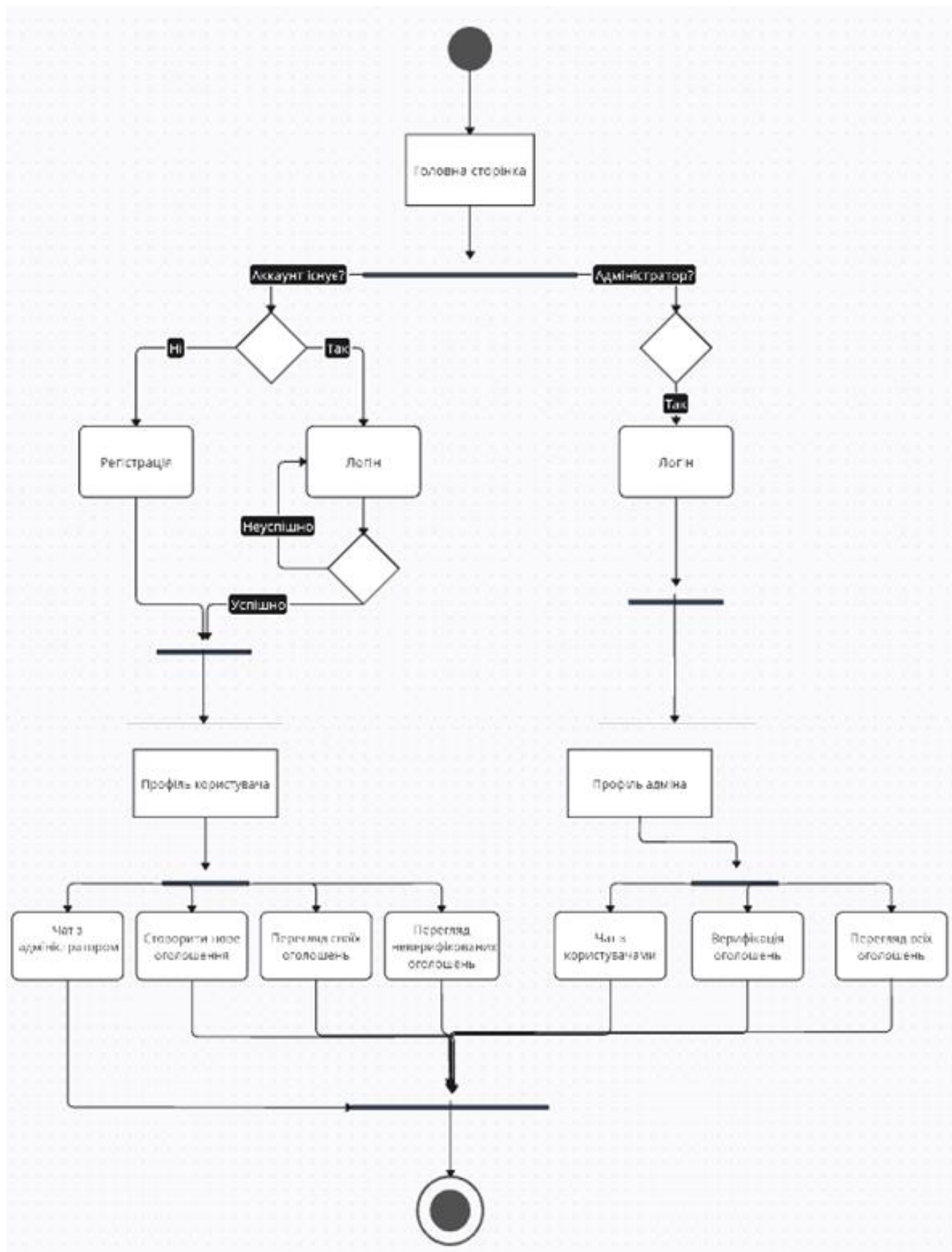


Рисунок 2.3 – Діаграма активності

2.4 Реалізація функціоналу застосунку для продажу автомобілів

Почнемо з етапу реалізації функціоналу реєстрації та авторизації в застосунку, де користувачі зможуть вводити необхідні дані для реєстрації та авторизації для подальшого використання застосунку.

```
try {
  console.log(req.body)
  const { email, name, password } = req.body;

  const alreadyUser = await users.findOne({ email });
  if (alreadyUser) {
    return res.status(400).json({ error: "Такий користувач вже існує" });
  }

  const hashedPassword = await bcrypt.hash(password, 10);
  const user = new users({ email, name, password: hashedPassword });

  await user.save();
  res.status(201).json({ message: "User registered successfully", user });
} catch (error) {
  res.status(500).json({ error: "Internal Server Error" });
}
```

Цей фрагмент коду використовується на серверній частині застосунку, в ньому виконується перевірка чи є зареєстрований користувач з таким Email. Якщо такий користувач існує, то користувачу покаже помилку, що такий вже існує. Якщо ж такого користувача немає, то сервер пропустить користувача далі та успішно зареєструє.

При авторизації виконується також перевірка по Email, чи такий користувач існує. Якщо користувач існує, то на сервер відправляється статус код, що користувач існує і успішно пропускає його у застосунок. Відповідний фрагмент коду:

```
const { email, password } = req.body;

const user = await users.findOne({ email });
if (!user) {
  return res.status(401).json({ error: "Authentication failed" });
}

const isPasswordMatch = await bcrypt.compare(password, user.password);
if (!isPasswordMatch) {
```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		34

```

    return res.status(401).json({ error: "Authentication failed" });
  }

```

Також на стороні користувача при авторизації приймаються його основні дані у вигляді Email та пароля та за допомогою “useState” [9] зберігається локальний стан “formData” і цей стан буде оновлюватися, коли користувач буде вводити дані у форму:

```

const [formData, setFormData] = useState({ email: "", password: "" });

```

Далі, якщо користувач захоче додати авто, то спочатку виконується перевірка чи користувач авторизований через хук “useEffect” [10]. Він виконує перевірку, чи є storedUser. Якщо ні, то перенаправляє на сторінку login, а якщо існує, то використовує його Email у додаванні оголошення:

```

useEffect(() => {
  const storedUser = localStorage.getItem("user");
  if (!storedUser) {
    router.push("/login");
  } else {
    const user = JSON.parse(storedUser);
    setUserEmail(user.user.email);
    setFormData((prev) => ({ ...prev, owner: user.user.email }));
  }
}, [router]);

```

Нижче зображений фрагмент коду, за допомогою якого здійснюється запит до API, щоб отримати інформацію про автомобіль і заповнити поля у формі. Цей фрагмент є прикладом асинхронного запиту в React:

```

if (vin.length !== 17) {
  setVinError("VIN повинен містити 17 символів");
  return;
}
setVinError(null);
setVinCheckLoading(true);
try {
  const response = await
  fetch(`https://vpic.nhtsa.dot.gov/api/vehicles/DecodeVin/${vin}?format=json`);
  const data = await response.json();
  const make = data.Results.find((r: any) => r.Variable === "Make").Value;
  const model = data.Results.find((r: any) => r.Variable === "Model").Value;

  if (make && !formData.brand) {
    setFormData((prev) => ({ ...prev, brand: make }));
    setVinDetectedBrand(true);
  }
}

```

```

    if (model && !formData.model) {
      setFormData((prev) => ({ ...prev, model }));
      console.log(model)
    }
  } catch (err) {
    console.error("Помилка VIN-перевірки", err);
    setVinError("Не вдалося перевірити VIN");
  } finally {
    setVinCheckLoading(false);
  }
};

```

Наступний фрагмент коду показує реалізацію завантаження зображень у форматі “Multipart/form-data”. Цей підхід виконано для більш зручної та швидкої інтеграції серверної та клієнтської частин застосунку:

```

const uploadImageAsBinary = async (file: File): Promise<string> => {
  const reader = new FileReader();
  const binaryData = await new Promise<ArrayBuffer>((resolve, reject) => {
    reader.onloadend = () => resolve(reader.result as ArrayBuffer);
    reader.onerror = reject;
    reader.readAsArrayBuffer(file);
  });
  const formData = new FormData();
  formData.append("image", new Blob([binaryData]), file.name);
  try {
    const response = await api.post("/upload-images", formData, {
      headers: { "Content-Type": "multipart/form-data" },
    }
  );
};

```

Наведений нижче фрагмент коду відображає частину React Redux [11] логіки, яка відповідає за додавання нового авто в глобальний стан. Якщо запит виконано успішно, то користувач перенаправляється на головну сторінку, а додане авто відправляється на верифікацію адміністратору:

```

if (formData.payed) {
  localStorage.setItem("pendingCar", JSON.stringify({ ...payload, payed: true }));
  router.push("/payment");
} else {
  const result = await dispatch(addCar(payload));
  if (addCar.fulfilled.match(result)) {
    router.push("/");
  }
}
};

```

Наступний наведений фрагмент коду реалізує фільтрацію масиву автомобілей, тобто всіх оголошень, які є у застосунку відповідно до вибраних користувачем критеріїв. Якщо певний фільтр не задано, то відповідна умова автоматично вважається істинною, що дозволяє відображати всі авто без обмежень. Результати фільтрації зберігаються у стані “filteredCars”, який може використовуватись для відображення на сторінці:

```
const filtered = cars.filter((car) => {
  return (
    (filters.name ? car.name.toLowerCase().includes(filters.name.toLowerCase()) : true) &&
    (filters.statusOfUse ? car.statusOfUse === filters.statusOfUse : true) &&
    (filters.brand ? car.details.includes(filters.brand) : true) &&
    (filters.minPrice ? car.price >= Number(filters.minPrice) : true) &&
    (filters.maxPrice ? car.price <= Number(filters.maxPrice) : true)
  );
});
setFilteredCars(filtered);
```

Далі фрагмент коду демонструє використання React хука “useEffect” для ініціалізації створення платіжного наміру на сервері при завантаженні компонента. Всередині хука викликається асинхронна функція “fetchClientSecret”, яка відправляє POST-запит на “/api/create-payment-intent” з сумою платежу. У відповідь сервер повертає об’єкт із “clientSecret”, який необхідний для подальшої інтеграції з платіжною системою, наприклад Stripe. Отриманий “clientSecret” зберігається у стані для подальшого використання в інтерфейсі оплати:

```
useEffect(() => {
  const fetchClientSecret = async () => {
    try {
      const response = await fetch("/api/create-payment-intent", {
        method: "POST",
        body: JSON.stringify({ amount: 500 }),
        headers: {
          "Content-Type": "application/json",
        },
      });
      const data = await response.json();
      setClientSecret(data.clientSecret);
    } catch (error) {
      console.error("Ошибка получения client_secret", error);
    }
  };
  fetchClientSecret();
}, []);
```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		37

```

    }
  };

```

Нижче наданий код перевіряє, чи користувач має рівень доступу 1 (адміністратор). Якщо так, то завантажує всі типи транспортних засобів. Якщо ні, то перенаправляє користувача на головну сторінку:

```

useEffect(() => {
  const storedUser = localStorage.getItem("user");
  ...
  if (user.user.access === 1) {
    setIsAuthorized(true);
    dispatch(fetchCars());
    dispatch(fetchMotos());
    dispatch(fetchVans());
  } else {
    router.replace("/");
  }
},
[]);

```

Наведений нижче фрагмент коду оновлює статус підтвердження авто, відповідний dispatch, а потім відправляє лист власнику. Вона демонструє асинхронну логіку взаємодії з Redux та використання побічних ефектів, що дуже важливо в сучасних SPA:

```

const handleVerify = async (car: Car) => {
  switch (car.typeOfVeh) {
    case "car":
      await dispatch(updateCarVerification(...));
      await dispatch(fetchCars());
      break;
    ...
  }
  await sendConfirmationEmail(car.owner, car.name);
};

```

Наступний фрагмент коду реалізує відправлення листа про підтвердження оголошення і того, що воно опубліковано у застосунку. Функція використовує “emailjs.send()” для надсилання даних у заздалегідь створений шаблон, де змінні “to_email” та “car_name” підставляються автоматично:

```

export const sendConfirmationEmail = async (toEmail: string, carName: string) => {
  console.log(toEmail, carName)
  try {
    const res = await emailjs.send(
      "service_b1h16io", // service id

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

```

"template_ypn2uaj", // your_template_id
{
  to_email: toEmail,
  car_name: carName,
},
"Z4xECnLxPxHbTZWV4" // your_public_key
);
console.log("Підтвердження на email надіслано:", res.text);
} catch (error) {
  console.error("Помилка надсилання листа:", error);
}
};

```

Наведений нижче фрагмент коду використовує хук “useEffect” з метою автоматичного завантаження новин про автомобілі одразу після того, як компонент з’явиться на сторінці. У межах “useEffect” викликається асинхронна функція “fetchNews”, яка відправляє запит до API сервісу NewsData за допомогою функції “fetch”. У разі успішної відповіді дані розшифровуються з формату JSON, після чого результати новин зберігаються у стані компонента за допомогою функції “setArticles”:

```

useEffect(() => {
  const fetchNews = async () => {
    try {
      const res = await fetch(
        `https://newsdata.io/api/1/latest?apikey=pub_79155479e55d5c9e119e24a142e8229534fe6&q=%
D0%BD%D0%BE%D0%B2%D1%96%20%D0%B0%D0%B2%D1%82%D0%BE%D0%BC%
D0%BE%D0%B1%D1%96%D0%BB%D1%96`
      );
      const data = await res.json();
      setArticles(data.results || []);
    }
    catch (err) {
      setError("Не вдалося завантажити новини");
    }
    finally {
      setLoading(false);
    }
  };

  fetchNews();
},
[]);

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

Наступний фрагмент коду описує компонент “CarCard” у React – це картка, яка виводить стислу інформацію про автомобіль. Він приймає об’єкт car як пропс, який містить усі необхідні дані для відображення: фото, назву, опис, ціну, статус перевірки та тип авто. Унизу картки розташована кнопка-посилання на сторінку з повною інформацією про авто. Посилання формує URL динамічно – на основі типу авто “car.typeOfVeh” і його унікального ідентифікатора “car._id”. Загалом, компонент побудований з використанням Tailwind CSS, що забезпечує адаптивний і сучасний зовнішній вигляд. Така картка легко масштабується й може використовуватись у списках, галереях або результатах пошуку:

```
export default function CarCard({ car }: CarCardProps) {
  return (
    <div className="bg-white rounded-lg shadow-md overflow-hidden">
      <Image
        src={car.images[0] || "/placeholder.jpg"}
        alt={car.name}
        width={300}
        height={200}
        className="w-full h-48 object-cover"
      />
      <div className="p-4">
        <h2 className="text-xl font-semibold">{car.name}</h2>
        <p className="text-gray-600">{car.details}</p>
        <p className="text-lg font-bold text-blue-600 mt-2">{car.price} $</p>
        <p className={`text-sm mt-2 ${car.statusOfVerification ? "text-green-500" : "text-red-500"}`}>
          {car.statusOfVerification ? "Перевірено" : "Не перевірено"}
        </p>
        <Link
          href={`/${car.typeOfVeh}/${car._id}`}
          className="block text-center bg-blue-500 text-white px-4 py-2 mt-4 rounded-lg"
        >
          Деталі
        </Link>
      </div>
    </div>
  );
}
```

Висновки до другого розділу

У процесі розробки застосунку для продажу автомобілів було детально проаналізовано можливі сценарії його використання відповідно до поставлених вимог. На основі цього аналізу було спроектовано загальну структуру платформи, з урахуванням логіки взаємодії між основними її компонентами.

Особливу увагу приділено створенню зрозумілої архітектури та бази даних, яка відповідає реальним потребам користувачів. Було спроектовано зберігання ключових даних: інформації про автомобілі, профілі користувачів, а також додаткові категорії, що допомагають у пошуку й навігації по сайту.

Крім того, було описано покрокову логіку користування застосунком: від перегляду авто та їх фільтрації до комунікації з адміністратором. У центрі розробки – простота для користувача: кожна функція продумана так, щоб взаємодія з платформою була інтуїтивно зрозумілою.

Основний функціонал включає можливість розміщення оголошень про продаж автомобіля, їхнє підтвердження адміністрацією та подальшу публікацію у застосунку. Також реалізовано зручні механізми для комунікації між користувачами і адміністратором.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		41

РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ ІЗ ЗАСТОСУНКОМ ДЛЯ ПРОДАЖУ АВТОМОБІЛІВ

3.1 Порядок встановлення та налаштування параметрів системи

Налаштування та запуск застосунку для продажу автомобілів передбачає роботу з кількома окремими компонентами системи. Кожен із них має свою специфіку, тому потребує індивідуального розгортання та налаштування. Щоб краще зрозуміти, як ці частини пов'язані між собою і як саме вони взаємодіють, було створено діаграму розгортання. Вона наочно демонструє загальну структуру системи та допомагає уявити, як усе працює у комплексі (рис. 3.1).

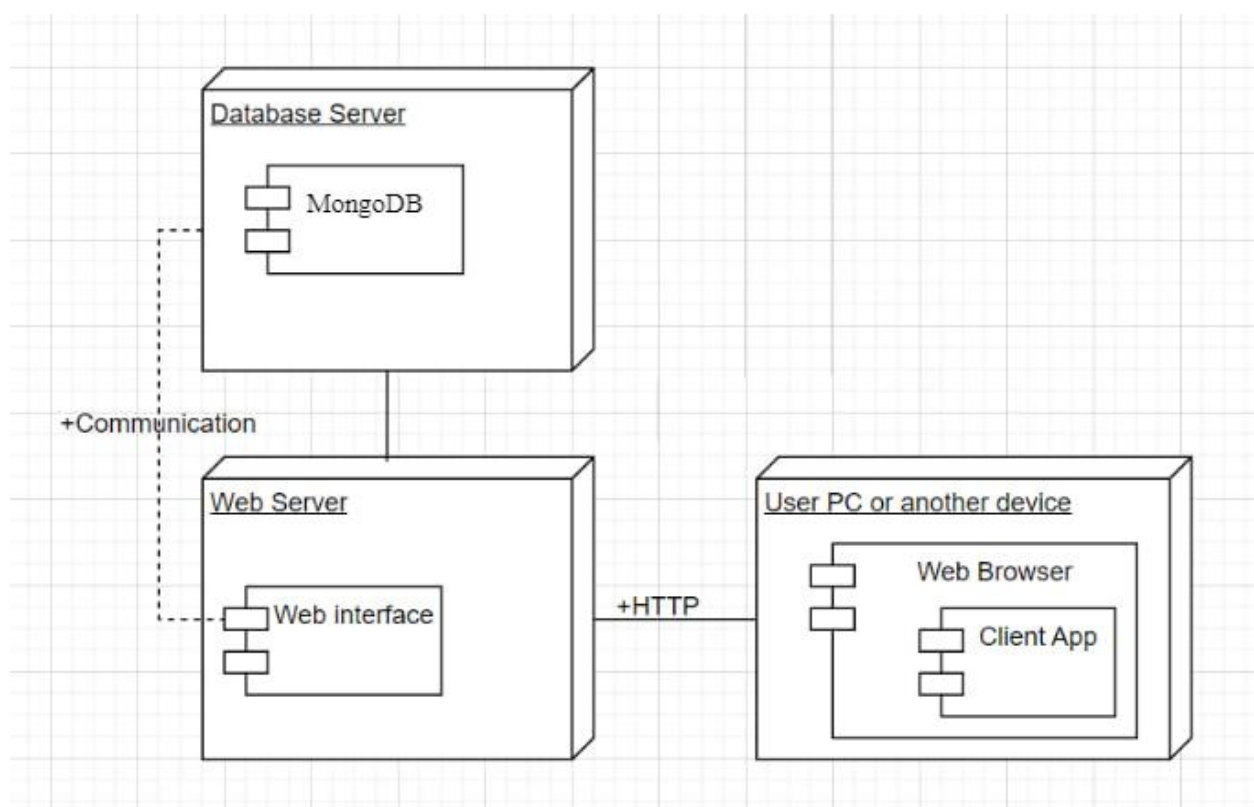


Рисунок 3.1 – Діаграма розгортання застосунку

Ця діаграма розгортання ілюструє, як працює система, розділена на три ключові частини: клієнтський пристрій (наприклад, комп'ютер або смартфон), вебсервер і сервер бази даних.

Коли користувач відкриває вебзастосунок, його пристрій надсилає запити до вебсервера через звичний нам інтернет-протокол HTTP. Це той

самий механізм, за допомогою якого щодня переглядаються сайти. HTTP працює на базі надійного мережевого протоколу TCP, який гарантує, що всі дані будуть передані точно і без втрат, навіть якщо мережа нестабільна.

Далі вебсервер, отримавши запит, при потребі звертається до сервера бази даних – наприклад, щоб дізнатися, які автомобілі доступні, або перевірити дані користувача. Це відбувається через протокол TCP/IP, що забезпечує стабільну та захищену передачу інформації між серверами.

Загалом, така архітектура дозволяє системі працювати чітко та злагоджено: користувачі отримують швидкий і безперебійний доступ до потрібної інформації, а всі компоненти платформи – взаємодіють без збоїв. Це створює надійну основу для ефективної роботи онлайн-сервісу з продажу автомобілів.

Крім того, таке розділення обов'язків між клієнтом, сервером і базою даних спрощує масштабування системи у майбутньому. Наприклад, можна окремо оптимізувати кожен компонент: збільшити продуктивність бази, балансувати навантаження на вебсервер або адаптувати інтерфейс під нові пристрої. Такий підхід забезпечує гнучкість розробки, стабільність у роботі та готовність до подальших оновлень.

3.2 Структура інтерфейсу застосунку для продажу автомобілів

Після відкриття застосунку, користувач потрапляє на головну сторінку застосунку, яка зображена на рис. 3.2. На ньому користувач бачить всі оголошення від інших користувачів і одразу може відсортувати їх або виконати пошук по назві. Фільтрація або сортування виконується за декількома параметрами – новий чи вживаний, за типом, за маркою, а також сортування від мінімальною та максимальною ціною. Також користувач бачить актуальні новини з автосвіту, які постійно оновлюються. Цей функціонал реалізований за допомогою сервісу NewsData [12].

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		43

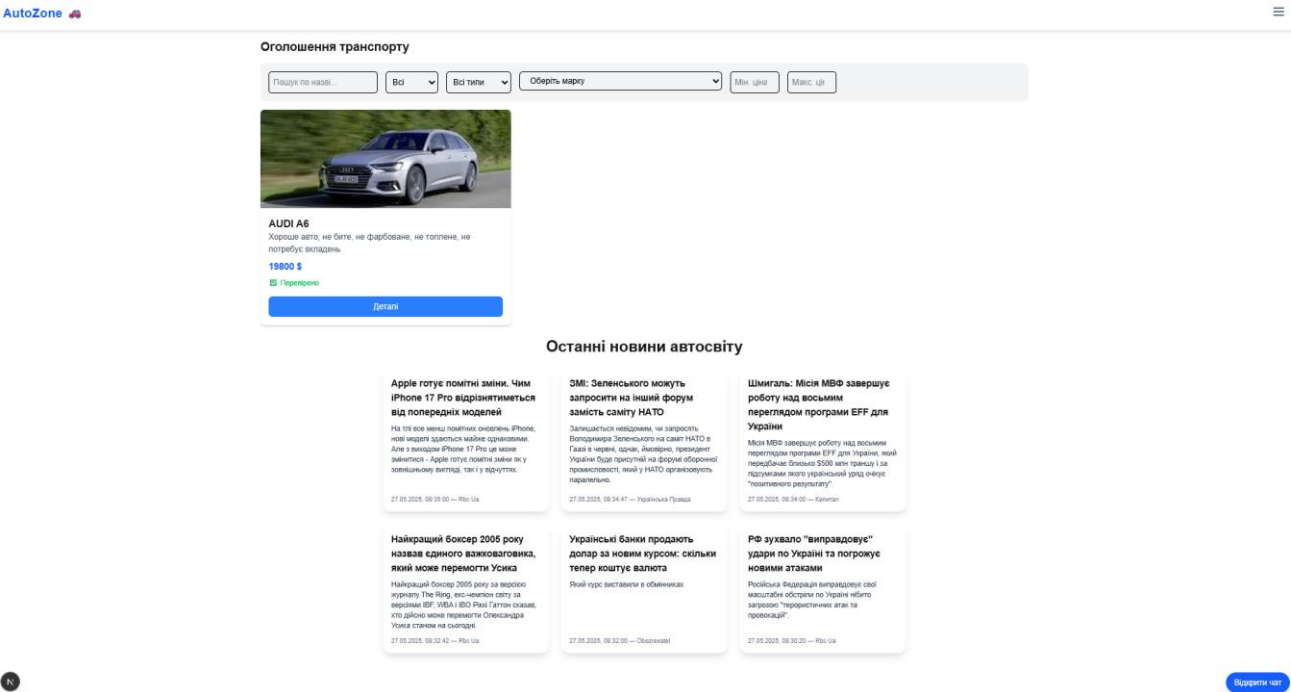


Рисунок 3.2 – Головна сторінка застосунку

Після того як користувач обрав автомобіль, який його цікавить, він потрапляє на сторінку оголошення, яка зображена на рис. 3.3. На ній користувач бачить основні параметри, який вказав продавець. Це бренд автомобіля, модель, ціну, тип, стан, VIN-код, опис, контактний Email та фотографії автомобіля.

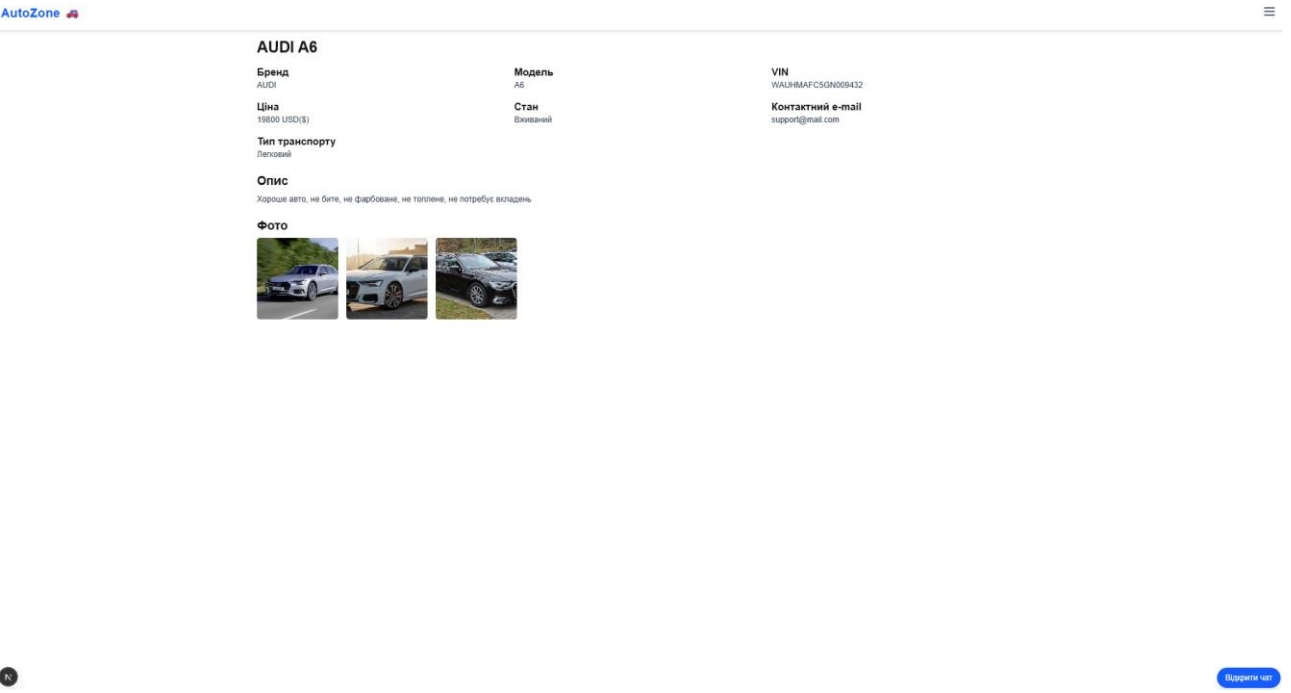


Рисунок 3.3 – Сторінка оголошення

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				44
Змн.	Арк.	№ докум.	Підпис	Дата		

На сторінці оголошень користувача, яка зображена на рис. 3.4, користувач бачить свої оголошення та може видаляти їх.

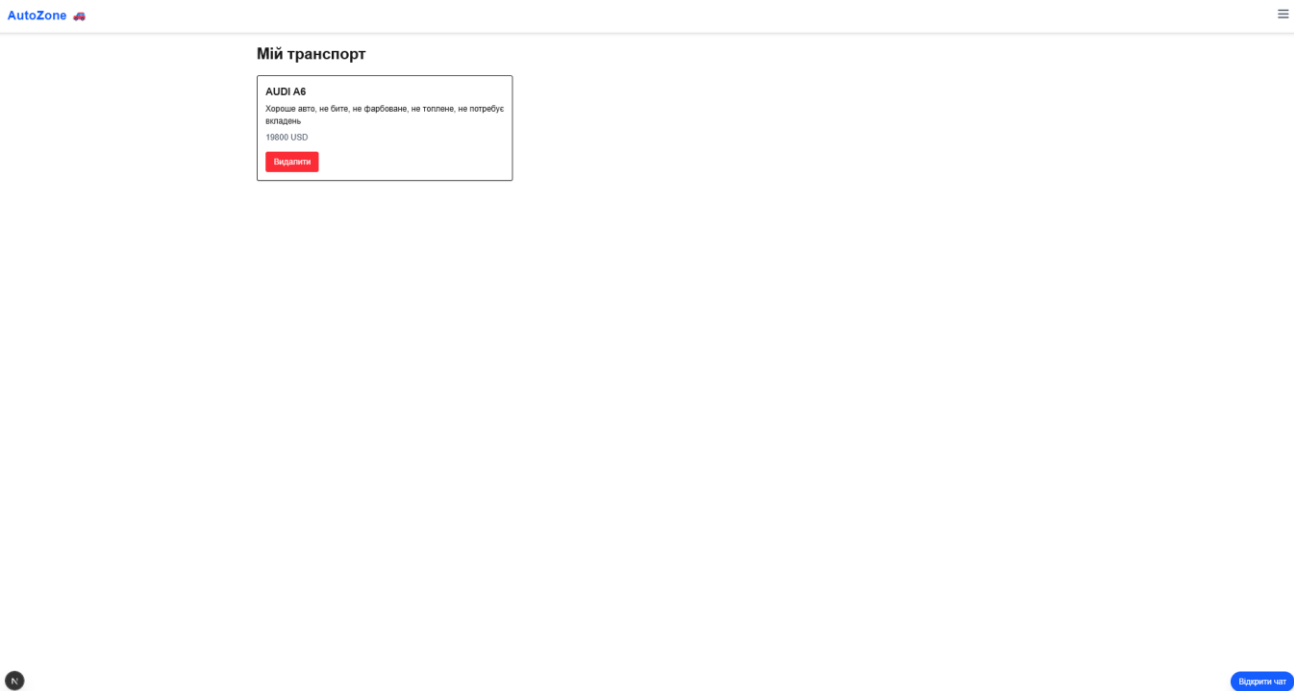


Рисунок 3.4 – Сторінка оголошень користувача

Далі у користувача є вибір, що він хоче виставити на продаж – автомобіль, грузовик чи мотоцикл. Обирає тип і потрапляє на сторінку додавання оголошення, там заповнює відповідні поля, але також є автоматичний спосіб заповнення полів за допомогою ведення VIN-кода. Це реалізується через сервіс NHTSA [13], який має можливість за допомогою API дати можливість дізнатися дані по автомобілі за VIN-кодом. Можливість додавання фотографій обмежується трьома фотографіями – через платність використаного сервісу Cloudinary [14], який зберігає у себе фотографії, а на базу даних відправляє масив посилань. Це все зображено на рис. 3.5. Також у користувача є можливість просувати оголошення, щоб вони відображалися першими, але це є платною функцією для нього.

Рисунок 3.5 – Сторінка додавання нового оголошення

Якщо користувач обрав, що він хоче просувати оголошення, то він переадресовується на сторінку сплати за оголошення. Оплата за оголошення працює через сервіс Stripe [15], тестову оплату зображено на рис. 3.6.

Рисунок 3.6 – Сторінка сплати платного оголошення

Після того як користувач додав оголошення, воно не одразу потрапляє у список оголошень, а потрапляє до адміністратора на верифікацію (рис. 3.7.).

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				46
Змн.	Арк.	№ докум.	Підпис	Дата		

Це зроблено для того, щоб адміністратор міг перевірити оголошення перед публікацією. Якщо адміністратор підтверджує публікацію, то користувачу надходить сповіщення про публікацію на його поштову адресу (рис. 3.8.). Реалізовано це через сервіс EmailJS [16], який дозволяє здійснювати розсилку поштових листів із заготовленим текстом.

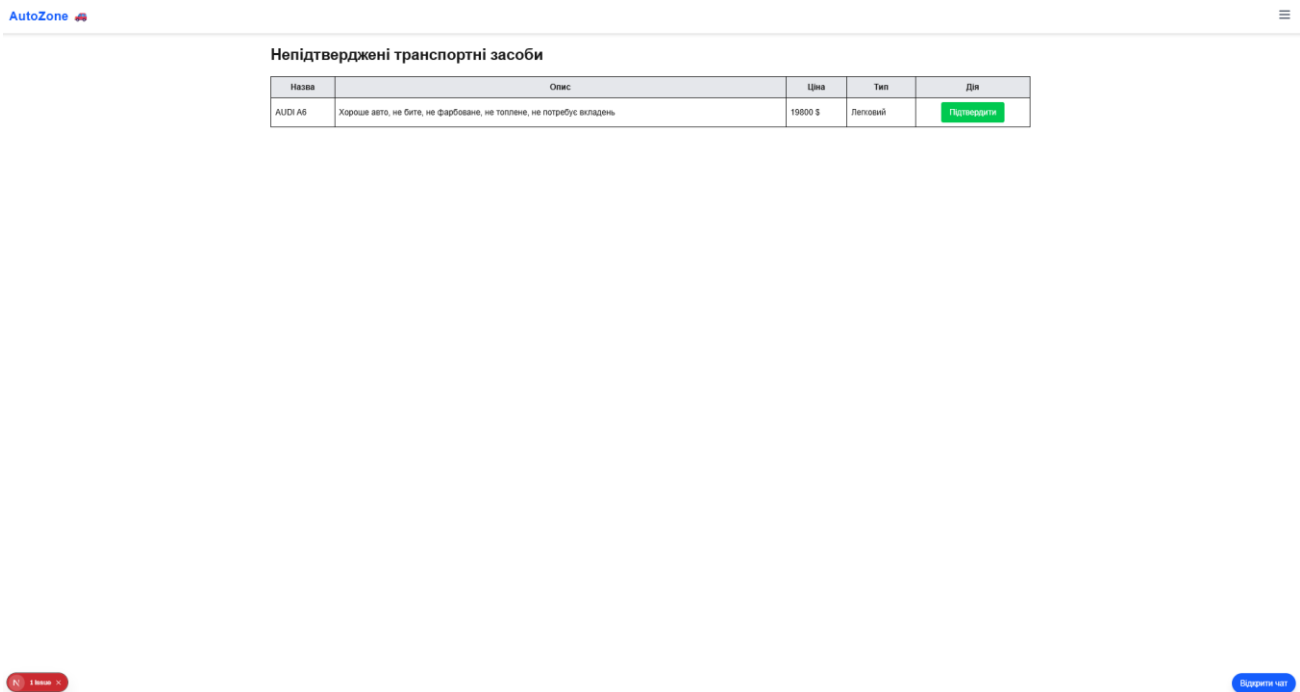


Рисунок 3.7 – Сторінка верифікації оголошення у адміністратора

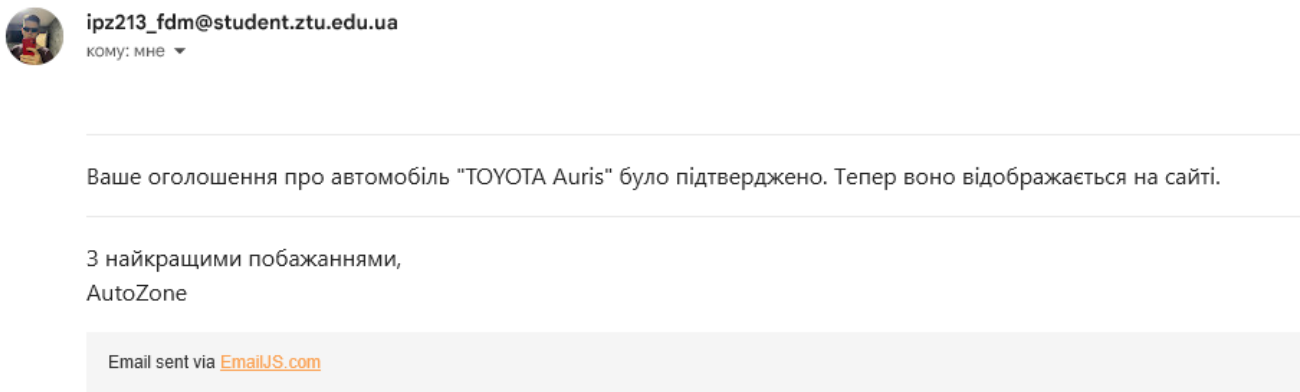


Рисунок 3.8 – Вигляд листа після верифікації адміністратором

Також розглянемо процес створення акаунту, якщо у користувача його немає, а також авторизації користувача, якщо такий існує (рис. 3.9 – рис. 3.10.)

Рисунок 3.9 – Сторінка реєстрації

Рисунок 3.10 – Сторінка входу

У користувача є можливість зв'язку з адміністрацією застосунку, якщо в нього є така потреба. Це відображено на рис. 3.11.

Рисунок 3.11– Інтерфейс чату з боку користувача

У адміністратора інтерфейс чату виглядає трохи інакше (рис. 3.12.), через те, що у адміністратора є вибір користувача, якому потрібно відповісти.

Рисунок 3.12 – Інтерфейс чату з боку адміністратора

3.3 Тестування роботи застосунку

У процесі тестування перевірялась робота ключових функцій платформи: пошук автомобілів за параметрами, перегляд детальної інформації про авто, формування замовлення, а також обробка різних користувацьких сценаріїв – наприклад, зміна фільтрів або помилкове введення даних.

Були використані такі методи тестування, як «причина-наслідок» та «аналіз граничних значень». Це дозволило змодельовати реальні ситуації, коли користувачі шукають авто за певним бюджетом, змінюють тип, марку чи рік випуску. Система мала швидко реагувати на ці запити, оновлюючи результати без збоїв.

Окрему увагу приділили методу “передбачення помилок”. Завдяки йому вдалося виявити випадки, коли користувач може випадково залишити поле порожнім або ввести некоректні дані. Такі ситуації були враховані під час налаштування захисту й підказок в інтерфейсі, щоб зробити користування платформою максимально комфортним і зрозумілим навіть для новачків.

У результаті тестування було не лише знайдено та виправлено технічні недоліки, а й значно покращено взаємодію користувачів із системою, що особливо важливо для застосунку, який має працювати швидко, точно і без збоїв.

Основні етапи тестування є такими.

Функціональне тестування

1. Реєстрація та авторизація користувача: були виконані тести, при створенні нового профілю, а також авторизація вже існуючого профілю. Перевірялась обробка даних з правильно веденими даними та обробка помилок при введенні некоректних даних.

2. Тестувалась верифікація автомобілів і надходження листів на пошти користувачів.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

3. Тестувалась правильність роботи чату застосунку, щоб користувачі мали змогу одразу отримувати відповідь від адміністрації.

Тестування інтерфейсу

1. Оцінювалась простота та зрозумілість інтерфейсу для користувачів, щоб вони могли швидко знайти потрібний їм автомобіль. Наприклад, застосувати фільтри, переглянути опис, фотографії та зв'язатися з продавцем.

2. Перевірялась адаптивність інтерфейсу на різних пристроях, щоб користувачі мали змогу користуватися застосунком з будь-якого пристрою.

Тестування безпеки

1. Тестувалась безпека облікових записів у процесі реєстрації та авторизації, щоб їх дані не потрапили до третіх осіб.

2. Також, були проведені тести на типові загрози типу XSS-атаки та SQL-ін'єкції, щоб переконатися, що застосунок не дозволяє отримати злоумисникам доступ до інформації користувачів та застосунку.

Навантажувальні тести

1. При оцінці продуктивності оцінювалось, як платформа працює при великій кількості користувачів, як швидко завантажуються сторінки, як обробляються запити до бази даних і чи не виникає затримок

2. Імітувалися екстремальні ситуації з навантаженням, щоб перевірити чи застосунок витримує надмірні запити і як швидко відновлюється після збоїв.

Ручне тестування

Ручне тестування або мануальне тестування включає в себе тестування застосунку, не використовуючи засобів автоматизації. Це простий тип тестування, але дуже ефективний.

При ручному тестуванні було перевірено декілька тестів для повного усунення проблем, які могли б виникнути у користувачів. Ці тести описані у табл. 3.1-3.6.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

Таблиця 3.1

Тест “Реєстрація користувача”

Кроки для перевірки	Ввести коректні дані у форму реєстрації
Вхідні дані	ім'я, Email, пароль
Очікуваний результат	Успішна реєстрація, перенаправлення на головну
Фактичний результат	Відповідає

Таблиця 3.2

Тест “Реєстрація з некоректними даними”

Кроки для перевірки	Ввести невалідний Email
Вхідні дані	"user@"
Очікуваний результат	Помилка валідації, користувача не створено
Фактичний результат	Відповідає

Таблиця 3.3

Тест “Авторизація”

Кроки для перевірки	Увійти з правильними обліковими даними
Вхідні дані	Існуючий Email та правильний пароль
Очікуваний результат	Успішна вхід, перенаправлення на головну
Фактичний результат	Відповідає

Таблиця 3.4

Тест “Пошук авто за маркою”

Кроки для перевірки	Ввести у пошукове поле "BMW"
Вхідні дані	"BMW"
Очікуваний результат	Список авто марки BMW
Фактичний результат	Відповідає

Таблиця 3.5

Тест “Додавання оголошення”

Кроки для перевірки	Заповнити форму, додати фото, натиснути "Додати авто"
Вхідні дані	Назва, опис, фото, ціна
Очікуваний результат	Оголошення створено зі статусом “Очікує верифікації”
Фактичний результат	Відповідає

Таблиця 3.6

Тест “Верифікація оголошення адміністратором”

Кроки для перевірки	Зайти в меню верифікації та верифікувати потрібне оголошення
Вхідні дані	Нове оголошення
Очікуваний результат	Верифіковане оголошення
Фактичний результат	Відповідає

Завдяки комплексному тестуванню вдалося значно покращити стабільність, безпеку та зручність користування сайтом. Платформа готова до запуску та відповідає очікуванням користувачів.

Тестування не зупиняється навіть після релізу – постійно збирається зворотний зв'язок, проводиться моніторинг та вдосконалення функціоналу, щоб застосунок залишався актуальним і зручним.

Висновки до третього розділу

У цьому розділі було детально описано, як саме працює застосунок для продажу автомобілів – від налаштування системи й закінчуючи тестуванням його роботи у реальних сценаріях.

Було показано, що процес встановлення та конфігурації системи не викликає складнощів і може бути реалізований навіть на базовому технічному рівні. Це дозволяє легко розгорнути платформу на сервері,

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		53

швидко налаштувати параметри безпеки, базу даних та взаємодію між модулями.

Структура інтерфейсу була спроектована з урахуванням потреб звичайного користувача. Навігація є інтуїтивно зрозумілою: від пошуку автомобіля до оформлення замовлення. Інтерфейс вийшов лаконічним та зручним, що забезпечує позитивний користувацький досвід.

Результати тестування підтвердили стабільну та надійну роботу вебзастосунку. Усі ключові функції, а саме реєстрація, пошук авто, фільтрація, чат, працюють коректно. Особливу увагу було приділено безпеці та адаптивності, і ці аспекти пройшли перевірку на високому рівні.

Загалом, реалізований застосунок відповідає сучасним вимогам до подібних платформ. Він є гнучким та готовий до впровадження в реальне середовище. Подальше вдосконалення може включати інтеграцію додаткових сервісів, однак вже на цьому етапі застосунок є повністю працездатним та корисним.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		54

ВИСНОВКИ

У ході виконання кваліфікаційної роботи бакалавра було спроектовано і реалізовано застосунок для продажу автомобілів, який дозволяє людям зручно шукати, обирати та купувати автомобілі. Від початку і до завершення проєкту увага зосереджувалась на тому, щоб зробити сервіс максимально зрозумілим для користувача та технічно надійним.

У першому розділі було досліджено аналоги існуючих рішень для продажу автомобілів, проаналізовано ринкові потреби та сформульовано технічні й функціональні вимоги до майбутнього застосунку. Також визначено технології, які найкраще підходять для реалізації проєкту.

У другому розділі описано процес проєктування структури системи, логіку роботи основних компонентів та взаємодію між ними. Спроектовано архітектуру, базу даних і реалізовано основний користувацький функціонал, з урахуванням простоти та зручності взаємодії.

У третьому розділі було реалізовано налаштування, запуск і тестування застосунку в умовах, наближених до реального використання. Перевірено працездатність усіх ключових функцій, адаптивність інтерфейсу, а також безпечність і стабільність системи загалом.

У підсумку вдалося реалізувати не просто застосунок, а повноцінну платформу, яка поєднує простоту для користувача й ефективність для бізнесу. Такий інструмент має всі шанси бути корисним для автосалонів та приватних продавців, що займаються продажем та купівлею авто.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AUTO.RIA [Електронний ресурс] – Режим доступу до ресурсу:
<https://auto.ria.com/uk/>
2. RST [Електронний ресурс] – Режим доступу до ресурсу:
<https://rst.ua/ukr/>
3. React [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/>
4. NodeJS [Електронний ресурс] – Режим доступу до ресурсу:
<https://nodejs.org/uk>
5. MongoDB [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.mongodb.com/>
6. WebStorm [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.jetbrains.com/webstorm/>
7. JavaScript [Електронний ресурс] – Режим доступу до ресурсу:
<https://developer.mozilla.org/en-US/docs/Web/JavaScript>
8. GitLab [Електронний ресурс] – Режим доступу до ресурсу:
<https://about.gitlab.com/>
9. React useState [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/reference/react/useState>
10. React useEffect [Електронний ресурс] – Режим доступу до ресурсу:
<https://react.dev/reference/react/useEffect>
11. React Redux [Електронний ресурс] – Режим доступу до ресурсу:
<https://react-redux.js.org/>
12. NewsData [Електронний ресурс] – Режим доступу до ресурсу:
<https://newsdata.io/>
13. NHTSA [Електронний ресурс] – Режим доступу до ресурсу:
<https://vpic.nhtsa.dot.gov/>
14. Cloudinary [Електронний ресурс] – Режим доступу до ресурсу:
<https://cloudinary.com/>

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		56

15. Stripe [Електронний ресурс] – Режим доступу до ресурсу:
<https://stripe.com/>

16. EmailJS [Електронний ресурс] – Режим доступу до ресурсу:
<https://www.emailjs.com/>

17. Creately [Електронний ресурс] – Режим доступу до ресурсу:
<https://creately.com/>

18. Next.js [Електронний ресурс] – Режим доступу до ресурсу:
<https://nextjs.org/>

19. Типи тестування [Електронний ресурс] – Режим доступу до ресурсу:
<https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/>

20. Федорович Д.М., Кушнір Н.О., Лисогор Ю.І. Застосунок для продажу автомобілів. Тези доповідей Всеукраїнської науково-практичної on-line конференції здобувачів вищої освіти і молодих учених, присвяченої Дню науки, 12 травня 2025 року. Подані та прийняті до опублікування.

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				57
Змн.	Арк.	№ докум.	Підпис	Дата		

ДОДАТКИ

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				58
Змн.	Арк.	№ докум.	Підпис	Дата		

User model

```
const { Schema, model } = require('mongoose')
```

```
const UserSchema = new Schema({
  email: {
    type: String,
    required: true,
    unique: true
  },
  name: {
    type: String,
    required: true,
  },
  password: {
    type: String,
    required: true,
  },
  access: {
    type: Number,
    default: 0
  }
})
```

```
module.exports = model('Users', UserSchema)
```

carDetails model

```
const { Schema, model } = require('mongoose')
```

```
const CarsSchema = new Schema({
  typeOfVeh: {
    type: String,
    required: true
  },
  name: {
    type: String,
    required: true,
  },
  details: {
    type: String,
    required: true,
  },
  images: {
    type: [String],
    required: true,
  },
  vin: {
    type: String,
    required: false,
  },
  brand: {
```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				59
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    type: String,
    required: false,
  },
  model: {
    type: String,
    required: false,
  },
  owner: {
    type: String,
    required: false,
  },
  statusOfUse: {
    type: String,
    required: true,
  },
  statusOfUse: {
    type: String,
    required: true,
  },
  price: {
    type: String,
    required: true,
  },
  statusOfUse: {
    type: String,
    required: true,
  },
  statusOfVerification: {
    type: Boolean,
    required: true,
  },
  payed: {
    type: Boolean,
    required: false,
  },
  dateOfPlacement: {
    type: String,
    required: false,
  },
  dateOfEnding: {
    type: String,
    required: false
  }
})

```

```

module.exports = model('Cars', CarsSchema)

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				60
Змн.	Арк.	№ докум.	Підпис	Дата		

```

car router
require('dotenv').config();
const { Router } = require("express");
const cloudinary = require("cloudinary").v2;
const toStream = require("buffer-to-stream");
const Cars = require("../models/carDetails");
const router = Router();

cloudinary.config({
  cloud_name: process.env.CLOUD_NAME,
  api_key: process.env.API_KEY,
  api_secret: process.env.API_SECRET,
});

// Завантаження в Cloudinary
router.post("/upload-images", async (req, res) => {
  try {
    if (!req.files || !req.files.image) {
      return res.status(400).json({ error: "Image file is required." });
    }

    const file = req.files.image;
    const uploadPromise = new Promise((resolve, reject) => {
      const upload = cloudinary.uploader.upload_stream(
        { resource_type: "image", folder: "cars" },
        (error, result) => {
          if (error) return reject(error);
          resolve(result.secure_url);
        }
      );
    });
    toStream(file.data).pipe(upload);
  });

  const imageUrl = await uploadPromise;
  console.log(imageUrl);
  res.json({ imageUrl });
} catch (error) {
  console.error(error);
  res.status(500).json({ error: "Server error" });
}
});

router.get("/get-user-cars", async (req, res) => {
  const email = req.query.email;
  try {
    const cars = await CarModel.find({ ownerEmail: email });
    res.json(cars);
  } catch (err) {
    res.status(500).json({ error: "Server error" });
  }
}

```

```

// Додання авто
router.post("/add-car", async (req, res) => {
  try {
    const {
      name,
      details,
      statusOfUse,
      price,
      payed,
      images,
      brand,
      model,
      vin,
      owner
    } = req.body;
    if (!images || !Array.isArray(images) || images.length === 0) {
      return res.status(400).json({ error: "At least one image URL is required." });
    }

    const dateOfPlacement = new Date().toISOString();
    const dateOfEnding = new Date();
    dateOfEnding.setDate(dateOfEnding.getDate() + 30);

    const newCar = new Cars({
      typeOfVeh: "car",
      name,
      details,
      statusOfUse,
      brand,
      model,
      vin,
      owner,
      price,
      images,
      statusOfVerification: false,
      payed: payed || false,
      dateOfPlacement,
      dateOfEnding: dateOfEnding.toISOString(),
    });

    const savedCar = await newCar.save();
    res.status(201).json(savedCar);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: "Server error" });
  }
});

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		62

```

// Список авто
router.get("/get-cars", async (req, res) => {
  try {
    const cars = await Cars.find();
    res.json(cars);
  } catch (error) {
    res.status(500).json({ error: "Server error" });
  }
});

// Авто по ID
router.get("/get-car/:id", async (req, res) => {
  try {
    const car = await Cars.findById(req.params.id);
    if (!car) {
      return res.status(404).json({ error: "Car not found" });
    }
    res.json(car);
  } catch (error) {
    res.status(500).json({ error: "Server error" });
  }
});

// Видалити авто
router.delete("/delete-car/:id", async (req, res) => {
  try {
    const deletedCar = await Cars.findByIdAndDelete(req.params.id);
    if (!deletedCar) {
      return res.status(404).json({ error: "Car not found" });
    }
    res.json({ message: "Car deleted", deletedCar });
  } catch (error) {
    res.status(500).json({ error: "Server error" });
  }
});

// Оновлення машини за ID
router.put("/update-car/:id", async (req, res) => {
  try {
    const { id } = req.params;
    const { statusOfVerification } = req.body;

    const updatedCar = await Cars.findByIdAndUpdate(
      id,
      { statusOfVerification },
      { new: true }
    );

    if (!updatedCar) {
      return res.status(404).json({ error: "Car not found" });
    }
  }
});

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		63

```

    }

    res.json(updatedCar);
  } catch (error) {
    console.error(error);
    res.status(500).json({ error: "Server error" });
  }
});

module.exports = router;

user router
const { Router } = require("express");
const bcrypt = require("bcryptjs");
const users = require("../models/user");
const router = Router();
const cors = require("cors");

router.use(cors());

// тест роут
router.get("/", async (req, res) => {
  res.json({ test: "test" });
});

// Регістрацію звичайного юзера
router.post("/sign-up", async (req, res) => {
  try {
    console.log(req.body)
    const { email, name, password } = req.body;

    const alreadyUser = await users.findOne({ email });
    if (alreadyUser) {
      return res.status(400).json({ error: "Такий користувач вже існує" });
    }

    const hashedPassword = await bcrypt.hash(password, 10);
    const user = new users({ email, name, password: hashedPassword });

    await user.save();
    res.status(201).json({ message: "User registered successfully", user });
  } catch (error) {
    res.status(500).json({ error: "Internal Server Error" });
  }
});

// Регістрація адміна
router.post("/sign-up-admin", async (req, res) => {
  try {
    const { email, name, password, access } = req.body;

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		64

```

const alreadyUser = await users.findOne({ email });
if (alreadyUser) {
  return res.status(400).json({ error: "Такий користувач вже існує" });
}

const hashedPassword = await bcrypt.hash(password, 10);
const user = new users({ email, name, password: hashedPassword, access });

await user.save();
res.status(201).json({ message: "Admin created successfully", user });
} catch (error) {
  res.status(500).json({ error: "Internal Server Error" });
}
});

// Авторизація
router.post("/sign-in", async (req, res) => {
  try {
    const { email, password } = req.body;

    const user = await users.findOne({ email });
    if (!user) {
      return res.status(401).json({ error: "Authentication failed" });
    }

    const isPasswordMatch = await bcrypt.compare(password, user.password);
    if (!isPasswordMatch) {
      return res.status(401).json({ error: "Authentication failed" });
    }

    res.status(200).json({ message: "Authentication successful", user });
  } catch (error) {
    res.status(500).json({ error: "Internal Server Error" });
  }
});

module.exports = router;

chat
const express = require("express");
const http = require("http");
const WebSocket = require("ws");

const app = express();
const server = http.createServer(app);
const wss = new WebSocket.Server({ server });

const clients = new Map(); // socket -> { email }

wss.on("connection", (ws) => {

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				65
Змн.	Арк.	№ докум.	Підпис	Дата		

```

console.log("Клієнт в чаті");

ws.on("message", (data) => {
  try {
    const parsed = JSON.parse(data);
    const { type, email, message, to } = parsed;

    if (type === "auth") {
      clients.set(ws, { email });
      console.log(`Користувач аутентифікован: ${email}`);
    }

    if (type === "message") {
      for (let [client, info] of clients.entries()) {
        if (info.email === to && client.readyState === WebSocket.OPEN) {
          client.send(JSON.stringify({
            from: email,
            message,
            type: "message"
          }));
        }
      }
    }
  } catch (err) {
    console.error("Помилка парсинга:", err);
  }
});

ws.on("close", () => {
  clients.delete(ws);
  console.log("Клієнт відключився");
});

server.listen(3001, () => {
  console.log("WebSocket сервер працює на порту 3001");
});

cars page
"use client";

import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { AppDispatch, RootState } from "@redux/store";
import { fetchCars } from "@redux/slices/carsSlice";
import FilterSearch from "@components/FilterSearch";
import CarCard from "@components/CarCard";

const CarsPage = () => {
  const dispatch = useDispatch<AppDispatch>();
  const { cars, status } = useSelector((state: RootState) => state.cars);

```

```

const [filteredCars, setFilteredCars] = useState(cars);

useEffect(() => {
  dispatch(fetchCars());
}, [dispatch]);

useEffect(() => {
  setFilteredCars(cars);
}, [cars]);

const handleFilterChange = (filters: {
  name: string;
  statusOfUse: string;
  brand: string;
  minPrice: string;
  maxPrice: string;
}) => {
  const filtered = cars.filter((car) => {
    return (
      (filters.name ? car.name.toLowerCase().includes(filters.name.toLowerCase()) : true) &&
      (filters.statusOfUse ? car.statusOfUse === filters.statusOfUse : true) &&
      (filters.brand ? car.details.includes(filters.brand) : true) &&
      (filters.minPrice ? car.price >= Number(filters.minPrice) : true) &&
      (filters.maxPrice ? car.price <= Number(filters.maxPrice) : true)
    );
  });
  setFilteredCars(filtered);
};

return (
  <div className="container mx-auto p-4">
    <h1 className="text-2xl font-bold mb-4">Список автомобілей</h1>

    <FilterSearch onFilterChange={handleFilterChange} />

    {status === "loading" && <p>Завантаження...</p>}
    {status === "failed" && <p>Ошибка завантаження</p>}

    <div className="grid grid-cols-1 md:grid-cols-2 lg:grid-cols-3 gap-4 mt-4">
      {filteredCars.map((car) => (
        <CarCard key={car._id} car={car} />
      ))}
    </div>
  </div>
);
};

export default CarsPage;

Add car page

```

```

"use client";
import { useState, useEffect } from "react";
import { useDispatch } from "react-redux";
import { AppDispatch } from "@redux/store";
import { addCar } from "@redux/slices/carsSlice";
import { useRouter } from "next/navigation";
import api from "@services/api";
import CarBrandSelect from "@components/CarBrandSelect";
import { CloudCog } from "lucide-react";

export default function AddCarPage() {
  const dispatch = useDispatch<AppDispatch>();
  const router = useRouter();

  const [userEmail, setUserEmail] = useState<string | null>(null);
  const [vinError, setVinError] = useState<string | null>(null);
  const [vinCheckLoading, setVinCheckLoading] = useState(false);
  const [vinDetectedBrand, setVinDetectedBrand] = useState(false);
  const [imagesUploading, setImagesUploading] = useState(false);

  const [formData, setFormData] = useState({
    name: "",
    brand: "",
    model: "",
    vin: "",
    details: "",
    statusOfUse: "new",
    price: "",
    images: [] as string[],
    payed: false,
    owner: "",
  });

  useEffect(() => {
    const storedUser = localStorage.getItem("user");
    if (!storedUser) {
      router.push("/login");
    } else {
      const user = JSON.parse(storedUser);
      setUserEmail(user.user.email);
      setFormData((prev) => ({ ...prev, owner: user.user.email }));
    }
  }, [router]);

  const handleChange = (
    e: React.ChangeEvent<HTMLInputElement | HTMLTextAreaElement |
HTMLSelectElement>
  ) => {
    setFormData({ ...formData, [e.target.name]: e.target.value });
  };

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				68
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const handleBrandChange = (value: string) => {
  if (!vinDetectedBrand) {
    setFormData((prev) => ({ ...prev, brand: value }));
  }
};

const handleCheckboxChange = (e: React.ChangeEvent<HTMLInputElement>) => {
  setFormData({ ...formData, payed: e.target.checked });
};

const handleFileChange = async (e: React.ChangeEvent<HTMLInputElement>) => {
  if (e.target.files) {
    const files = Array.from(e.target.files).slice(0, 3);
    setImagesUploading(true);
    const uploadedUrls = await Promise.all(files.map(uploadImageAsBinary));
    setImagesUploading(false);

    setFormData((prev) => ({
      ...prev,
      images: [...prev.images, ...uploadedUrls.filter(Boolean)],
    }));
  }
};

const uploadImageAsBinary = async (file: File): Promise<string> => {
  const reader = new FileReader();
  const binaryData = await new Promise<ArrayBuffer>((resolve, reject) => {
    reader.onloadend = () => resolve(reader.result as ArrayBuffer);
    reader.onerror = reject;
    reader.readAsArrayBuffer(file);
  });

  const formData = new FormData();
  formData.append("image", new Blob([binaryData]), file.name);

  try {
    const response = await api.post("/upload-images", formData, {
      headers: { "Content-Type": "multipart/form-data" },
    });
    return response.data.imageUrl;
  } catch (error) {
    console.error("Помилка завантаження фотографії", error);
    return "";
  }
};

const handleVinBlur = async () => {
  const vin = formData.vin.trim().toUpperCase();

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		69

```

setFormData((prev) => ({ ...prev, vin }));

if (vin.length !== 17) {
  setVinError("VIN повинен містити 17 символів");
  return;
}

setVinError(null);
setVinCheckLoading(true);
try {
  const response = await
fetch(`https://vpic.nhtsa.dot.gov/api/vehicles/DecodeVin/${vin}?format=json`);
  const data = await response.json();
  const make = data.Results.find((r: any) => r.Variable === "Make")?.Value;
  const model = data.Results.find((r: any) => r.Variable === "Model")?.Value;

  if (make && !formData.brand) {
    setFormData((prev) => ({ ...prev, brand: make }));
    setVinDetectedBrand(true);
  }

  if (model && !formData.model) {
    setFormData((prev) => ({ ...prev, model }));
    console.log(model)
  }
} catch (err) {
  console.error("Помилка VIN-перевірки", err);
  setVinError("Не вдалося перевірити VIN");
} finally {
  setVinCheckLoading(false);
}
};

const handleSubmit = async (e: React.FormEvent) => {
  e.preventDefault();

  if (vinError) {
    alert("Перевірте VIN-код перед відправкою.");
    return;
  }

  if (formData.images.some((url) => url === "")) {
    alert("Помилка завантаження фотографії");
    return;
  }

  const now = new Date();
  const payload = {
    ...formData,
    name: `${formData.brand} ${formData.model}`,
  }

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		70

```

    dateOfPlacement: now.toISOString(),
    dateOfEnding: new Date(now.setMonth(now.getMonth() + 1)).toISOString(),
  };

  if (formData.payed) {
    localStorage.setItem("pendingCar", JSON.stringify({ ...payload, payed: true }));
    router.push("/payment");
  } else {
    const result = await dispatch(addCar(payload));
    if (addCar.fulfilled.match(result)) {
      router.push("/");
    }
  }
};

if (!userEmail) {
  return <p className="text-center mt-10 text-gray-600">Перенаправлення...</p>;
}

return (
  <div className="flex items-center justify-center min-h-screen bg-gray-100">
    <div className="bg-white p-8 rounded-lg shadow-lg w-96">
      <h2 className="text-2xl font-bold text-center mb-6">Додати авто</h2>

      <form onSubmit={handleSubmit} className="space-y-4">
        <div>
          <label className="block text-gray-700">VIN</label>
          <input
            type="text"
            name="vin"
            value={formData.vin}
            onChange={(e) => {
              const vin = e.target.value.toUpperCase();
              setFormData((prev) => ({ ...prev, vin }));
            }}
            onBlur={handleVinBlur}
            className={`w-full px-4 py-2 border rounded-lg ${vinError ? "border-red-500" : ""}`}
            placeholder="Наприклад: WAUZZZ8K9AA123456"
          />
          {vinCheckLoading && <p className="text-sm text-blue-500 mt-1">Перевірка
VIN...</p>}
          {vinError && <p className="text-sm text-red-500 mt-1">{vinError}</p>}
        </div>

        <div>
          <label className="block text-gray-700">Марка</label>
          <CarBrandSelect value={formData.brand} onChange={handleBrandChange}
disabled={vinDetectedBrand} />
        </div>
      </form>
    </div>
  </div>
);

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		71

```

<div>
  <label className="block text-gray-700">Модель</label>
  <input
    type="text"
    name="model"
    value={formData.model}
    onChange={handleChange}
    className={`w-full px-4 py-2 border rounded-lg ${formData.model.trim() === "" &&
formData.vin && vinDetectedBrand ? "border-red-500" : ""}
  `}
  />
  {formData.model.trim() === "" && formData.vin && vinDetectedBrand && (
    <p className="text-sm text-red-500 mt-1">Модель не визначено</p>
  )}
</div>

<div>
  <label className="block text-gray-700">Опис</label>
  <textarea
    name="details"
    value={formData.details}
    onChange={handleChange}
    className="w-full px-4 py-2 border rounded-lg"
  />
</div>

<div>
  <label className="block text-gray-700">Статус</label>
  <select
    name="statusOfUse"
    value={formData.statusOfUse}
    onChange={handleChange}
    className="w-full px-4 py-2 border rounded-lg"
  >
    <option value="new">Новий</option>
    <option value="used">Б/у</option>
  </select>
</div>

<div>
  <label className="block text-gray-700">Ціна у $</label>
  <input
    type="text"
    name="price"
    value={formData.price}
    onChange={handleChange}
    className="w-full px-4 py-2 border rounded-lg"
  />
</div>

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		72

```

<div>
  <label className="block text-gray-700">Фото (до 3)</label>
  <input
    type="file"
    accept="image/*"
    multiple
    onChange={handleFileChange}
    className="w-full px-4 py-2 border rounded-lg"
  />
  <div className="mt-2 flex space-x-2">
    {formData.images.map((url, index) => (
      <img key={index} src={url} alt={`car-${index}`} className="w-20 h-20 object-cover rounded" />
    ))}
  </div>
</div>

<div className="flex items-center">
  <input
    type="checkbox"
    checked={formData.payed}
    onChange={handleCheckboxChange}
    className="mr-2"
  />
  <label className="text-gray-700">Просувати оголошення? ($5)</label>
</div>

<button
  type="submit"
  disabled={imagesUploading}
  className={`w-full py-2 rounded-lg transition ${imagesUploading ? "bg-gray-400 cursor-not-allowed" : "bg-blue-500 hover:bg-blue-600 text-white"}`}
  >
  {imagesUploading ? "Завантаження фото..." : "Додати авто"}
</button>

</form>
</div>
</div>
);
}

```

Payment page

// app/payment/page.tsx

"use client";

```

import { useEffect, useState } from "react";
import { loadStripe } from "@stripe/stripe-js";
import { Elements, useStripe, useElements, CardElement } from "@stripe/react-stripe-js";
import { useRouter } from "next/navigation";
import { useDispatch } from "react-redux";

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		73

```

import { AppDispatch } from "@redux/store";
import { addCar } from "@redux/slices/carsSlice";

const stripePromise = loadStripe(process.env.NEXT_PUBLIC_STRIPE_PUBLIC_KEY!);

export default function PaymentPage() {
  return (
    <Elements stripe={stripePromise}>
      <CheckoutForm />
    </Elements>
  );
}

function CheckoutForm() {
  const stripe = useStripe();
  const elements = useElements();
  const router = useRouter();
  const [clientSecret, setClientSecret] = useState<string | null>(null);

  const dispatch = useDispatch<AppDispatch>();

  useEffect(() => {
    const fetchClientSecret = async () => {
      try {
        const response = await fetch("/api/create-payment-intent", {
          method: "POST",
          body: JSON.stringify({ amount: 500 }),
          headers: {
            "Content-Type": "application/json",
          },
        });
        const data = await response.json();
        setClientSecret(data.clientSecret);
      } catch (error) {
        console.error("Ошибка получения client_secret", error);
      }
    };

    fetchClientSecret();
  }, []);

  const handlePayment = async () => {
    if (!stripe || !elements || !clientSecret) return;

    const { error, paymentIntent } = await stripe.confirmCardPayment(clientSecret, {
      payment_method: {
        card: elements.getElement(CardElement)!,
      },
    });
  };
}

```

```

if (!error && paymentIntent?.status === "succeeded") {
  alert("Оплата успішна!");

  const rawData = localStorage.getItem("pendingCar");
  if (!rawData) {
    alert("Не знайдено дані автомобіля");
    return;
  }

  const carData = JSON.parse(rawData);

  const now = new Date();
  const payload = {
    ...carData,
    payed: true,
    dateOfPlacement: now.toISOString(),
    dateOfEnding: new Date(now.setMonth(now.getMonth() + 1)).toISOString(),
  };

  try {
    const result = await dispatch(addCar(payload));
    if (addCar.fulfilled.match(result)) {
      localStorage.removeItem("pendingCar");
      router.push("/");
    } else {
      alert("Помилка при додаванні авто");
    }
  } catch (err) {
    console.error("Помилка при диспатчі addCar", err);
  }
  } else if (error) {
    console.error("Помилка:", error.message);
    alert("Помилка: " + error.message);
  }
};

return (
  <div className="p-8 bg-white rounded shadow-lg max-w-md mx-auto mt-10">
    <h2 className="text-2xl font-bold">Сплатити оголошення</h2>
    <p>Сума: $5</p>
    <CardElement className="border p-2 mt-4" />
    <button
      onClick={handlePayment}
      className="mt-4 w-full bg-blue-500 text-white py-2 rounded-lg"
      disabled={!clientSecret}
    >
      Сплатити
    </button>
  </div>

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		75

```
);
}
```

Profile page

"use client";

```
import { useEffect, useState } from "react";
import { useDispatch, useSelector } from "react-redux";
import { fetchCars, deleteCar } from "@redux/slices/carsSlice";
import { fetchMotos, deleteMoto } from "@redux/slices/motosSlice";
import { fetchVans, deleteVan } from "@redux/slices/vansSlice";
import { RootState } from "@redux/store";
import { AppDispatch } from "@redux/store";
```

```
interface VehicleItem {
  _id: string;
  name?: string;
  details?: string;
  price?: number;
  owner: string;
  typeOfVeh?: "car" | "moto" | "van";
}
```

```
const ProfilePage = () => {
  const dispatch = useDispatch<AppDispatch>();

  const cars = useSelector((state: RootState) => state.cars.cars);
  const motos = useSelector((state: RootState) => state.motos.motos);
  const vans = useSelector((state: RootState) => state.vans.vans);

  const carsStatus = useSelector((state: RootState) => state.cars.status);
  const motosStatus = useSelector((state: RootState) => state.motos.status);
  const vansStatus = useSelector((state: RootState) => state.vans.status);

  const [filtered, setFiltered] = useState<VehicleItem[]>([]);
  const [isModalOpen, setIsModalOpen] = useState(false);
  const [vehicleToDelete, setVehicleToDelete] = useState<VehicleItem | null>(null);

  useEffect(() => {
    dispatch(fetchCars());
    dispatch(fetchMotos());
    dispatch(fetchVans());
  }, [dispatch]);

  useEffect(() => {
    const storedUser = localStorage.getItem("user");
    if (storedUser) {
      const user = JSON.parse(storedUser);
      const email = user.user.email;

      const addType = (items: any[], type: "car" | "moto" | "van") =>
        items.map((item) => ({ ...item, typeOfVeh: type }));
```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		76

```

const allWithType: VehicleItem[] = [
  ...addType(cars, "car"),
  ...addType(motos, "moto"),
  ...addType(vans, "van"),
];

const owned = allWithType.filter((item) => item.owner === email);
setFiltered(owned);
}
}, [cars, motos, vans]);

const openModal = (vehicle: VehicleItem) => {
  setVehicleToDelete(vehicle);
  setIsModalOpen(true);
};

const closeModal = () => {
  setIsModalOpen(false);
  setVehicleToDelete(null);
};

const confirmDelete = () => {
  if (vehicleToDelete) {
    const { _id, typeOfVeh } = vehicleToDelete;
    switch (typeOfVeh) {
      case "car":
        dispatch(deleteCar(_id));
        break;
      case "moto":
        dispatch(deleteMoto(_id));
        break;
      case "van":
        dispatch(deleteVan(_id));
        break;
      default:
        console.warn("Невідомий тип транспорту:", typeOfVeh);
    }
  }
  closeModal();
};

const isLoading =
  carsStatus === "loading" || motosStatus === "loading" || vansStatus === "loading";

return (
  <div className="container mx-auto py-6">
    <h1 className="text-3xl font-bold mb-6">Мій транспорт</h1>
    {isLoading && <p>Завантаження...</p>}
    {filtered.length === 0 && !isLoading && <p>Немає доданого транспорту</p>}
    <div className="grid grid-cols-1 sm:grid-cols-2 lg:grid-cols-3 gap-4">

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    {filtered.map((item) => (
      <div key={item._id} className="border p-4 rounded shadow">
        <h2 className="text-xl font-semibold mb-2">{item.name || "Без назви"}</h2>
        <p>{item.details}</p>
        <p className="text-gray-600 mt-2">{item.price} USD</p>
        <button
          onClick={() => openModal(item)}
          className="mt-4 px-4 py-2 bg-red-500 hover:bg-red-600 text-white rounded"
        >
          Видалити
        </button>
      </div>
    )))
  </div>

  {isModalOpen && vehicleToDelete && (
    <div className="fixed inset-0 bg-black bg-opacity-50 flex items-center justify-center z-50">
      <div className="bg-white p-6 rounded-lg shadow-lg max-w-sm w-full">
        <h2 className="text-lg font-semibold mb-4">Підтвердіть видалення</h2>
        <p className="mb-6">Ви впевнені, що бажаєте видалити транспорт?</p>
        <div className="flex justify-end gap-3">
          <button
            onClick={closeModal}
            className="px-4 py-2 bg-gray-300 hover:bg-gray-400 rounded"
          >
            Відміна
          </button>
          <button
            onClick={confirmDelete}
            className="px-4 py-2 bg-red-500 hover:bg-red-600 text-white rounded"
          >
            Видалити
          </button>
        </div>
      </div>
    </div>
  )}
</div>
);
};

export default ProfilePage;

```

		Федорович Д.М.			ІПЗ.КР.Б-121-25-ПЗ	Арк.
		Кушнір Н.О.				
Змн.	Арк.	№ докум.	Підпис	Дата		78