

БАКАЛАВРСЬКА

РОБОТА

БР.ІП-05.00.00.000 ІПЗ

Група ІП-22-4

Левицький Богдан

Івано-Франківський національний технічний університет нафти і газу

Факультет інформаційних технологій

Кафедра інженерії програмного забезпечення

Левицький Богдан Любомирович

(прізвище, ім'я, по батькові)

УДК 004.942

(індекс)

БАКАЛАВРСЬКА РОБОТА

Проектування та розробка веб-застосунку для автоматизації продажу

автомобільних запчастин

(назва роботи)

Інженерія програмного забезпечення

(назва освітньої програми)

121 - Інженерія програмного забезпечення

(шифр і назва спеціальності)

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

Здобувач освітнього ступеня Левицький Б.Л.

(підпис, ініціали та прізвище здобувача)

Науковий керівник Дмитрик Тарас Богданович асистент

(підпис, прізвище, ім'я, по батькові, науковий ступінь, вчене звання керівника)

Допущено до захисту

Завідувач кафедри

доц. Бандура В.В.

(посада) (підпис) (дата) (ініціали та прізвище)

Івано-Франківськ -2026 рік

Івано-Франківський національний технічний університет нафти і газу
Інститут, факультет інформаційних технологій
Кафедра інженерії програмного забезпечення
Ступінь вищої освіти бакалавр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:

Зав. кафедрою ІІЗ

доц. В.В. Бандура

“ ” 2026 р.

ЗАВДАННЯ

НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТОВІ

Левицькому Богдану Любомировичу

(прізвище, ім'я, по-батькові)

1. Тема проекту (роботи) “Проектування та розробка веб-застосунку для автоматизації продажу автомобільних запчастин”

керівник проекту (роботи) Дмитрик Тарас Богданович асистент

затверджені наказом закладу вищої освіти від “ 21 ” квітня 2026 р. № 179/7

2. Строк подання студентом проекту (роботи) червня 2026 р.

3. Вихідні дані до проекту (роботи) розроблення вебзастосунку для онлайн-продажу автозапчастин, вимоги до системи та матеріали аналізу предметної області.

4. Зміст розрахунково - пояснювальної записки (перелік питань, які потрібно розробити)

1. Аналіз предметної області та технологій розроблення вебзастосунку.

2. Формування вимог і користувацьких сценаріїв системи.

3. Проектування архітектури системи та бази даних.

4. Реалізація функціоналу й інтерфейсу вебзастосунку.

5. Тестування програмного продукту.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1. Узагальнений процес взаємодії (рис. 1.1, ст. 20)

2. Концептуальна схема взаємодії компонентів (рис. 1.2, ст. 25)

3. Діаграма варіантів використання (рис. 2.1, ст. 30)

4. Архітектура вебзастосунку (рис. 3.1, ст. 39)

5. Логічна модель бази даних (рис. 3.2, ст. 43)

6. Діаграма послідовності оформлення замовлення (рис. 3.3, ст. 45)

1. Консультанти розділів проекту (роботи)

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

2. Дата видачі завдання _____

Керівник

_____ (підпис)

Дмитрик Т.Б.

_____ (розшифровка підпису)

Завдання прийняв до
виконання

_____ (підпис)

Левицький Б.Л

_____ (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту	Примітка
1	Опрацювання теоретичних матеріалів з теми роботи	16.01.2026	виконано
2	Аналіз предметної області та визначення вимог до системи	29.01.2026	виконано
3	Проектування архітектури вебзастосунку та бази даних	23.02.2026	виконано
4	Розроблення користувацької та серверної частини системи	07.03.2026	виконано
5	Тестування функціональності та усунення виявлених помилок	23.03.2026	виконано
6	Підготовка та оформлення бакалаврської роботи	27.04.2026	виконано

Студент – дипломник

_____ (підпис)

Левицький Б.Л.

_____ (розшифровка підпису)

Керівник роботи

_____ (підпис)

Дмитрик Т.Б.

_____ (розшифровка підпису)

АНОТАЦІЯ

Бакалаврська робота містить 73 сторінок, 19 рисунків, список використаних джерел із 38 найменувань.

Метою роботи є розробка вебзастосунку для онлайн-продажу автозапчастин, що забезпечує перегляд каталогу, пошук товарів, роботу з кошиком, оформлення замовлень, профілем користувача та адміністративною панеллю.

Об'єкт дослідження: процеси розробки та функціонування вебзастосунків електронної комерції.

Предмет дослідження: методи та технології розробки клієнт-серверного вебзастосунку з каталогом товарів, базою даних, бізнес-логікою та зовнішніми інтеграціями.

Результати дослідження: розроблено вебзастосунок Auto Parts на базі React, Vite, Node.js, Express.js і PostgreSQL, що забезпечує роботу каталогу автозапчастин, кошика, замовлень, профілю користувача, гаража автомобілів, доставки та адміністративної панелі.

У першому розділі проведено аналіз предметної області та програмних рішень.

У другому розділі визначено вимоги й постановку задачі.

У третьому розділі спроектовано архітектуру, базу даних та API-взаємодію.

У четвертому розділі реалізовано користувацьку частину, серверну логіку, адміністративну панель і додаткові функції.

У п'ятому розділі проведено тестування вебзастосунку.

Висновок: розроблений вебзастосунок відповідає поставленій меті та забезпечує основний функціонал інтернет-магазину автозапчастин.

КЛЮЧОВІ СЛОВА: ЕЛЕКТРОННА КОМЕРЦІЯ, ВЕБЗАСТОСУНОК, ІНТЕРНЕТ-МАГАЗИН, АВТОЗАПЧАСТИН, REACT, NODE.JS, EXPRESS.JS, POSTGRESQL, REST API.

ANNOTATION

The bachelor's thesis consists of 73 pages, 19 figures, and a bibliography of 38 references.

The aim of the thesis is to develop a web application for online auto parts sales that provides catalog browsing, product search, shopping cart functionality, order placement, user profile management, and an administrative panel.

Research object: the processes of developing and operating e-commerce web applications.

Research subject: methods and technologies for developing a client-server web application with a product catalog, database, business logic, and external integrations.

Research results: the Auto Parts web application was developed using React, Vite, Node.js, Express.js, and PostgreSQL. The system provides the operation of the auto parts catalog, shopping cart, orders, user profile, vehicle garage, delivery, and administrative panel.

The first chapter analyzes the subject area and existing software solutions.

The second chapter defines the requirements and problem statement.

The third chapter presents the architecture, database, and API interaction.

The fourth chapter describes the implementation of the user part, server logic, administrative panel, and additional functions.

The fifth chapter presents the testing of the web application.

Conclusion: the developed web application meets the stated aim and provides the main functionality of an online auto parts store.

KEYWORDS: E-COMMERCE, WEB APPLICATION, ONLINE STORE, AUTO PARTS, REACT, NODE.JS, EXPRESS.JS, POSTGRESQL, REST API.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ.....	8
ВСТУП.....	9
РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ РОЗРОБКИОНЛАЙН-МАГАЗИНУ АВТОЗАПЧАСТИН.....	11
1.1. Аналіз сучасного стану проблематики.....	11
1.2. Основні поняття та нормативно-технічні аспекти предметної області.....	14
1.3. Сучасні технології та тенденції розробки вебзастосунків електронної комерції.....	18
1.4. Критичний аналіз ризиків і обмежень розроблюваної системи.....	23
1.5. Висновки по розділу.....	24
РОЗДІЛ 2. ФОРМУВАННЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ AUTO PARTS.....	26
2.1. Визначення користувачів та сценаріїв роботи системи.....	26
2.2. Функціональні та нефункціональні вимоги до вебзастосунк.....	28
2.3. Постановка задачі розроблення системи.....	31
2.4. Висновки по розділу.....	33
РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛІ ДАНИХ ВЕБЗАСТОСУНКУ AUTO PARTS.....	35
3.1. Обґрунтування архітектури системи.....	35

					БР.ІП – 05.00.00.000 ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дат				
Розроб.		Левицький Б.Л.			Проектування та розробка веб-застосунку для автоматизації продажу автомобільних запчастин Пояснювальна записка	Літ.	Арк.	Аркушів
Перевір.		Дмитрик Т.Б.					6	80
Реценз.		Царева О.С..				ІФНТУНГ, ІП-22-4		
Н. Контр.		Піх М. М.						
Затверд.		Бандура В. В.						

3.2. Проєктування програмних модулів і API-взаємодії.....	37
3.3. Проєктування структури бази даних.....	39
3.4. Проєктування сценарію оформлення замовлення та інтеграцій.....	42
3.5. Висновки по розділу.....	44
РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ AUTO PARTS.....	45
4.1. Структура проєкту та підготовка середовища.....	45
4.2. Реалізація клієнтської частини.....	47
4.3. Реалізація серверної частини та авторизації.....	49
4.4. Реалізація каталогу, гаража та адміністративної панелі.....	51
4.5. Висновки по розділу.....	63
РОЗДІЛ 5. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ AUTO PARTS.....	64
5.1. Стратегія та середовище тестування.....	64
5.2. Функціональне тестування користувацьких сценаріїв.....	66
5.3. Інтеграційне тестування та перевірка обробки помилок.....	70
5.4. Перевірка якості, безпеки та production-збірки.....	71
5.5. Впровадження системи в експлуатацію.....	74
5.6. Висновки по розділу.....	75
ВИСНОВКИ.....	77
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	80
БІБЛІОГРАФІЧНА ДОВІДКА	

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

API - інтерфейс прикладного програмування

REST - архітектурний стиль взаємодії клієнта і сервера

JSON - формат обміну даними

HTTP - протокол передавання даних у вебзастосунках

SQL - мова структурованих запитів до бази даних

БД - база даних

СУБД - система управління базами даних

UML - уніфікована мова моделювання

UI - користувацький інтерфейс

a - бібліотека для створення користувацького інтерфейсу

Node.js - платформа для виконання JavaScript на сервері

PostgreSQL - реляційна система управління базами даних

ID - унікальний ідентифікатор об'єкта

OEM - оригінальний номер деталі виробника

checkout - процес оформлення замовлення в інтернет-магазині

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дат		

ВСТУП

Сучасний ринок автозапчастин активно переходить у цифрове середовище, оскільки користувачі все частіше очікують швидкого пошуку потрібної деталі, можливості порівняти товари, перевірити сумісність із власним автомобілем та оформити замовлення без необхідності особистого звернення до магазину. Особливо актуальною є проблема правильного підбору запчастин, адже одна й та сама категорія товарів може мати різні параметри залежно від марки, моделі, покоління та року випуску автомобіля.

У зв'язку з цим виникає потреба у вебзастосунках, які поєднують електронний каталог автозапчастин, зручний пошук, фільтрацію товарів, механізм підбору деталей під автомобіль користувача та повний цикл оформлення замовлення. Така система дозволяє скоротити час пошуку потрібної позиції, зменшити ймовірність помилки при виборі товару та підвищити зручність взаємодії клієнта з онлайн-магазином.

Вебсайт онлайн-магазину автозапчастин Auto Parts, який забезпечує перегляд каталогу товарів, пошук за назвою, брендом, артикулом і моделлю автомобіля, додавання товарів до кошика та обраного, оформлення замовлень, роботу з особистим кабінетом, гаражем користувача та адміністративною панеллю. Серверна частина системи реалізує обробку даних користувачів, товарів, автомобілів, замовлень, відгуків і рейтингів. Об'єктом бакалаврської роботи є процес розроблення веборієнтованої інформаційної системи для продажу автозапчастин.

Предметом бакалаврської роботи є методи, засоби та технології реалізації онлайн-магазину автозапчастин із функціями підбору товарів за автомобілем, керування каталогом, оформлення замовлень і адміністрування даних.

Мета бакалаврської роботи - розробити вебзастосунок для онлайн-продажу автозапчастин, який забезпечує зручний пошук товарів, підбір деталей

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дат		

відповідно до автомобіля користувача, оформлення замовлень та керування даними через адміністративну панель.

Для досягнення поставленої мети необхідно виконати такі завдання:

- проаналізувати предметну область онлайн-продажу автозапчастин та визначити основні вимоги до вебзастосунку;
- сформулювати функціональні та нефункціональні вимоги до системи;
- спроектувати структуру вебзастосунку, базу даних і взаємодію між клієнтською та серверною частинами;
- реалізувати серверну частину на основі Node.js, Express.js і PostgreSQL;
- розробити модулі каталогу, кошика, обраних товарів, гаража користувача, оформлення замовлень, авторизації та адміністративної панелі;
- реалізувати збереження і обробку даних про користувачів, товари, автомобілі, замовлення, відгуки та рейтинги;
- передбачити можливість онлайн-оплати через зовнішній платіжний сервіс;
- провести перевірку працездатності основних функцій вебзастосунку.

Методи, що застосовувалися під час виконання роботи, включають аналіз предметної області, порівняння існуючих рішень, моделювання структури даних, проєктування REST API, компонентний підхід до розробки інтерфейсу користувача, роботу з реляційною базою даних та функціональне тестування реалізованих можливостей.

Практичне значення отриманих результатів полягає в тому, що розроблений вебзастосунок може бути використаний як основа для створення повноцінного онлайн-магазину автозапчастин. Система дає змогу користувачам швидко знаходити потрібні товари, зберігати автомобілі у власному гаражі, формувати кошик, оформлювати замовлення та відстежувати історію покупок, а адміністратору - керувати товарами, замовленнями, користувачами й автомобільними відповідностями.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дат		

РОЗДІЛ 1. ОГЛЯД ІСНУЮЧИХ РІШЕНЬ ТА ТЕОРЕТИЧНИХ ОСНОВ РОЗРОБКИ ОНЛАЙН-МАГАЗИНУ АВТОЗАПЧАСТИН

1.1. Аналіз сучасного стану проблематики

Розвиток електронної комерції суттєво змінив спосіб взаємодії між продавцем і покупцем. Якщо раніше покупець здебільшого отримував інформацію про товар безпосередньо в магазині або через консультацію менеджера, то сьогодні значна частина цього процесу перенесена у вебсередовище. Онлайн-магазин повинен не лише показувати асортимент, а й забезпечувати пошук, фільтрацію, перегляд характеристик, формування кошика, оформлення замовлення, оплату, вибір доставки та збереження історії взаємодії з користувачем.

Сфера продажу автозапчастин має специфіку, яка робить її складнішою за більшість напрямів класичної електронної комерції. Автозапчастина не є універсальним товаром: одна й та сама деталь може мати декілька модифікацій, відрізнятися артикулом, OEM-номером, виробником, геометричними параметрами, матеріалами, гарантійними умовами та сумісністю з конкретними моделями автомобілів. Для користувача це означає, що звичайного пошуку за назвою товару часто недостатньо. Покупцеві потрібно переконатися, що обрана позиція підходить до марки, моделі, покоління, року випуску, типу кузова та двигуна його автомобіля.

Помилка при виборі автозапчастини може призвести до повернення товару, повторного замовлення, додаткових витрат часу та потреби в консультації фахівця. Для бізнесу це означає збільшення навантаження на менеджерів, ризик негативного користувацького досвіду та додаткові витрати на обробку повернень. Саме тому сучасний онлайн-магазин автозапчастин повинен поєднувати класичні функції інтернет-магазину з механізмами технічного підбору товарів.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		11

На сучасному ринку можна виділити кілька типів систем, які використовуються для пошуку та купівлі автозапчастин. Перший тип - спеціалізовані інтернет-магазини, орієнтовані саме на автотовари. Такі сервіси зазвичай мають великий каталог, поділ за категоріями, пошук за артикулами та можливість консультації з менеджером. Другий тип - маркетплейси та дошки оголошень, де товари розміщують різні продавці. Вони забезпечують великий вибір, але не завжди мають єдину структуру опису товарів і перевірку сумісності.

Третій тип - внутрішні системи обліку магазинів або постачальників, які зручні для працівників, але часто не мають повноцінного клієнтського інтерфейсу.

До поширених онлайн-платформ, які використовуються для пошуку та купівлі автозапчастин, можна віднести Exist.ua, Emex, Autodoc та OLX Автозапчастини. Ці сервіси мають різне призначення та різний підхід до організації даних.

Спеціалізовані онлайн-магазини більше орієнтуються на структурований каталог і технічні характеристики, тоді як дошки оголошень роблять акцент на кількості пропозицій від різних продавців. Водночас для користувача важливо, щоб система не вимагала надмірних технічних знань і давала можливість швидко перейти від пошуку товару до оформлення замовлення.

У таблиці 1.1 наведено порівняльний аналіз поширених онлайн-платформ, що надають можливість пошуку або придбання автозапчастин. Для аналізу обрано сервіси Exist.ua, Emex, Autodoc та OLX Автозапчастини, оскільки вони представляють різні підходи до організації продажу: спеціалізований магазин, каталог постачальника, міжнародний інтернет-магазин та дошку оголошень.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		12

Порівняльний аналіз поширених онлайн-платформ

Платформа	Характеристика	Переваги	Обмеження
Exist.ua	Спеціалізований онлайн-магазин автозапчастин	Великий каталог, орієнтація на український ринок, можливість консультації	Для точного підбору часто потрібні технічні дані автомобіля або допомога менеджера
Emex	Каталог і сервіс продажу автозапчастин	Значна кількість позицій, пошук за каталогом, робота з артикулами	Інтерфейс і логіка підбору можуть бути складними для користувача без досвіду
Autodoc	Інтернет-магазин автозапчастин та аксесуарів	Широкий асортимент, категоризація товарів, підбір деталей	Великий обсяг каталогу потребує якісної фільтрації та уважного вибору
OLX Автозапчастини	Дошка оголошень для нових і вживаних деталей	Багато пропозицій від різних продавців, можливість пошуку вживаних деталей	Немає єдиної структури каталогу, складніше перевірити сумісність і надійність продавця

Порівняльний аналіз показує, що наявні рішення мають розвинений функціонал, але кожне з них має певні обмеження. Спеціалізовані магазини забезпечують широкий каталог і технічну орієнтацію, але можуть залишатися складними для користувача, який не знає точного артикула або OEM-номера. Маркетплейси та дошки оголошень дають багато пропозицій, проте не гарантують однакової якості опису товарів, а також не завжди мають зручний підбір під автомобіль. Отже, актуальним залишається завдання створення вебзастосунку, який поєднає структурований каталог, простий користувацький інтерфейс, підбір за автомобілем, оформлення замовлення, онлайн-оплату та доставку.

У межах бакалаврської роботи розроблено вебзастосунок Auto Parts, який спрямований на вирішення цих проблем. Система містить каталог автозапчастин, сторінки товарів, пошук, фільтрацію, кошик, список обраних товарів, модуль гаража користувача, оформлення замовлень, рейтинг, відгуки та адміністративну панель. Таке поєднання функцій дозволяє розглядати

систему не лише як каталог товарів, а як повноцінний інструмент автоматизації продажу автозапчастин.

Окремою перевагою розроблюваної системи є модуль гаража користувача. Користувач може додати до гаража свій автомобіль, вказавши марку, модель, покоління, роки випуску, тип кузова та додаткову інформацію про двигун. Надалі ці дані можуть використовуватися для пошуку й фільтрації сумісних товарів. Такий підхід підвищує зручність роботи з каталогом і зменшує потребу в ручному введенні однакової інформації під час кожного пошуку.

Крім того, сучасний онлайн-магазин повинен забезпечувати не лише пошук товарів, а й повний сценарій покупки. У проєкті Auto Parts реалізовано оформлення замовлення з вибором доставки, збереженням контактної інформації, обліком позицій замовлення та зміною залишків товарів. Для доставки використовується інтеграція з API Нової пошти, що дає змогу отримувати список міст, відділень, поштоматів і підтримувати кур'єрський сценарій [9]. Для оплати передбачено збереження способу та статусу оплати в межах замовлення.

Таким чином, проблема, яка розглядається в бакалаврській роботі, полягає у необхідності розроблення веборієнтованої інформаційної системи для онлайн-продажу автозапчастин, яка поєднує каталог, підбір за автомобілем, кошик, обрані товари, оформлення замовлення, онлайн-оплату, доставку, історію покупок і адміністративне керування даними.

1.2. Основні поняття та нормативно-технічні аспекти предметної області

Онлайн-магазин автозапчастин можна розглядати як веборієнтовану інформаційну систему, яка забезпечує зберігання, обробку та представлення даних про товари, користувачів, автомобілі й замовлення. На відміну від простого інформаційного сайту, така система має виконувати транзакційні

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		14

операції: створювати замовлення, змінювати залишки товарів, зберігати історію покупок, обробляти оплату та передавати дані для доставки.

Основними учасниками системи є покупець, адміністратор і програмна система. Покупець переглядає каталог, шукає товари, додає автомобілі в гараж, формує кошик, оформлює замовлення та залишає відгуки. Адміністратор керує товарами, користувачами, замовленнями й автомобільними відповідностями. Програмна система забезпечує взаємодію між клієнтським інтерфейсом, серверною логікою, базою даних і зовнішніми API.

Каталог товарів є центральною частиною онлайн-магазину. У випадку автозапчастин каталог повинен містити не лише назву та ціну, а й технічні характеристики, бренд, артикул, OEM-номер, категорію, підкатегорію, залишок на складі, зображення, опис, гарантійні умови, країну походження та інформацію про сумісність із транспортними засобами. Чим повнішим є опис товару, тим легше користувачеві прийняти рішення без додаткової консультації.

Артикул - це ідентифікатор товару в межах каталогу або системи постачальника. OEM-номер - це номер оригінальної деталі виробника автомобіля. У сфері автозапчастин ці ідентифікатори мають особливе значення, оскільки одна й та сама назва може відповідати різним деталям. Наприклад, фільтр, гальмівні колодки або ремінь можуть відрізнятися залежно від двигуна, комплектації чи року випуску автомобіля. Тому пошук за артикулом та OEM-номером є важливою функцією для такого вебзастосунку.

Гараж користувача є модулем, який дозволяє зберігати інформацію про автомобілі покупця. У системі Auto Parts для автомобіля передбачені такі характеристики: марка, модель, покоління, роки випуску, тип кузова та примітка щодо двигуна. Збереження цих даних дає змогу формувати персоналізований сценарій пошуку, у якому користувач працює не лише з загальним каталогом, а й з товарами, що можуть відповідати його автомобілю.

Кошик виконує роль проміжного сховища товарів перед оформленням покупки. Він містить вибрані позиції, кількість, ціну, загальну суму та

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						15
Змн.	Арк.	№ докум.	Підпис	Дат		

додаткову інформацію про варіант товару. У системі Auto Parts кошик зберігається на клієнтській стороні, що дозволяє користувачеві продовжити вибір товарів без обов'язкової авторизації. Після переходу до оформлення замовлення дані кошика передаються на сервер, де перевіряється наявність товарів і формується замовлення.

Модуль обраних товарів дозволяє користувачу зберігати позиції, які його зацікавили, але ще не додані до кошика. Для онлайн-магазину автозапчастин це зручно, оскільки покупець часто порівнює декілька аналогів різних брендів, різні ціни або варіанти за гарантією. Обрані товари підвищують зручність роботи з каталогом і зменшують імовірність втрати знайденої позиції.

Оформлення замовлення є одним із найважливіших бізнес-процесів системи. Воно включає перевірку кошика, введення контактних даних, вибір доставки, вибір способу оплати, створення замовлення, запис позицій замовлення та оновлення залишків товарів. У системі Auto Parts також передбачено різні способи доставки: відділення Нової пошти, поштомат, кур'єрська доставка та самовивіз.

Під час оформлення замовлення система обробляє персональні дані користувача: ім'я, прізвище, телефон, електронну пошту, місто та адресу доставки. Тому під час розробки потрібно враховувати принципи конфіденційності, мінімізації зберезуваних даних і використання інформації лише для функціональних потреб системи: створення облікового запису, оформлення доставки, збереження історії замовлень і зв'язку з покупцем.

Важливим є також питання електронної оплати. У розроблюваній системі передбачено можливість подальшого підключення зовнішнього платіжного сервісу для онлайн-оплати. Такий підхід є безпечнішим, оскільки магазин не повинен самотійно зберігати чутливі платіжні реквізити користувачів, а може делегувати платіжний сценарій спеціалізованому провайдеру.

Окрему увагу слід приділити безпечній автентифікації користувачів. У системі Auto Parts реалізовано реєстрацію, вхід за email і паролем,

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						16
Змн.	Арк.	№ докум.	Підпис	Дат		

підтвердження електронної пошти, відновлення пароля та вхід через Google. Паролі не зберігаються у відкритому вигляді. Для їх захисту використовується bcrypt: під час реєстрації пароль перетворюється на хеш, а під час входу введене значення порівнюється з хешем, збереженим у базі даних. OWASP рекомендує використовувати спеціальні алгоритми хешування паролів і не зберігати паролі у відкритому вигляді [5].

Адміністративна панель є необхідним елементом системи, оскільки каталог автозапчастин потребує постійного оновлення. Адміністратор повинен мати можливість додавати нові товари, редагувати характеристики, змінювати ціну й залишок, додавати зображення, прив'язувати товар до автомобілів, переглядати замовлення та змінювати їх статуси. Без такого модуля система могла б працювати лише як статичний каталог, але не як повноцінний онлайн-магазин. На рисунку 1.1 наведено узагальнений процес взаємодії користувача з онлайн-магазином автозапчастин.



Рисунок 1.1 – Узагальнений процес взаємодії користувача з онлайн-магазином автозапчастин.

У таблиці 1.2 наведено основні сутності предметної області, які використовуються в системі Auto Parts.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		17

Основні сутності предметної області

Сутність	Призначення	Приклади даних
Користувач	Зберігання даних облікового запису та контактної інформації	ім'я, email, телефон, роль, адреса доставки
Товар	Представлення автозапчастини в каталозі	назва, бренд, артикул, ціна, залишок, опис
Автомобіль	Опис транспортного засобу для підбору запчастин	марка, модель, покоління, роки випуску, кузов
Гараж	Зв'язок користувача з його автомобілями	користувач, вибране авто
Замовлення	Фіксація покупки користувача	покупець, товари, доставка, оплата, статус
Відгук	Збереження думки користувача про товар	ім'я, текст, оцінка, дата
Рейтинг	Числова оцінка товару	товар, оцінка, токен клієнта

Таким чином, предметна область онлайн-магазину автозапчастин охоплює не лише представлення товарів, а й складну взаємодію між користувачами, автомобілями, товарами, замовленнями, оплатою та доставкою. Саме ця комплексність визначає вимоги до архітектури і технологій розроблюваної системи.

1.3. Сучасні технології та тенденції розробки вебзастосунків електронної комерції

Сучасні вебзастосунки електронної комерції зазвичай будуються за клієнт-серверною архітектурою. Такий підхід дозволяє розділити відповідальність між частинами системи. Клієнтська частина відповідає за відображення інтерфейсу, взаємодію з користувачем, роботу з формами, локальним станом, кошиком і навігацією. Серверна частина реалізує бізнес-

логіку, перевіряє дані, працює з базою даних, обробляє замовлення, автентифікує користувачів і взаємодіє із зовнішніми API.

У системі Auto Parts клієнтська частина реалізована з використанням React. React підтримує компонентний підхід, за якого інтерфейс поділяється на незалежні частини. Це важливо для онлайн-магазину, оскільки інтерфейс складається з багатьох повторюваних елементів: карток товарів, кнопок додавання до кошика, блоків рейтингу, форм, списків замовлень, панелей фільтрації та елементів навігації [2]. Компонентна структура спрощує підтримку проєкту, оскільки окремі частини інтерфейсу можна змінювати без повного переписування сторінки.

Для збирання й запуску клієнтської частини використовується Vite. Цей інструмент орієнтований на швидкий запуск dev-сервера та ефективну розробку сучасних вебзастосунків. Підтримка Hot Module Replacement дозволяє бачити зміни в інтерфейсі без повного перезавантаження сторінки, що пришвидшує розробку і тестування UI-компонентів [3]. Для дипломного проєкту це має практичне значення, оскільки інтерфейс магазину містить багато сторінок і станів: каталог, товар, кошик, checkout, кабінет, гараж, обране, адмін-панель.

Серверна частина системи реалізована на Node.js. Node.js дає змогу використовувати JavaScript не лише в браузері, а й на сервері, що спрощує загальну технологічну екосистему проєкту [4]. У поєднанні з Express.js він забезпечує створення API-маршрутів, обробку HTTP-запитів і повернення відповідей у форматі JSON. Express.js описує маршрути як спосіб визначити, як сервер реагує на запити до певних URI з використанням конкретних HTTP-методів [6].

Взаємодія між клієнтом і сервером організована через REST API. HTTP-методи GET, POST, PUT, PATCH і DELETE використовуються для роботи з ресурсами системи. Наприклад, GET застосовується для отримання списку товарів або замовлень, POST - для створення користувача, замовлення чи відгуку, PUT і PATCH - для оновлення профілю або статусу замовлення,

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дат		

DELETE - для видалення товару або автомобіля з гаража. Основні принципи семантики HTTP описані в RFC 9110 [1].

У проєкті Auto Parts реалізовано такі основні групи API-маршрутів: авторизація та реєстрація, Google-вхід, робота з користувачами, профілем, гаражем, товарами, популярними товарами, пошуком, відгуками, рейтингами, замовленнями, доставкою та checkout. Така структура відповідає функціональним модулям системи і дозволяє розділити бізнес-логіку на зрозумілі частини.

Для збереження даних використано PostgreSQL. Реляційна модель добре підходить для онлайн-магазину, оскільки дані мають чіткі зв'язки. Користувач може мати багато замовлень, замовлення містить багато позицій, товар може бути пов'язаний з кількома автомобільними відповідностями, а автомобіль може бути збережений у гаражі декількох користувачів. PostgreSQL надає засоби для роботи з таблицями, зв'язками, SQL-запитами та транзакційними операціями [7].

Під час оформлення замовлення важливо забезпечити цілісність даних. Якщо користувач купує товар, система повинна перевірити його наявність, зменшити залишок на складі, створити замовлення та записати позиції замовлення. Якщо одна з цих операцій не виконається, дані можуть стати некоректними. У серверній частині Auto Parts для таких сценаріїв застосовується транзакційна логіка: операції оформлення замовлення виконуються як єдиний процес, що дозволяє уникнути частково створених замовлень.

Однією з тенденцій сучасних вебзастосунків є інтеграція із зовнішніми сервісами. Для онлайн-магазину особливо важливими є сервіси доставки, авторизація через сторонні облікові записи, можливість підключення платіжного сервісу та допоміжні API. У системі Auto Parts реалізовано Google-авторизацію, інтеграцію з Новою поштою та допоміжне використання SerpApi для підготовки зображень товарів.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дат		

Google-авторизація спрощує вхід користувача в систему. Користувач може увійти через Google-акаунт, після чого сервер перевіряє ID-токен. Офіційна документація Google наголошує, що перевірку ID-токена потрібно виконувати на серверній стороні, зокрема перевіряти підпис, аудиторію, емітента та термін дії токена [8]. У проєкті для цього використовується бібліотека Google Auth Library.

Інтеграція з API Нової пошти потрібна для зручного оформлення доставки. Замість того щоб користувач вручну вводив місто та адресу відділення, система отримує актуальні дані з API. Це зменшує кількість помилок, пришвидшує оформлення замовлення і робить checkout ближчим до реального сценарію українських онлайн-магазинів [9].

Можливість онлайн-оплати розглядається як напрям подальшого розширення checkout. У такому сценарії магазин повинен зберігати лише спосіб і статус оплати, суму замовлення та службову інформацію, необхідну для обробки покупки. Повні платіжні реквізити користувача не повинні зберігатися в базі даних магазину, оскільки для цього використовуються спеціалізовані платіжні сервіси.

Допоміжне використання SerpApi пов'язане з наповненням каталогу зображеннями товарів. Для інтернет-магазину візуальна частина має велике значення, оскільки користувач очікує бачити товар, його упаковку або приклад зовнішнього вигляду.

У проєкті скрипти з SerpApi використовуються не як основна функція для кінцевого користувача, а як технічний інструмент підготовки каталогу [11]. На рисунку 1.2 наведено концептуальну схему взаємодії компонентів системи Auto Parts.

У таблиці 1.3 наведено узагальнення технологій та зовнішніх API, використаних у проєкті Auto Parts.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		21

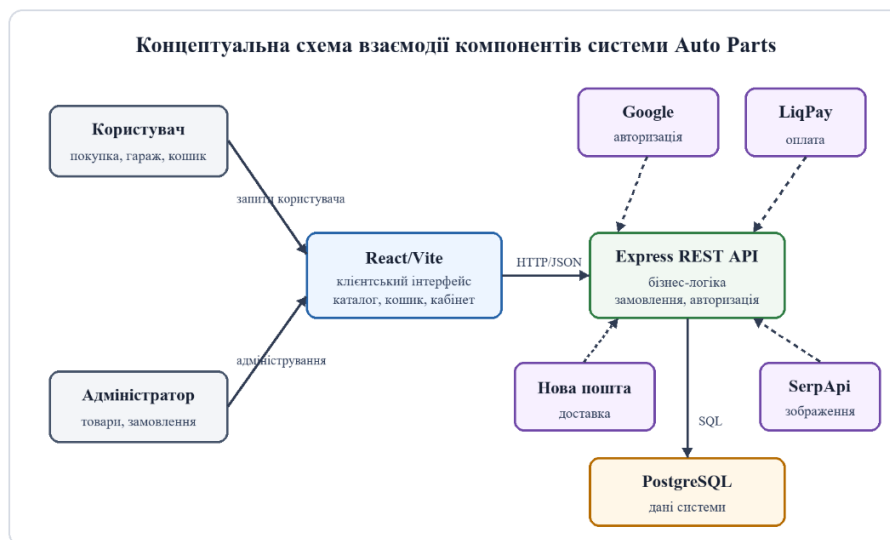


Рисунок 1.2 – Концептуальна схема взаємодії компонентів системи Auto Parts.

Таблиця 1.3

Узагальнення технологій та зовнішніх API

Компонент	Використаний засіб	Призначення в системі
Клієнтська частина	React	Побудова інтерфейсу користувача на основі компонентів
Збирання фронтенду	Vite	Запуск dev-сервера, збирання клієнтської частини
Серверна частина	Node.js	Виконання JavaScript на сервері
API-сервер	Express.js	Створення REST API та обробка HTTP-запитів
База даних	PostgreSQL	Зберігання користувачів, товарів, замовлень, відгуків
Захист паролів	bcrypt	Хешування паролів користувачів
Авторизація	Google Identity	Вхід через Google-акаунт
Доставка	API Нової пошти	Отримання міст, відділень, поштоматів
Оплата	Платіжний сервіс	Можливість подальшого підключення онлайн-оплати
Зображення товарів	SerpApi	Допоміжне наповнення каталогу зображеннями

Вибір таких технологій обґрунтовується їх поширеністю, достатньою функціональністю та відповідністю задачам системи. React і Vite забезпечують

зручну розробку інтерфейсу, Node.js та Express.js дозволяють реалізувати REST API, PostgreSQL підходить для структурованих даних, а зовнішні API дають можливість реалізувати практичні сценарії доставки, оплати й авторизації.

1.4. Критичний аналіз ризиків і обмежень розроблюваної системи

Під час розробки онлайн-магазину автозапчастин важливо враховувати не лише функціональні можливості, а й потенційні ризики. Перший ризик пов'язаний із якістю даних у каталозі. Якщо товар має неправильний артикул, неточний опис або помилкову прив'язку до автомобіля, користувач може замовити несумісну деталь. Тому система повинна забезпечувати зручне адміністрування товарів і автомобільних відповідностей.

Другий ризик стосується залишків товарів. В онлайн-магазині користувач очікує, що інформація про наявність є актуальною. Якщо товар відображається як доступний, але фактично відсутній на складі, це погіршує довіру до сервісу. У проєкті Auto Parts під час оформлення замовлення сервер перевіряє залишок товару і зменшує його після успішного створення замовлення.

Третій ризик пов'язаний із безпекою облікових записів. Оскільки система має реєстрацію, вхід, профіль користувача та історію замовлень, важливо захищати паролі й не зберігати їх у відкритому вигляді. Використання bcrypt зменшує ризик компрометації паролів у разі несанкціонованого доступу до бази даних [5].

Четвертий ризик стосується обробки персональних даних. Під час оформлення замовлення користувач вводить контактну інформацію та дані доставки. Тому система повинна використовувати ці дані тільки для цілей, пов'язаних із роботою магазину, та не збирати зайву інформацію.

П'ятий ризик пов'язаний із зовнішніми API. Інтеграції з Google, Новою поштою та SerpApi залежать від доступності сторонніх сервісів, правильності API-ключів і стабільності мережевого з'єднання. Якщо зовнішній сервіс

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		23

тимчасово недоступний, система повинна коректно обробляти помилку і не втрачати дані користувача. Наприклад, якщо не вдається отримати список відділень Нової пошти, користувач повинен побачити зрозуміле повідомлення, а не технічну помилку.

Шостий ризик стосується платіжних сценаріїв. Онлайн-оплата повинна бути реалізована так, щоб магазин не зберігав чутливі платіжні реквізити користувача. Навіть у разі подальшого підключення платіжного сервісу система повинна правильно зберігати статус оплати, суму замовлення та інформацію про метод оплати.

Окремим обмеженням дипломного проєкту є те, що система реалізована як прототип, а не як промислова платформа з повноцінною масштабованою інфраструктурою. У межах роботи не розглядаються складні механізми складського обліку, інтеграція з бухгалтерськими системами, автоматичне оновлення цін від постачальників або багаторівнева система ролей. Проте архітектура системи залишає можливість для подальшого розширення.

У майбутньому систему можна розвивати в кількох напрямках. По-перше, можна вдосконалити механізм підбору запчастин, додавши складніші правила сумісності. По-друге, можна реалізувати імпорт товарів від постачальників. По-третє, можна додати аналітику продажів, звіти для адміністратора та автоматичні рекомендації товарів. По-четверте, можна посилити безпеку через повноцінну серверну авторизацію на основі токенів або сесій. Таким чином, розроблювана система має практичну основу для онлайн-продажу автозапчастин, але також містить напрями для подальшого розвитку. Це відповідає характеру бакалаврської роботи, де основним результатом є робочий прототип із ключовими функціями предметної області.

1.5. Висновки по розділу

У першому розділі було розглянуто предметну область онлайн-продажу автозапчастин і визначено її ключові особливості. Встановлено, що основна

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		24

складність полягає не лише у продажу товарів через інтернет, а й у правильному підборі деталей відповідно до конкретного автомобіля користувача. Саме ця особливість відрізняє онлайн-магазин автозапчастин від багатьох інших систем електронної комерції. Проаналізовано існуючі онлайн-платформи для пошуку й придбання автозапчастин. Виявлено, що вони мають широкий функціонал, але не завжди забезпечують просте поєднання каталогу, підбору за автомобілем, кошика, обраних товарів, онлайн-оплати, доставки та адміністративного керування. Це обґрунтовує доцільність розробки вебзастосунку Auto Parts. У розділі також розглянуто основні поняття предметної області: каталог товарів, артикул, OEM-номер, гараж користувача, кошик, обрані товари, оформлення замовлення, адміністративна панель, рейтинг і відгуки. Окрему увагу приділено нормативним аспектам електронної комерції та обробки персональних даних.

Описано технології та зовнішні API, використані у проєкті. Для клієнтської частини обрано React і Vite, для серверної частини - Node.js та Express.js, для збереження даних - PostgreSQL, для захисту паролів - bcrypt. Також у системі реалізовано Google-авторизацію, інтеграцію з API Нової пошти та допоміжне використання SerpApi для підготовки зображень товарів.

Проведений аналіз показав, що обрані технології та архітектурні підходи відповідають задачам дипломного проєкту і дозволяють реалізувати функціональний вебзастосунок, наближений до реальних умов роботи онлайн-магазину автозапчастин.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		25

РОЗДІЛ 2. ФОРМУВАННЯ ВИМОГ ДО ВЕБЗАСТОСУНКУ AUTO PARTS

2.1. Визначення користувачів та сценаріїв роботи системи

Аналіз вимог є початковим етапом проєктування програмної системи, оскільки саме на цьому етапі визначається, які задачі має розв'язувати майбутній вебзастосунок і які обмеження потрібно врахувати під час його реалізації. Для проєкту Auto Parts основною метою є створення онлайн-магазину автозапчастин, який дозволяє користувачеві знайти потрібний товар, перевірити його характеристики, додати позицію до кошика, обрати спосіб доставки та оформити замовлення. Водночас система повинна мати адміністративну частину для підтримки каталогу товарів, контролю замовлень і керування автомобільними відповідностями.

Предметна область продажу автозапчастин має особливість, яка відрізняє її від більшості звичайних інтернет-магазинів. Користувач купує не просто товар із певною назвою та ціною, а деталь, яка повинна відповідати конкретному автомобілю. Для покупця важливими є артикул, бренд, OEM-номер, категорія, сумісність із маркою та моделлю автомобіля, наявність на складі, умови доставки й можливість швидкого оформлення замовлення. Тому вимоги до системи Auto Parts охоплюють не лише стандартний сценарій онлайн-покупки, а й механізми підбору товарів під автомобіль.

Під час аналізу вимог було визначено дві основні групи користувачів системи: покупець і адміністратор. Покупець використовує сайт для перегляду каталогу, пошуку товарів, додавання товарів до кошика або обраного, збереження автомобілів у гаражі, оформлення замовлень і перегляду історії покупок. Адміністратор відповідає за актуальність даних у системі: додає та редагує товари, вказує характеристики, змінює залишки, керує сумісністю деталей з автомобілями, переглядає користувачів і змінює статуси замовлень.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		26

На рисунку 2.1 наведено діаграму варіантів використання системи Auto Parts.

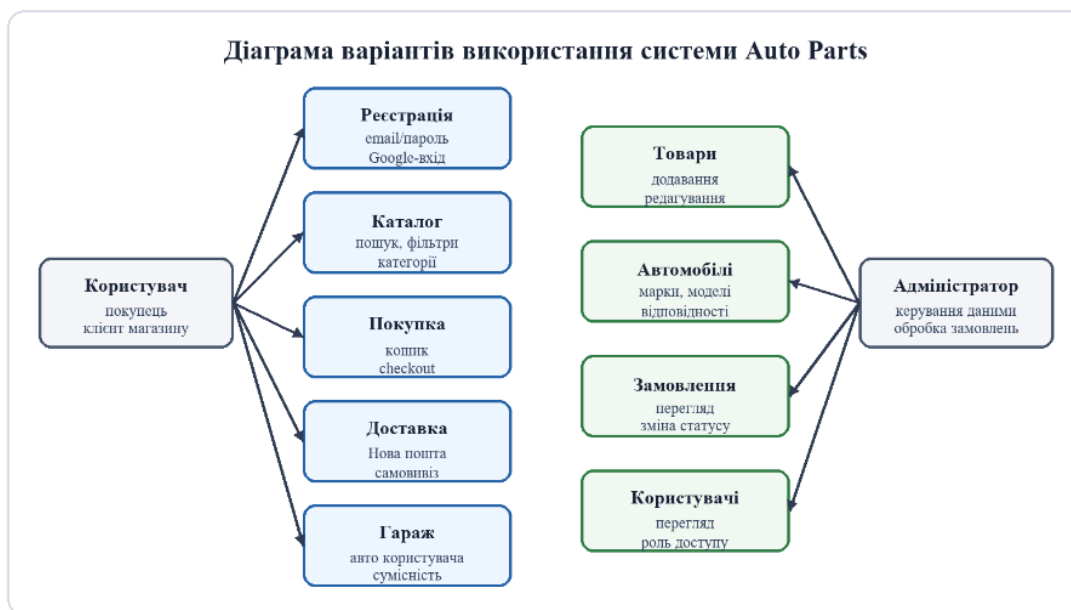


Рисунок 2.1 – Діаграма варіантів використання системи Auto Parts.

З рисунка 2.1 видно, що основна частина сценаріїв пов'язана з покупкою товару. Користувач повинен мати можливість перейти від пошуку товару до створення замовлення без зайвих дій. Для цього в системі реалізовано каталог, картку товару, кошик, checkout, вибір доставки й оплати. Додатковими, але важливими сценаріями є гараж користувача та список обраного, оскільки вони підвищують зручність повторного використання сайту.

Окремою вимогою є авторизація користувачів. У системі передбачено реєстрацію, вхід за email і паролем, підтвердження електронної пошти, відновлення пароля та вхід через Google. Паролі не зберігаються у відкритому вигляді: під час реєстрації вони перетворюються на хеш за допомогою bcrypt, а під час входу введене значення порівнюється із хешем у базі даних. Такий підхід відповідає рекомендаціям щодо безпечного зберігання паролів [18].

Вхід через Google реалізовано з перевіркою ID-токена на серверній стороні, що узгоджується з рекомендаціями Google Identity Services [21].

Вимоги до каталогу пов'язані з необхідністю швидкого пошуку товарів. Користувач може не знати точної назви деталі, але може знати бренд, артикул, OEM-номер, категорію або модель автомобіля. Тому каталог повинен підтримувати пошук за кількома полями, фільтрацію за категоріями, брендом, ціною, наявністю та сумісністю з автомобілем із гаража. Для інтернет-магазину автозапчастин це має практичне значення, оскільки помилка підбору товару може призвести до повернення або повторного замовлення.

Вимоги до оформлення замовлення включають перевірку кошика, контактних даних, способу доставки, способу оплати та залишків товару на складі. У проєкті реалізовано інтеграцію з API Нової пошти для отримання міст, відділень, поштоматів і підтримки кур'єрського сценарію [23]. Для оплати система повинна зберігати тільки дані, необхідні для обробки замовлення: спосіб оплати, статус оплати, суму, контактну інформацію та дані доставки.

Вимоги до адміністративної частини сформовано з урахуванням того, що каталог автозапчастин потребує постійного оновлення. Адміністратор повинен мати можливість створювати нові товари, змінювати ціну, залишок, опис, зображення, категорію, підкатегорію, OEM-номер і сумісність із автомобілями. Також адміністратор повинен переглядати замовлення і змінювати їх статуси. Без такого функціоналу система швидко втратить актуальність, оскільки дані про товари та складські залишки постійно змінюються.

2.2 Функціональні та нефункціональні вимоги до вебзастосунку

Функціональні вимоги описують конкретні дії, які система повинна виконувати. Для Auto Parts базовими є вимоги, пов'язані з користувацьким сценарієм покупки: реєстрація, авторизація, перегляд каталогу, пошук товару, перегляд картки товару, додавання до кошика, додавання до обраного, вибір автомобіля, оформлення доставки, оплата та створення замовлення. Ці вимоги

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		28

визначають мінімальний набір можливостей, без яких система не може виконувати роль онлайн-магазину.

Першою функціональною вимогою є підтримка облікових записів користувачів. Користувач повинен мати можливість зареєструватися, увійти в систему, підтвердити електронну пошту, відновити пароль і вийти з акаунта. Після входу система повинна зберігати дані поточного користувача, щоб надалі використовувати їх у профілі, гаражі, історії замовлень і checkout. Для адміністратора роль користувача має визначати доступ до службових функцій.

Другою вимогою є робота з каталогом товарів. Каталог повинен відображати список автозапчастин із назвою, брендом, ціною, зображенням, категорією, рейтингом і наявністю. Користувач повинен мати змогу шукати товар за текстовим запитом, фільтрувати товари за категорією та брендом, обмежувати ціну, переглядати лише доступні позиції й сортувати результати. Оскільки вебзастосунок працює через HTTP-запити, взаємодія клієнта із сервером має бути організована через зрозумілі API-маршрути, що відповідає загальній семантиці HTTP [1].

На клієнтській стороні такі запити доцільно виконувати через стандартний механізм Fetch API, який підтримується сучасними браузерами і використовується для обміну даними між інтерфейсом та сервером [26].

Третьою вимогою є наявність детальної сторінки товару. На цій сторінці користувач повинен бачити не лише назву й ціну, а й повний опис, характеристики, артикул, OEM-номер, зображення, гарантію, країну походження, стан товару, рейтинг, відгуки та інформацію про сумісність. Якщо користувач має збережений автомобіль у гаражі, система повинна показувати, чи може обрана деталь відповідати цьому автомобілю.

Четвертою вимогою є реалізація кошика. Користувач повинен мати змогу додати товар до кошика, змінити кількість, видалити позицію та перейти до оформлення замовлення. Кошик повинен зберігати товари між переходами сторінками, а кількість товарів має відображатися в шапці сайту. Для зручності користувача система також повинна підтримувати список обраних товарів. Для

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дат		

збереження частини клієнтського стану між переглядами сторінок може використовуватися Web Storage API, однак критичні дані замовлення після оформлення мають фіксуватися на сервері та в базі даних [27].

П'ятою вимогою є оформлення замовлення. Checkout повинен об'єднувати контактні дані, доставку, оплату, промокод і підсумок замовлення. Для доставки через Нову пошту користувач має обрати місто, відділення, поштомат або кур'єрський сценарій. Для самовивозу користувач обирає пункт видачі. Після підтвердження сервер повинен перевірити залишки товарів, створити замовлення, створити позиції замовлення та зменшити кількість товарів на складі.

Шостою вимогою є підтримка оплати. У системі передбачено різні способи оплати, зокрема оплату при отриманні та можливість подальшого підключення онлайн-оплати через зовнішній платіжний сервіс. Для онлайн-оплати важливо, щоб магазин не зберігав чутливі платіжні реквізити користувача. У базі даних повинні зберігатися лише спосіб оплати, статус оплати, сума та зв'язок із замовленням.

Сьомою вимогою є адміністративне керування системою. Адміністратор повинен мати доступ до списку товарів, користувачів і замовлень. Для товарів потрібні операції створення, редагування і видалення. Для замовлень потрібна зміна статусу: нове, підтверджено, комплектується, відправлено, доставлено або скасовано. Для автомобілів потрібне створення й використання записів сумісності, які потім застосовуються в каталозі й гаражі.

Нефункціональні вимоги визначають якість роботи системи. До них належать зручність інтерфейсу, надійність, безпека, продуктивність, підтримуваність і розширюваність. Такі характеристики відповідають загальній логіці оцінювання якості програмного продукту, де важливими є функціональна придатність, надійність, зручність використання, безпека, супроводжуваність і переносимість [32]. Інтерфейс повинен бути зрозумілим для користувача, який не має спеціальних технічних знань. Основні дії мають

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дат		

бути доступні без складної навігації: пошук товару, додавання до кошика, вибір доставки та оформлення замовлення.

Вимога безпеки стосується авторизації, паролів, ролей і зовнішніх API. Паролі повинні зберігатися тільки у вигляді хешів [18]. Google-вхід повинен перевірятися на сервері [21]. API-ключі зовнішніх сервісів не повинні розміщуватися у клієнтському коді. Адміністративні маршрути повинні бути доступні лише користувачу з відповідною роллю.

Вимога надійності особливо важлива для checkout. Замовлення не повинно створюватися частково. Якщо один із товарів відсутній або його недостатньо на складі, система повинна скасувати створення замовлення і повідомити користувача. Для цього операції створення замовлення, запису позицій і зміни залишків доцільно виконувати як єдиний процес із використанням можливостей реляційної бази даних, зокрема транзакцій і контролю цілісності даних [12].

Вимога підтримуваності означає, що система повинна мати зрозумілу структуру коду. Клієнтська частина розділена на сторінки та компоненти, а серверна частина реалізує окремі маршрути для різних груп операцій. Такий підхід спрощує подальше розширення: наприклад, у майбутньому можна додати нові фільтри, способи доставки, аналітику продажів або розширений модуль рекомендацій.

2.3. Постановка задачі розроблення системи

На основі аналізу предметної області та сформульованих вимог задача проєктування полягає у створенні вебзастосунку Auto Parts, який забезпечує повний цикл роботи онлайн-магазину автозапчастин: від перегляду каталогу до оформлення замовлення і подальшої адміністративної обробки. Система повинна бути реалізована як клієнт-серверний застосунок із розділенням інтерфейсу користувача, серверної бізнес-логіки та бази даних.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		31

Клієнтську частину доцільно реалізувати за допомогою React, оскільки інтерфейс магазину складається з багатьох повторюваних компонентів: карток товарів, форм, кнопок, списків, модальних вікон, блоків рейтингу та панелей фільтрів [4]. Для збирання клієнтської частини використовується Vite, який забезпечує швидкий запуск і зручну розробку сучасних вебзастосунків [6].

Серверну частину потрібно реалізувати на Node.js з використанням Express.js. Node.js дозволяє виконувати JavaScript на сервері, а Express.js забезпечує створення API-маршрутів і обробку HTTP-запитів [8], [10]. Сервер повинен приймати запити від клієнта, виконувати перевірки, працювати з базою даних, звертатися до зовнішніх API та повертати відповіді у форматі JSON.

Для збереження даних потрібно використати PostgreSQL. База даних повинна містити таблиці користувачів, тимчасових токенів, товарів, автомобільних відповідностей, гаража, замовлень, позицій замовлень, рейтингів і відгуків. PostgreSQL підходить для такої задачі, оскільки підтримує реляційні зв'язки, SQL-запити, обмеження цілісності й транзакції [12].

Задача проєктування передбачає реалізацію кількох основних модулів. Модуль авторизації повинен забезпечувати реєстрацію, вхід, відновлення пароля та Google-вхід. Модуль каталогу повинен надавати перегляд товарів, пошук, фільтри та сортування. Модуль товару повинен відображати детальну інформацію, рейтинг, відгуки та сумісність.

Модуль гаража повинен дозволяти користувачеві зберігати автомобілі. Модуль checkout повинен обробляти кошик, доставку, оплату та створення замовлення. Адміністративний модуль повинен забезпечувати керування товарами, автомобілями, користувачами й замовленнями.

Окремо потрібно реалізувати інтеграції із зовнішніми сервісами. Google Identity використовується для входу через Google-акаунт [21]. API Нової пошти використовується для отримання актуальних даних про міста, відділення і поштомати [23]. Для онлайн-оплати може бути підключений зовнішній платіжний сервіс, однак у межах основної логіки замовлення система зберігає тільки спосіб і статус оплати.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						32
Змн.	Арк.	№ докум.	Підпис	Дат		

Під час проєктування важливо врахувати обмеження навчального проєкту. Система повинна бути достатньо повною, щоб продемонструвати роботу реального онлайн-магазину, але при цьому не повинна перетворюватися на надмірно складну промислову платформу. Тому першочерговими є базові сценарії: каталог, товар, кошик, checkout, доставка, оплата, гараж, замовлення та адмін-панель. Розширені можливості, наприклад складна аналітика продажів або рекомендаційні алгоритми, можна розглядати як перспективу подальшого розвитку.

У результаті постановки задачі визначено, що розроблювана система повинна забезпечити:

- Зручний пошук і перегляд автозапчастин.
- Перевірку та використання даних про сумісність із автомобілями.
- Реєстрацію, авторизацію та профіль користувача.
- Кошик, обране та оформлення замовлення.
- Інтеграцію з доставкою Нової пошти.
- Підтримку способів оплати та можливість подальшого підключення онлайн-оплати.
- Адміністративне керування товарами, автомобілями, користувачами та замовленнями.
- Збереження основних даних у PostgreSQL.
- Безпечну роботу з паролями, ролями та зовнішніми API.

2.4. Висновки по розділу

У другому розділі було виконано аналіз вимог до вебзастосунку Auto Parts та сформульовано постановку задачі проєктування. Визначено дві основні ролі системи: користувач і адміністратор. Для користувача ключовими є пошук автозапчастин, перегляд товарів, гараж автомобілів, кошик, доставка, оплата та історія замовлень. Для адміністратора важливими є керування каталогом, автомобільними відповідностями, користувачами та замовленнями.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						33
Змн.	Арк.	№ докум.	Підпис	Дат		

Сформульовано функціональні вимоги, які описують основні можливості системи, а також нефункціональні вимоги, пов'язані із безпекою, надійністю, зручністю, підтримуваністю і розширюваністю. Найвищий пріоритет мають авторизація, каталог, пошук, кошик, checkout, доставка, підтримка оплати та адміністративне керування.

Окрему увагу приділено узгодженню вимог із реальними сценаріями користування онлайн-магазином автозапчастин. Для системи важливо не лише відобразити каталог товарів, а й забезпечити логічний шлях користувача від пошуку потрібної деталі до оформлення замовлення. Тому під час формування вимог враховано необхідність фільтрації та пошуку товарів, перевірки сумісності з автомобілем, збереження позицій у кошику та обраному, вибору способу доставки, обробки замовлень і підтримки актуальності даних через адміністративну панель.

На основі аналізу вимог визначено задачу проєктування: створити клієнт-серверний вебзастосунок онлайн-магазину автозапчастин із React/Vite клієнтом, Express/Node.js сервером і PostgreSQL базою даних. Також передбачено використання зовнішніх API: Google Identity, API Нової пошти та допоміжне використання SerpApi. Отримані результати цього розділу є основою для подальшого проєктування системи: розробки архітектури, моделі бази даних, структури серверних маршрутів і взаємодії.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		34

РОЗДІЛ 3. ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА МОДЕЛІ ДАНИХ ВЕБЗАСТОСУНКУ AUTO PARTS

3.1. Обґрунтування архітектури системи

Вебзастосунок Auto Parts спроектовано як клієнт-серверну систему, у якій інтерфейс користувача, серверна бізнес-логіка та база даних є окремими частинами. Такий підхід є доцільним для онлайн-магазину, оскільки користувач працює із сайтом через браузер, сервер перевіряє запити та виконує бізнес-правила, а база даних зберігає користувачів, товари, автомобільні відповідності, замовлення та службову інформацію.

Клієнтська частина реалізується на основі React та Vite. React використовується для побудови інтерфейсу з окремих компонентів, а Vite забезпечує швидкий запуск під час розроблення і підготовку production-збірки [4, 5, 6]. Для Auto Parts це важливо, оскільки сайт містить багато повторюваних елементів: картки товарів, фільтри, форми авторизації, блоки кошика, checkout, адміністративні панелі та компоненти вибору автомобіля.

Серверна частина побудована на Node.js та Express.js. Node.js використовується як середовище виконання JavaScript на сервері, а Express.js забезпечує створення маршрутів, обробку HTTP-запитів, роботу з проміжними обробниками та повернення відповідей у форматі JSON [8, 9, 10, 11]. Сервер виконує роль центрального шару: приймає запити від клієнта, перевіряє дані, працює з PostgreSQL, звертається до зовнішніх сервісів і формує відповідь для інтерфейсу.

Взаємодія між клієнтом і сервером організовується через HTTP-запити. Для системи важливо дотримуватися зрозумілої семантики маршрутів: отримання даних виконується через GET, створення через POST, оновлення через PUT або PATCH, а видалення через DELETE. Такий підхід відповідає загальним принципам HTTP та REST-архітектури [1, 2, 10, 40]. У проєкті

					БР.ІІІ - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		35

ресурсами є користувачі, товари, замовлення, автомобілі, гараж, відгуки та службові дані доставки.

База даних реалізується на PostgreSQL. Вона зберігає основні сутності предметної області та підтримує реляційні зв'язки між ними: користувач може мати кілька автомобілів у гаражі, товар може підходити до багатьох автомобілів, а замовлення складається з кількох позицій [12, 13, 15]. На рисунку 3.1 наведено загальну архітектуру вебзастосунку Auto Parts.

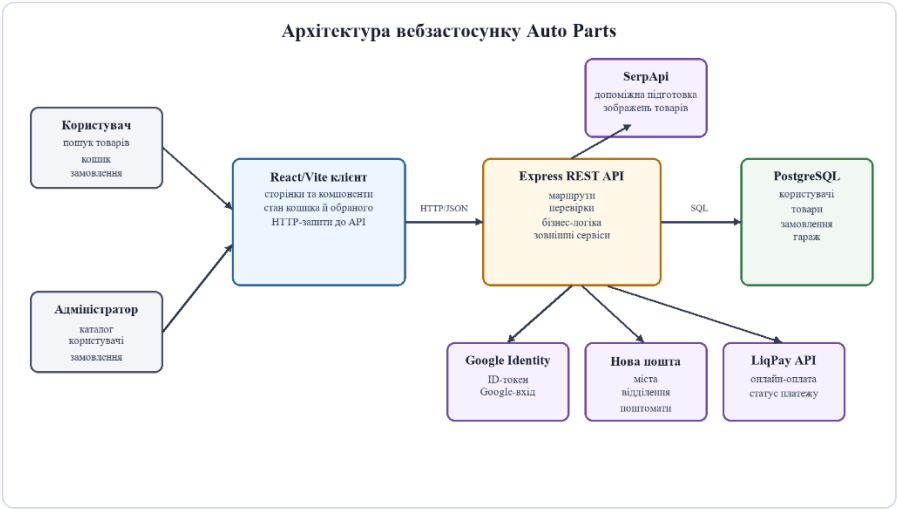


Рисунок 3.1 – Архітектура вебзастосунку Auto Parts.

З рисунка 3.1 видно, що клієнтська частина є точкою взаємодії користувача та адміністратора із системою. Користувач переглядає каталог, працює з кошиком, обирає доставку й оформляє замовлення. Адміністратор через вебінтерфейс керує товарами, автомобільними відповідностями, користувачами та замовленнями.

Окреме місце в архітектурі займають зовнішні сервіси. Google Identity використовується для входу через Google-акаунт і перевірки ID-токена на серверній стороні [21]. API Нової пошти використовується для отримання міст, відділень і поштоматів під час оформлення доставки [23]. Для онлайн-оплати в архітектурі передбачено можливість подальшого підключення зовнішнього

платіжного сервісу. SerpApi застосовується як допоміжний сервіс для підготовки зображень товарів у каталозі [25].

Система також враховує обмеження безпеки вебвзаємодії. Клієнт і сервер працюють як окремі частини, тому для обміну між ними використовується CORS, а дані передаються у форматі JSON [26, 28]. API-ключі зовнішніх сервісів мають зберігатися на сервері або в змінних середовища [20, 37]. Паролі користувачів зберігаються не у відкритому вигляді, а як хеші, що відповідає рекомендаціям OWASP і документації bcrypt [18, 19, 36].

3.2. Проєктування програмних модулів і API-взаємодії

Архітектура Auto Parts складається з кількох програмних модулів, кожен із яких відповідає за окрему частину функціональності. Такий поділ спрощує підтримку системи, оскільки авторизація, каталог, checkout, гараж і адміністративна панель можуть розвиватися окремо. У таблиці 3.1 наведено основні програмні модулі вебзастосунку Auto Parts.

Таблиця 3.1

Основні програмні модулі вебзастосунку

Модуль	Призначення
Модуль навігації та загального інтерфейсу	Забезпечує переходи між сторінками, роботу шапки сайту, футера, теми оформлення та повідомлень.
Модуль авторизації	Реалізує реєстрацію, вхід, підтвердження email, відновлення пароля, вихід з акаунта та Google-вхід.
Модуль каталогу	Відображає товари, категорії, фільтри, сортування, пошук і популярні позиції.
Модуль картки товару	Показує характеристики, зображення, сумісність, рейтинг, відгуки та додавання товару до кошика або обраного.
Модуль гаража	Дозволяє користувачеві зберігати автомобілі та використовувати їх для перевірки сумісності деталей.
Модуль кошика й обраного	Зберігає вибрані товари, кількість позицій, промокод і підготовку до оформлення замовлення.
Модуль checkout	Об'єднує контактні дані, доставку, оплату, перевірку залишків і створення замовлення.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		37

Адміністративний модуль	Забезпечує керування товарами, автомобілями, користувачами, замовленнями та статусами виконання.
Модуль зовнішніх інтеграцій	Забезпечує взаємодію з Google Identity, API Нової пошти та SerpApi.

На клієнтській стороні основні сторінки представлені компонентами Home, Catalog, ProductDetails, Cart, Checkout, Favorites, Garage, Orders, Account, Admin та AuthPage. Кожна сторінка відповідає окремому користувацькому сценарію. Наприклад, Catalog відповідає за пошук і фільтрацію товарів, ProductDetails – за детальний перегляд товару, Checkout – за оформлення замовлення, а Admin – за службові дії адміністратора.

Навігація в клієнтській частині реалізується через зміну поточного шляху сторінки та відображення відповідного компонента. Це дозволяє зробити застосунок схожим на багатосторінковий сайт, хоча сторінки відображаються в межах одного React-застосунку [4, 5].

Для обміну даними клієнт використовує HTTP-запити до серверних маршрутів. У браузері такі запити виконуються через Fetch API [26]. Частина клієнтського стану зберігається локально: тема, кошик, обрані товари, промокод і дані поточного користувача. Для цього може використовуватися Web Storage API [27]. Водночас замовлення, профіль, гараж, статуси оплати та залишки товарів мають зберігатися на сервері.

Серверні маршрути поділяються на кілька груп. Перша група відповідає за авторизацію: /register, /login, /verify-email, /email-verification/resend, /password-reset/request, /password-reset/confirm та /google-login. Ці маршрути обробляють створення облікового запису, вхід за email і паролем, підтвердження пошти, відновлення доступу та Google-вхід. Під час реєстрації пароль перетворюється на хеш, а під час входу сервер порівнює введений пароль із збереженим хешем [18, 19].

Друга група маршрутів пов'язана з каталогом товарів: /products, /products/popular, /products/search, /products/:id, /products/:id/rating та /products/:id/reviews. Вони забезпечують отримання списку товарів, пошук,

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		38

перегляд детальної інформації, рейтинг і відгуки. Для адміністратора також передбачено створення, редагування та видалення товарів [10, 11, 40].

Третя група маршрутів описує роботу з автомобілями та гаражем користувача. Маршрут /vehicles повертає доступні автомобільні відповідності, а маршрути /users/:id/garage дозволяють додавати, переглядати й видаляти автомобілі з гаража [12, 15].

Четверта група маршрутів відповідає за замовлення і доставку. Маршрут /checkout створює замовлення, перевіряє кошик, зменшує залишки товарів і записує позиції замовлення. Маршрути /users/:id/orders, /users/:id/orders/:orderId/cancel, /orders і /orders/:id/status використовуються для історії замовлень, скасування та адміністративної обробки [1, 10, 12, 14].

Окремо виділяються маршрути доставки /nova-poshta/cities та /nova-poshta/warehouses. Вони звертаються до API Нової пошти і повертають клієнтській частині нормалізований список міст, відділень або поштоматів [23]. Такий підхід дозволяє приховати API-ключ, контролювати формат відповіді та обробляти помилки зовнішнього сервісу [11, 28, 37].

Для обробки помилок сервер повертає HTTP-статуси та JSON-повідомлення. Наприклад, помилка введення даних має повертатися як 400, відсутність прав доступу – як 403, відсутність ресурсу – як 404, а внутрішня помилка сервера – як 500 [3, 11, 40].

3.3. Проєктування структури бази даних

База даних Auto Parts побудована навколо основних сутностей інтернет-магазину автозапчастин: користувачів, товарів, автомобільних відповідей, гаража, замовлень, позицій замовлення, рейтингів і відгуків. Окремо зберігаються службові токени для підтвердження email або відновлення пароля. На рисунку 3.2 наведено логічну модель бази даних Auto Parts.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						39
Змн.	Арк.	№ докум.	Підпис	Дат		

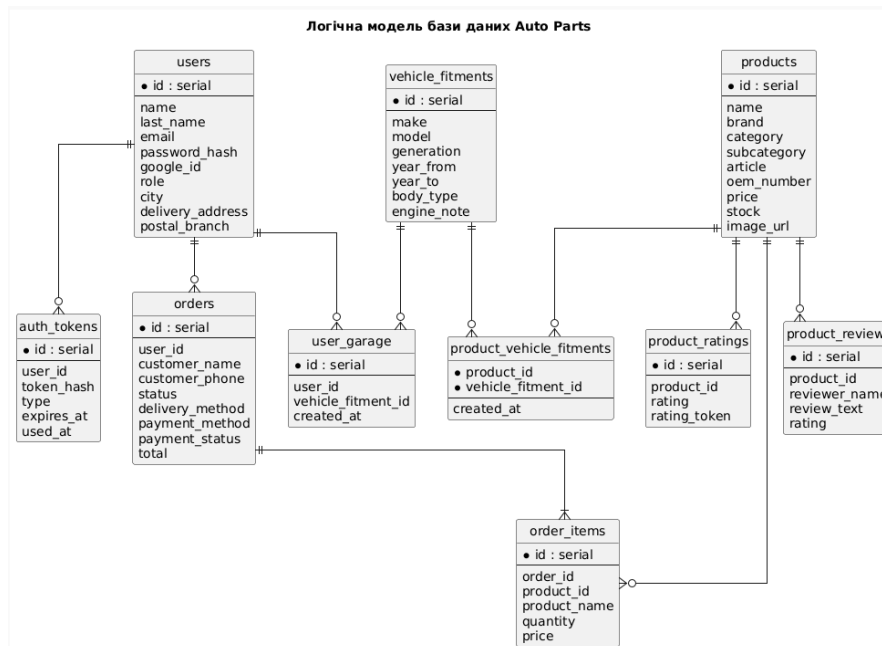


Рисунок 3.2 – Логічна модель бази даних Auto Parts.

Сутність `users` зберігає інформацію про зареєстрованих користувачів: ім'я, прізвище, email, телефон, хеш пароля, Google ID, роль і дані доставки. Поле ролі дозволяє відрізнити звичайного користувача від адміністратора та обмежувати доступ до службових функцій [18, 19, 20].

Сутність `auth_tokens` використовується для тимчасових токенів. Вона містить ідентифікатор користувача, хеш токена, тип, дату завершення дії та позначку використання. Зберігання саме хешу токена є безпечнішим, ніж зберігання відкритого значення [18, 19].

Сутність `vehicle_fitments` описує автомобілі, для яких можуть підходити товари: марку, модель, покоління, роки випуску, тип кузова та примітку щодо двигуна. Для уникнення дублювання використовується унікальне обмеження, а для швидшого пошуку створено індекс [12, 15].

Сутність `user_garage` пов'язує користувача з його автомобілями. Один користувач може мати кілька автомобілів, а один запис автомобільної ідповідності може використовуватися багатьма користувачами. Тому цей зв'язок реалізовано через проміжну таблицю.

Сутність `products` є центральною для каталогу. Вона зберігає назву товару, бренд, категорію, підкатегорію, артикул, OEM-номер, ціну, залишок,

зображення, опис, характеристики, гарантію, країну походження та стан товару. Для магазину автозапчастин важливо зберігати не лише опис, а й технічні параметри, оскільки користувач приймає рішення на основі сумісності, артикулу, бренду й характеристик.

Зв'язок між товарами та автомобілями реалізовано через таблицю `product_vehicle_fitments`. Один товар може підходити до кількох автомобілів, а один автомобіль може мати багато сумісних товарів. Первинним ключем є пара `product_id` і `vehicle_fitment_id`, що не дозволяє дублювати одну й ту саму відповідність.

Таблиці `product_ratings` і `product_reviews` відповідають за оцінки та відгуки. Рейтинг зберігається як числове значення від 1 до 5, а відгук містить ім'я автора, текст, дату створення та за потреби оцінку.

Сутності `orders` і `order_items` описують оформлені покупки. Таблиця `orders` зберігає контактні дані, статус, доставку, оплату, знижку та суму. Таблиця `order_items` зберігає склад замовлення: товар, назву товару на момент покупки, бренд, артикул, кількість, ціну та зображення. Збереження копії назви й ціни потрібне тому, що товар у каталозі може змінитися після оформлення покупки.

Під час створення замовлення важливо не допустити ситуації, коли два користувачі одночасно купують останню одиницю одного товару. Для цього сервер використовує транзакцію та блокує вибрані товари через `SELECT ... FOR UPDATE`. Якщо на будь-якому етапі виникає помилка, транзакція скасовується [12, 14, 15].

Для підключення до PostgreSQL використовується бібліотека `nod-postgres`, яка надає механізм пулу з'єднань. Пул дозволяє повторно використовувати відкриті з'єднання з базою даних і не створювати нове підключення для кожного HTTP-запиту [16, 17].

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дат		

3.4. Проектування сценарію оформлення замовлення та інтеграцій

Оформлення замовлення є найважливішим бізнес-сценарієм системи, оскільки саме він перетворює перегляд товарів у завершену покупку. У цьому сценарії поєднуються клієнтський стан, серверні перевірки, база даних, зовнішній сервіс доставки та платіжна інтеграція. На рисунку 3.3 наведено послідовність оформлення замовлення в системі Auto Parts.

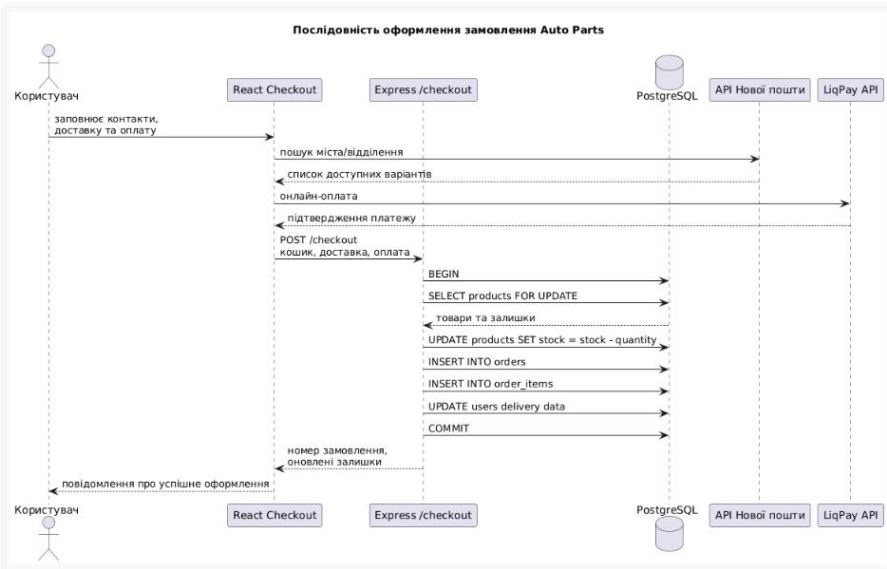


Рисунок 3.3 – Діаграма послідовності оформлення замовлення Auto Parts.

Користувач починає сценарій із кошика. Кошик може зберігатися в локальному сховищі браузера, щоб не зникати після переходу між сторінками або оновлення сторінки [27]. Однак сервер не повинен повністю довіряти даним із клієнта, тому під час оформлення замовлення він повторно перевіряє товари, ціну та наявність на складі [12, 14, 20].

Під час вибору доставки клієнтська частина звертається до серверних маршрутів Нової пошти. Користувач вводить місто або обирає тип доставки, після чого система отримує список доступних міст, відділень або поштоматів [23]. Сервер нормалізує відповідь зовнішнього API і повертає клієнту дані у зручному форматі.

Оплата може виконуватися кількома способами: при отриманні, переказом або онлайн після підключення зовнішнього платіжного сервісу. Для онлайн-оплати система повинна передати користувача в платіжний сценарій, отримати результат оплати та зберегти в замовленні спосіб і статус оплати. При цьому магазин не повинен зберігати повні платіжні реквізити користувача [18, 20].

Після підтвердження форми checkout клієнт надсилає на сервер запит POST /checkout. До запиту входять позиції кошика, ідентифікатор користувача, контактні дані, спосіб доставки, місто, відділення або адреса, спосіб оплати, знижка та коментар. Сервер перевіряє обов'язкові поля та правила доставки.

Далі сервер відкриває транзакцію в PostgreSQL. Для кожного товару виконується вибірка з блокуванням рядка, перевіряється актуальний залишок, зменшується кількість товару, створюється запис у таблиці orders і додаються записи до order_items. Завершення операції відбувається через COMMIT; у разі помилки виконується ROLLBACK [12, 14].

Під час створення позицій замовлення система зберігає не лише product_id, а й назву товару, бренд, артикул, ціну та зображення. Це потрібно для стабільності історії замовлень: якщо адміністратор пізніше змінить дані товару в каталозі, старе замовлення все одно показуватиме інформацію, актуальну на момент покупки.

Скасування замовлення також потребує контролю цілісності даних. Якщо статус дозволяє скасування, сервер відкриває транзакцію, блокує запис замовлення, повертає кількість товарів на склад, змінює статус на cancelled і фіксує зміни. Якщо скасування заборонене, система повертає повідомлення про неможливість операції [12, 14, 40].

Проектування інтеграцій виконано так, щоб зовнішні сервіси не замінювали внутрішню логіку магазину, а лише доповнювали її. Нова пошта надає дані для доставки, Google Identity допомагає авторизувати користувача, а SerpApi використовується для допоміжної підготовки зображень каталогу [21,

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						43
Змн.	Арк.	№ докум.	Підпис	Дат		

23, 25]. Основні дані магазину залишаються у PostgreSQL і контролюються серверною частиною Auto Parts.

3.5. Висновки по розділу

У третьому розділі спроектовано архітектуру вебзастосунку Auto Parts. Система побудована як клієнт-серверний застосунок, у якому React/Vite відповідає за інтерфейс користувача, Express/Node.js реалізує серверну бізнес-логіку, а PostgreSQL зберігає основні дані предметної області.

Описано основні програмні модулі системи: авторизацію, каталог, картку товару, гараж, кошик, checkout, замовлення, адміністративну панель і зовнішні інтеграції. Для серверної частини визначено групи API-маршрутів, які забезпечують роботу з користувачами, товарами, автомобілями, замовленнями, доставкою та адміністративними операціями.

Спроектовано логічну модель бази даних. Основними таблицями є users, auth_tokens, vehicle_fitments, user_garage, products, product_vehicle_fitments, product_ratings, product_reviews, orders та order_items. Для підтримки цілісності даних передбачено первинні ключі, зовнішні ключі, унікальні обмеження, індекси та транзакційне виконання критичних операцій.

Окремо розглянуто сценарій оформлення замовлення. Він включає перевірку кошика, вибір доставки через Нову пошту, вибір способу оплати, створення замовлення, запис позицій замовлення, оновлення складських залишків і фіксацію операції в межах транзакції.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		44

РОЗДІЛ 4. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ AUTO PARTS

4.1. Структура проєкту та підготовка середовища

Програмну реалізацію вебзастосунку Auto Parts виконано у вигляді окремих клієнтської та серверної частин. Клієнтська частина розміщена в каталозі `src` і містить сторінки, компоненти, стилі та допоміжні функції. Серверна частина розміщена в каталозі `server` і містить файл запуску API, підключення до бази даних, початкові дані каталогу, дані автомобілів і службові скрипти. Така структура відповідає клієнт-серверній архітектурі, описаній у попередньому розділі, і дозволяє розділити інтерфейс користувача, серверну логіку та роботу з PostgreSQL.

Для керування залежностями використано `npm`. У клієнтській частині налаштовано команди `dev`, `build`, `lint` і `preview`, що дозволяє запускати застосунок під час розроблення, створювати production-збірку, перевіряти код та переглядати зібраний варіант [6, 38]. Серверна частина має окремий `package.json`, у якому зазначено залежності для Express, PostgreSQL, `bcrypt`, `dotenv`, CORS і Google Auth Library [8, 10, 16, 21, 36, 37]. Лістинг 4.1 демонструє основні команди клієнтської частини.

Лістинг 4.1 – Команди запуску та збирання клієнтської частини.

```
"scripts": {  
  "dev": "vite",  
  "build": "vite build",  
  "lint": "eslint .",  
  "preview": "vite preview"  
}
```

Підключення до PostgreSQL винесено в окремий файл server/db.js. У ньому використовується пул з'єднань Pool з бібліотеки pg. Такий підхід дозволяє повторно використовувати підключення до бази даних, а не створювати нове з'єднання для кожного запиту [16, 17]. Параметри підключення беруться зі змінних середовища, що дозволяє не зберігати конфігурацію напряму в коді [37]. Лістинг 4.2 демонструє реалізацію підключення серверної частини до бази даних PostgreSQL.

Лістинг 4.2 – Підключення серверної частини до PostgreSQL.

```
const { Pool } = require('pg');
const path = require('path');
require('dotenv').config({ path: path.join(__dirname, '..', '.env'), quiet: true });

const pool = new Pool({
  user: process.env.DB_USER || process.env.PGUSER || 'postgres',
  host: process.env.DB_HOST || process.env.PGHOST || 'localhost',
  database: process.env.DB_NAME || process.env.PGDATABASE || 'autoparts_db',
  password: process.env.DB_PASSWORD || process.env.PGPASSWORD || '',
  port: Number(process.env.DB_PORT || process.env.PGPORT || 5432),
});

module.exports = pool;
```

Під час запуску сервер створює необхідні таблиці, якщо вони ще не існують: users, auth_tokens, vehicle_fitments, user_garage, products, product_vehicle_fitments, product_ratings, product_reviews, orders та order_items. Для підтримки цілісності даних використовуються первинні ключі, зовнішні ключі, унікальні обмеження та індекси [12, 15]. Це спрощує розгортання навчального проєкту, оскільки база даних може підготуватися автоматично після запуску сервера.

4.2. Реалізація клієнтської частини

Клієнтська частина Auto Parts реалізована на React. Основний компонент App відповідає за навігацію між сторінками, збереження поточного користувача, підрахунок товарів у кошику, кількість обраних товарів, тему оформлення та відображення повідомлень. Для маршрутизації між сторінками використано бібліотеку react-router-dom, що дозволяє описати сторінки як окремі маршрути та підтримувати пряме відкриття адрес каталогу, кошика, оформлення замовлення, профілю, адміністративної панелі й сторінки конкретного товару [4, 5]. Лістинг 4.3 демонструє фрагмент налаштування маршрутів клієнтської частини.

Лістинг 4.3 – Налаштування маршрутів клієнтської частини вебзастосунку.

```
<Routes>
  <Route path="/" element={<Home currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/catalog" element={<Catalog currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/cart" element={<Cart currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/checkout" element={<Checkout currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/favorites" element={<Favorites onNavigate={navigate} />} />
  <Route path="/garage" element={<Garage currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/account" element={<Account currentUser={currentUser} onNavigate={navigate} />} />
  <Route path="/admin" element={<Admin currentUser={currentUser} onNavigate={navigate} />} />
  <Route
    path="/product/:productId"
    element={
      <ProductDetailsRoute
        currentUser={currentUser}
        onNavigate={navigate}
      />
    }
  />
```

```

    onCartChange={refreshCartCount}
    onFavoritesChange={refreshFavoriteCount}
    onToast={showToast}
  />
  )}
/>
</Routes>

```

Для збереження частини клієнтського стану використовується `localStorage`: кошик, обрані товари, тема оформлення, промокод і дані поточного користувача. Такий підхід дозволяє не втрачати дані після оновлення сторінки або переходу між розділами сайту [27]. Водночас критичні дані, зокрема замовлення, профіль користувача, гараж і залишки товарів, зберігаються на сервері та в PostgreSQL.

Одним із важливих модулів клієнтської частини є каталог. Він забезпечує перегляд товарів, фільтрацію за категоріями, пошук за назвою, брендом, артикулом, OEM-номером і автомобільною сумісністю. У каталозі реалізовано нормалізацію пошукового запиту, розбиття на токени та приблизне порівняння через відстань Левенштейна. Це підвищує зручність пошуку, оскільки користувач може вводити неповну назву товару або частину артикулу.

Лістинг 4.4 демонструє фрагмент нормалізації пошукового запиту та перевірки відповідності товару автомобілю користувача.

Лістинг 4.4 – Фрагмент пошуку та перевірки сумісності товару з автомобілем.

```

const normalizeSearchValue = (value) => String(value || "")
  .toLowerCase()
  .replace(/['"]+/g, "")
  .replace(/^[^p{L}\p{N}]+/gu, '')
  .trim();

const productMatchesGarageVehicle = (product, vehicle) => {
  const selectedVehicleId = Number(vehicle.vehicle_fitment_id || vehicle.id);

```

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		48

```

const productFitments = Array.isArray(product.vehicle_fitments) ?
product.vehicle_fitments : [];

return productFitments.some((fitment) => (
  Number(fitment.id || fitment.vehicle_fitment_id) === selectedVehicleId
));
};

```

У наведеному фрагменті порожній рядок у виклику `.replace(/['"]/g, '')` означає видалення апострофів і лапок із пошукового запиту. Наступний виклик `.replace(/[^p{L}\p{N}]+/gu, ' ')` замінює всі символи, які не є літерами або цифрами, на пробіл. Це потрібно для приведення пошукового запиту до єдиного формату перед порівнянням із даними товарів.

Окремо реалізовано сторінку товару. Вона завантажує детальну інформацію про товар, галерею зображень, рейтинг, відгуки, сумісні автомобілі та пов'язані товари. Користувач може додати товар до кошика або обраного, оцінити його та залишити відгук. Для взаємодії з сервером використовуються HTTP-запити через Fetch API [26].

Для модуля гаража реалізовано сторінку Garage. Вона завантажує список доступних автомобільних відповідей і список автомобілів користувача. Користувач може обрати марку та модель, додати автомобіль у гараж або видалити його. Цей модуль пов'язаний із каталогом: якщо користувач має збережений автомобіль, система може перевіряти сумісність товарів із цим автомобілем.

4.3. Реалізація серверної частини та авторизації

Серверна частина реалізована на Express.js. Вона приймає HTTP-запити від клієнтської частини, виконує перевірку даних, звертається до PostgreSQL, працює із зовнішніми API та повертає відповіді у форматі JSON [8, 10, 11]. Для

обміну між клієнтом і сервером використовується CORS, а самі запити передаються у форматі JSON [26, 28].

Авторизація реалізована через реєстрацію, вхід за email і паролем, підтвердження email, відновлення пароля та Google-вхід. Під час реєстрації пароль не зберігається відкрито: сервер створює salt і формує хеш за допомогою bcrypt. Під час входу введений пароль порівнюється із хешем у базі даних. Такий підхід відповідає рекомендаціям OWASP щодо зберігання паролів [18, 19, 36]. Лістинг 4.5 демонструє створення хешу пароля перед збереженням користувача в базі даних.

Лістинг 4.5 – Створення хешу пароля під час реєстрації.

```
const salt = await bcrypt.genSalt(10);
const passwordHash = await bcrypt.hash(password, salt);

const newUser = await pool.query(
  `INSERT INTO users
    (name, last_name, email, phone, password_hash, role, email_verified,
    email_verified_at)
    VALUES ($1, $2, $3, $4, $5, $6, FALSE, NULL)
    RETURNING ${userSelect}` ,
  [name, lastName, normalizedEmail, phone || null, passwordHash, role]
);
```

Для входу через Google на сервері використовується перевірка ID-токена. Клієнтська частина отримує credential від Google Identity Services, після чого надсилає його на сервер. Сервер перевіряє токен, отримує email користувача і або знаходить наявний обліковий запис, або створює новий [21, 22]. Такий підхід є безпечнішим, ніж довіряти тільки даним, отриманим на клієнтській стороні.

Сервер також реалізує групу маршрутів для каталогу: отримання всіх товарів, популярних позицій, пошук, перегляд одного товару, додавання

					БР.ІІІ - 05.00.00.000 ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дат		

рейтингу та відгуків. Для адміністратора передбачено створення, редагування й видалення товарів. Адміністративні маршрути перевіряють роль користувача через службовий заголовок, тому звичайний користувач не повинен мати доступу до зміни каталогу [20].

У серверній частині реалізовано роботу з автомобілями. Маршрут `/vehicles` повертає перелік автомобільних відповідностей, а маршрути `/users/:id/garage` дають змогу працювати з гаражем користувача. Це важлива частина саме для магазину автозапчастин, оскільки система повинна враховувати сумісність товару з конкретною моделлю автомобіля.

4.4. Реалізація каталогу, гаража та адміністративної панелі

Каталог є одним із центральних елементів Auto Parts. Він відображає товари з назвою, брендом, ціною, категорією, зображенням, рейтингом і наявністю. Користувач може шукати товар, обирати категорію, фільтрувати результати, відкривати швидкий перегляд, переходити на детальну сторінку товару, додавати позицію до кошика або в обране. Для клієнтських запитів до каталогу використовується Fetch API [26].

Гараж користувача доповнює каталог. Якщо користувач зберігає свій автомобіль, система може показувати, чи підходить товар до цього автомобіля. Це реалізовано через таблиці `vehicle_fitments`, `user_garage` і `product_vehicle_fitments`. Така модель дозволяє не записувати сумісність у вигляді довгого тексту в кожному товарі, а підтримувати окремі структуровані зв'язки в базі даних [12, 15].

Адміністративна панель реалізована як окрема сторінка Admin. Вона завантажує користувачів, товари, замовлення та автомобільні відповідності. Адміністратор може додавати товар, редагувати його, видаляти позицію з каталогу, додавати автомобільну відповідність і змінювати статус замовлення. Захищені запити адміністративної панелі передають session token у заголовок Authorization, тому сервер перевіряє права користувача не за довільним

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		51

значенням ролі з клієнта, а за підписаним токеном. Це дозволяє підтримувати актуальність каталогу без ручної роботи з базою даних і зменшує ризик несанкціонованого доступу [20]. Лістинг 4.6 демонструє надсилання даних товару з адміністративної панелі на сервер.

Лістинг 4.6 – Збереження товару через адміністративну панель.

```
const response = await fetch(url, {  
  method: isEditing ? 'PUT' : 'POST',  
  headers: authHeaders({ 'Content-Type': 'application/json' }),  
  body: JSON.stringify(productForm),  
});  
  
const data = await response.json();
```

Після успішного збереження товару клієнтська частина оновлює локальний список товарів: якщо товар редагувався, він замінюється в масиві, а якщо створювався новий – додається на початок списку. Це дозволяє одразу показати зміни адміністратору без перезавантаження сторінки.

Взаємодія каталогу, гаража та адміністративної панелі формує єдиний цикл роботи з товарами в системі Auto Parts. Користувач бачить у каталозі актуальні дані, які підтримуються адміністратором, а збережений у гаражі автомобіль дає змогу швидше оцінити сумісність деталі перед додаванням її до кошика.

Усі зміни, внесені через адміністративну панель, впливають на клієнтську частину без потреби прямого редагування бази даних. Такий підхід підвищує зручність користування системою, зменшує ймовірність помилкового вибору товару та спрощує подальше обслуговування каталогу. На рисунку 4.1 наведено сторінку каталогу Auto Parts, де користувач може переглядати товари, використовувати пошук, фільтри, сортування, додавати позиції до кошика або зберігати їх в обране.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дат		

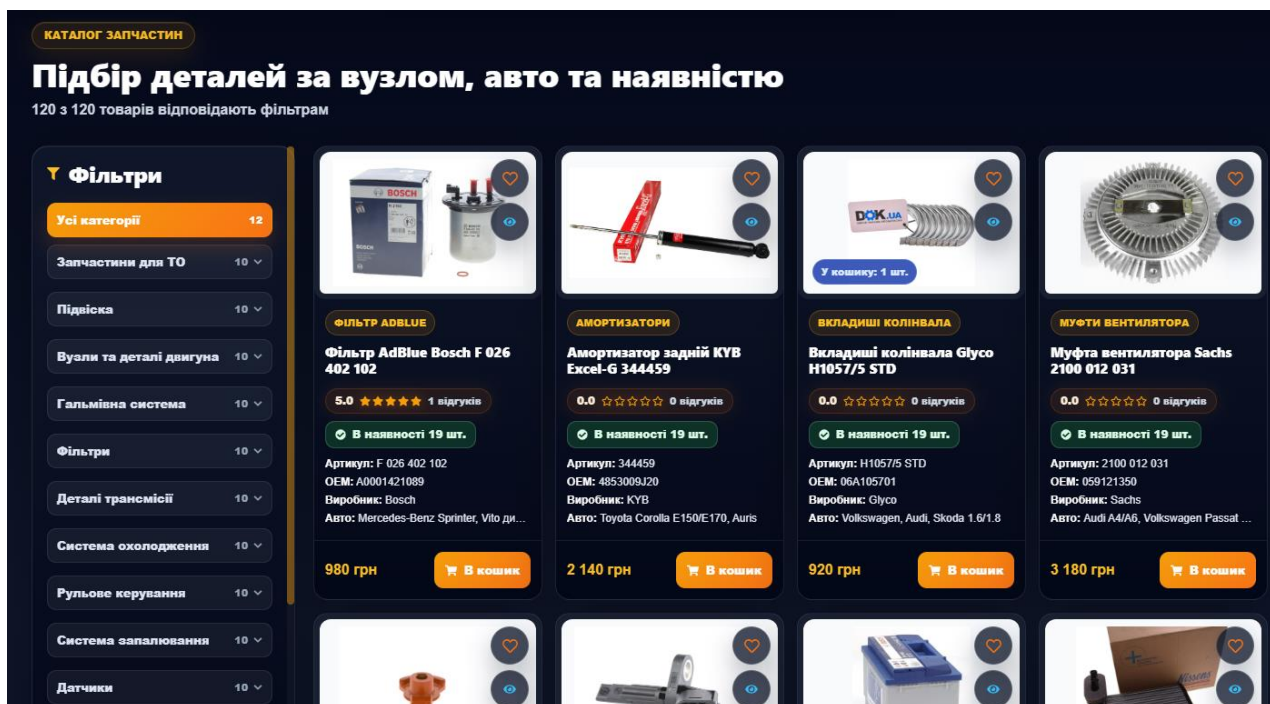
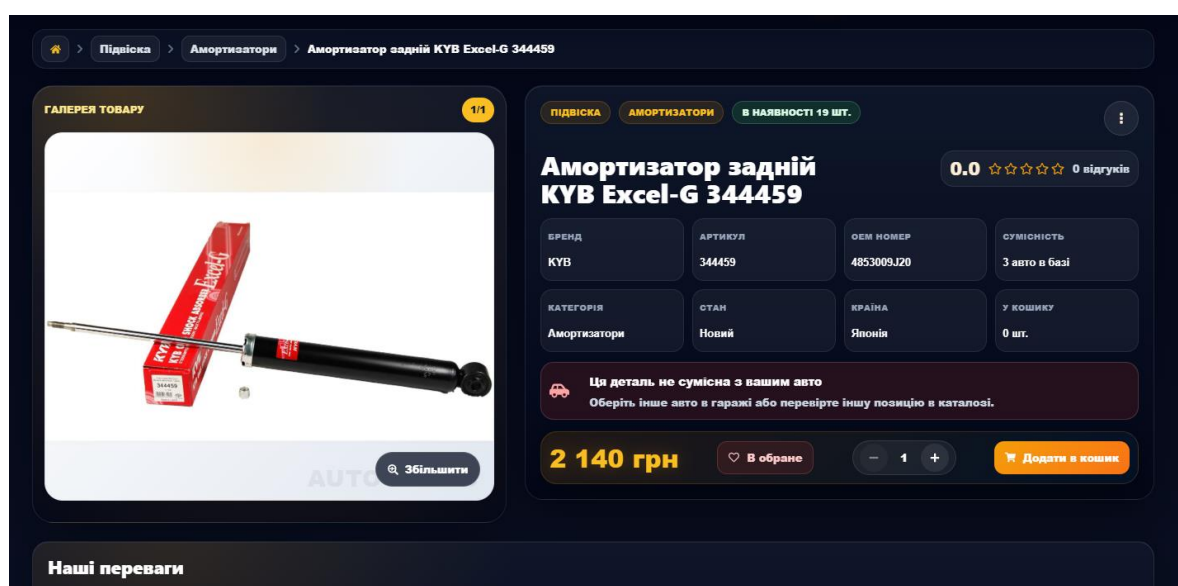


Рисунок 4.1 – Сторінка каталогу автозапчастин із фільтрами та товарами.

На рисунку 4.2 наведено детальну сторінку товару. На ній відображаються зображення товару, назва, бренд, ціна, артикул, OEM-номер, характеристики, рейтинг, відгуки та інформація про сумісність з автомобілями. Така сторінка допомагає користувачеві прийняти рішення перед додаванням товару до кошика.



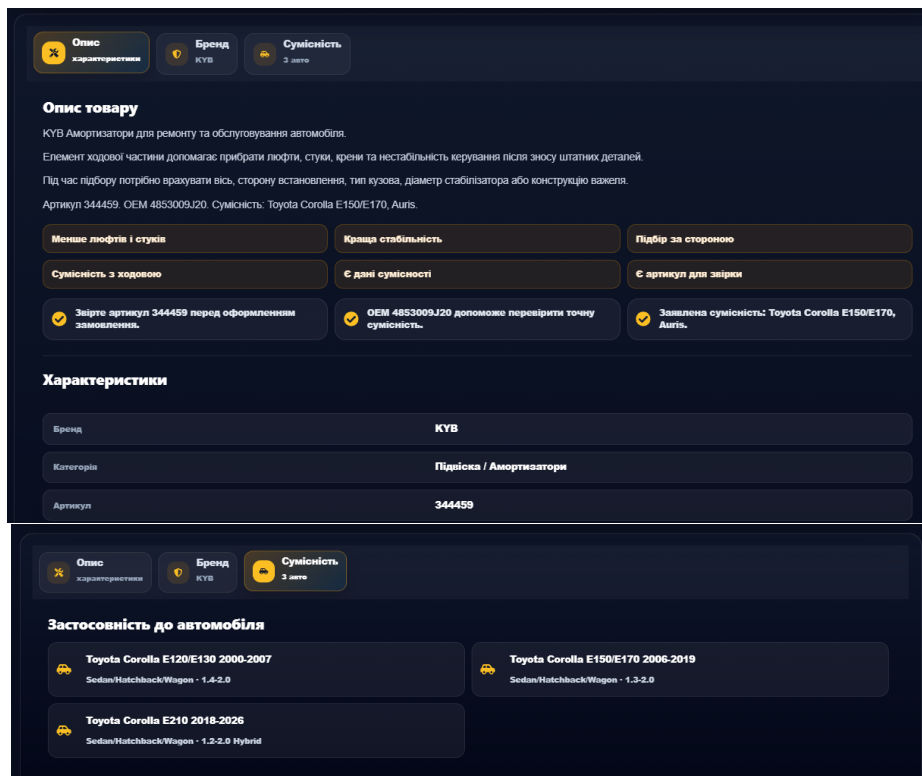


Рисунок 4.2 – Сторінка перегляду характеристик і сумісності товару.

На рисунку 4.3 наведено адміністративну панель вебзастосунку. Вона використовується для керування товарами, автомобільними відповідностями, користувачами та замовленнями. Через цю сторінку адміністратор може додавати нові товари, редагувати наявні позиції, видаляти їх із каталогу та змінювати статуси замовлень.

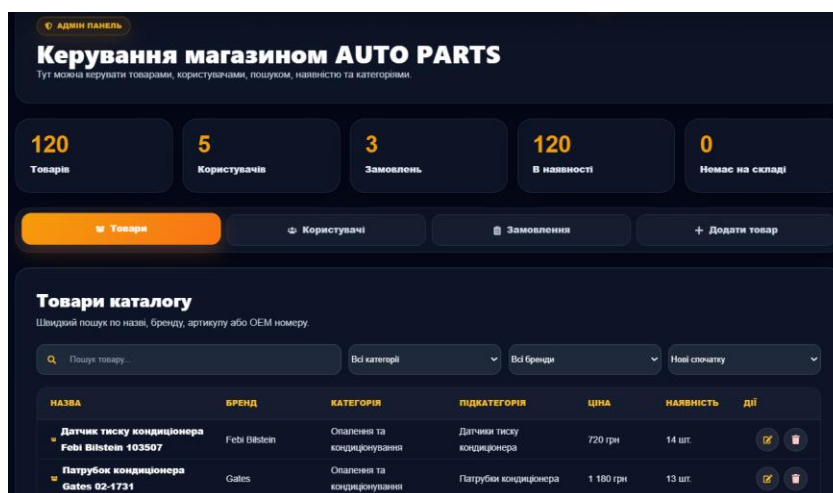


Рисунок 4.3 – Панель адміністратора для керування товарами та замовленнями.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дат		

На рисунку 4.4 наведено головну сторінку вебзастосунку Auto Parts. Вона є початковою точкою взаємодії користувача із сайтом і містить перехід до каталогу, гаража, популярних товарів та основних розділів магазину.

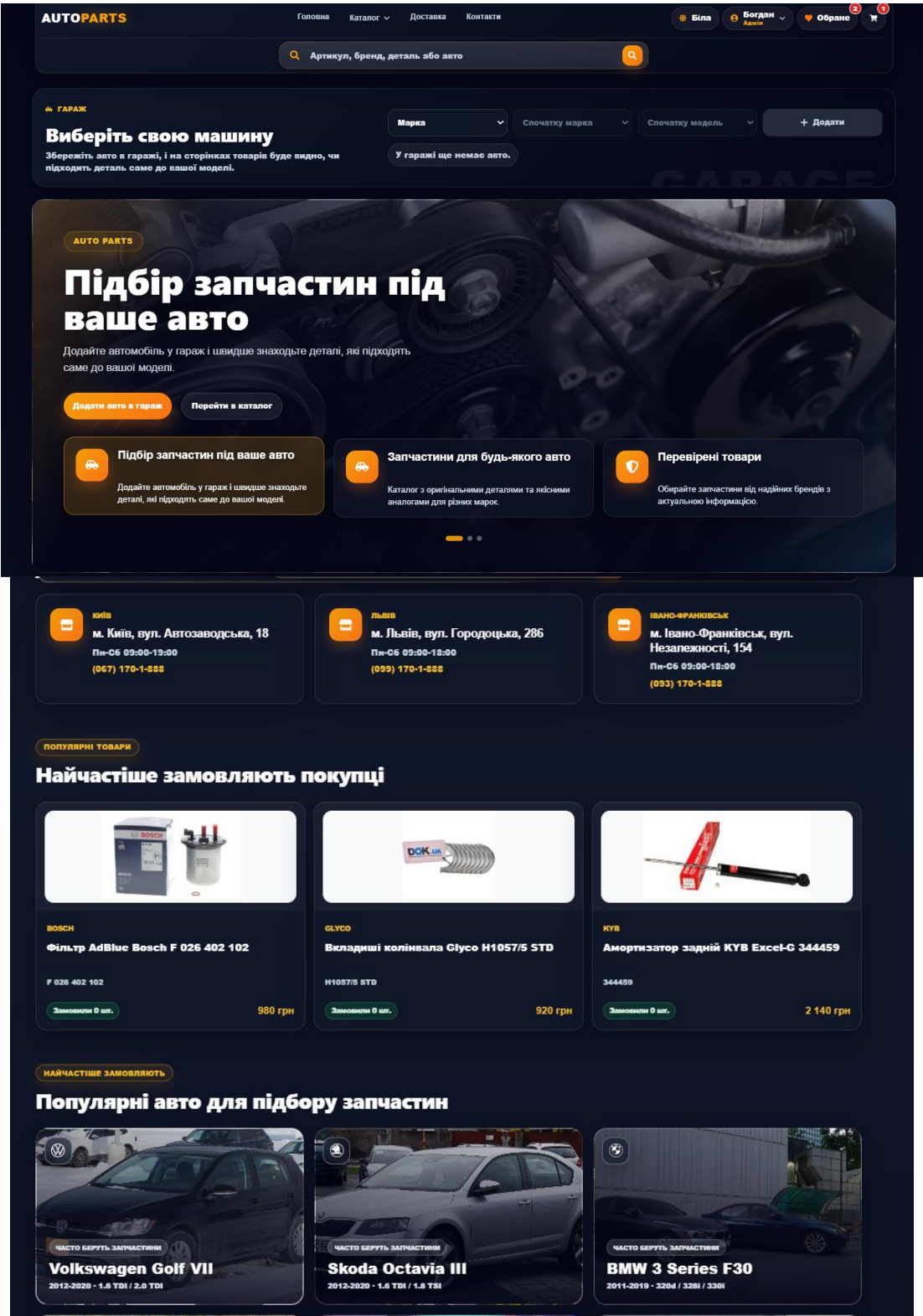


Рисунок 4.4 – Головна сторінка вебзастосунку Auto Parts.

На рисунку 4.5 наведено сторінку гаража користувача. У цьому розділі користувач може додати свій автомобіль, обрати марку та модель, переглянути збережені автомобілі й надалі використовувати ці дані для перевірки сумісності автозапчастин.

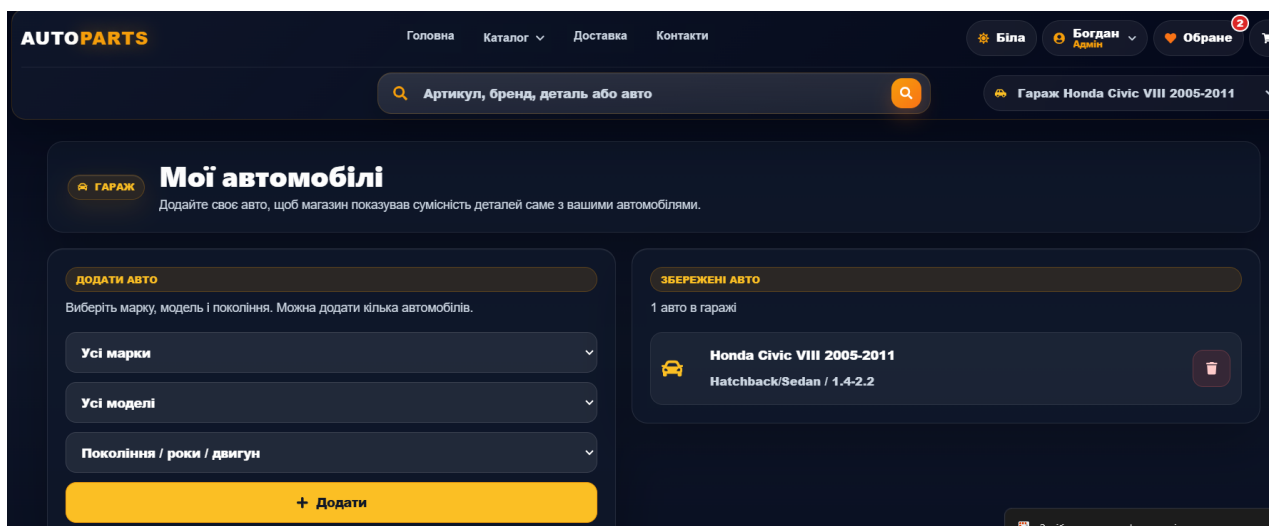


Рисунок 4.5 – Сторінка гаража автомобілів користувача.

4.4.1. Реалізація доставки, оплати та обробки замовлень

Модуль замовлень у четвертому розділі розглядається як одна з частин системи, а не як єдиний центр реалізації. У клієнтській частині сторінка Checkout збирає контактні дані, спосіб доставки, місто, відділення або адресу, спосіб оплати, промокод і товари з кошика. Після підтвердження форма надсилає дані на серверний маршрут /checkout.

На рисунку 4.6 наведено сторінку оформлення замовлення. У ній поєднано контактні дані користувача, вибір способу доставки, пошук міста та відділення Нової пошти, вибір способу оплати, промокод і підсумок замовлення.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		56

Рисунок 4.6 – Сторінка оформлення замовлення з вибором доставки та оплати.

Для доставки реалізовано інтеграцію з API Нової пошти. Клієнтська частина надсилає запити до серверних маршрутів /nova-poshta/cities і /nova-poshta/warehouses, а сервер звертається до зовнішнього API, нормалізує

БР.ІП - 05.00.00.000 ПЗ

відповідь і повертає список міст, відділень або поштоматів [23]. Такий підхід дозволяє не відкривати API-ключ у клієнтському коді.

У системі реалізовано збереження способу та статусу оплати в межах замовлення. Це дозволяє підтримувати різні сценарії оплати, зокрема оплату при отриманні та подальше підключення зовнішнього платіжного сервісу для онлайн-оплати. При цьому магазин не повинен зберігати повні платіжні реквізити користувача, що відповідає загальним вимогам безпеки вебзастосунків [18, 20].

Під час створення замовлення сервер перевіряє товари та залишки в межах транзакції PostgreSQL. Для кожної позиції виконується вибірка з блокуванням, після чого кількість товару зменшується, створюється запис у orders, а склад замовлення записується в order_items. Якщо будь-яка операція завершується помилкою, транзакція скасовується [12, 14]. Лістинг 4.7 демонструє фрагмент використання транзакції та блокування товару під час оформлення замовлення.

Лістинг 4.7 – Використання транзакції під час оформлення замовлення.

```
await client.query('BEGIN');
const productResult = await client.query(
  `SELECT ${productSelect}
  FROM products
  WHERE id = $1
  FOR UPDATE`,
  [item.id]
);
await client.query('COMMIT');
```

Окрім створення замовлення, реалізовано перегляд історії покупок, скасування замовлення користувачем і зміну статусу адміністратором. Під час скасування система повертає кількість товарів на склад, якщо статус

замовлення дозволяє це зробити. Для обробки помилок сервер використовує HTTP-статуси і текстові повідомлення у JSON-форматі [3, 11, 40].

4.4.2. Реалізація додаткових користувацьких можливостей

Окрім основних модулів каталогу, авторизації, гаража та замовлень, у вебзастосунку реалізовано низку додаткових можливостей, які підвищують зручність роботи користувача із сайтом. До них належать список обраних товарів, перемикання теми оформлення, toast-повідомлення, рейтинг і відгуки, профіль користувача, історія замовлень, промокоди та галерея зображень товару.

Список обраних товарів реалізовано на клієнтській стороні за допомогою localStorage. У файлі src/utils/favorites.js зберігаються ідентифікатори обраних товарів і скорочені дані самих товарів. Це дає змогу користувачеві швидко повернутися до потрібних позицій навіть після оновлення сторінки [27]. Такий функціонал не є критичним для оформлення покупки, але покращує повторне використання сайту.

Лістинг 4.8 демонструє логіку додавання товару в обране або видалення його зі списку обраного.

На рисунку 4.7 наведено сторінку обраних товарів. Вона дозволяє користувачеві переглядати збережені позиції, переходити до детальної сторінки товару, додавати товар до кошика або прибирати його зі списку обраного.

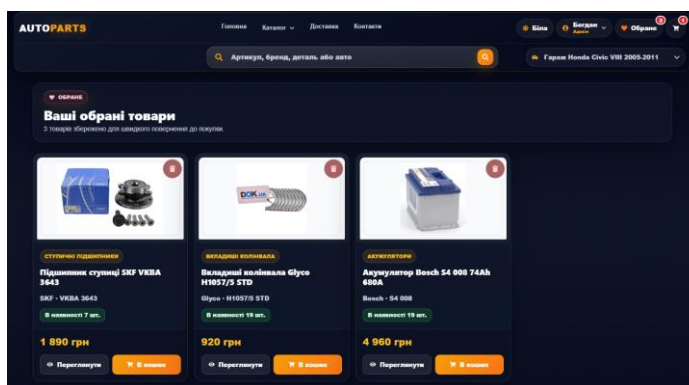


Рисунок 4.7 – Сторінка товарів, збережених користувачем в обране.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		59

Лістинг 4.8 – Перемикання товару в списку обраного.

```
export const toggleFavoriteProduct = (productId) => {
  const productId = getProductId(productId);
  const normalizedId = normalizeId(productId);
  const currentIds = getFavoriteProductIds();
  const isFavorite = currentIds.includes(normalizedId);
  const nextIds = isFavorite ? currentIds.filter((id) => id !== normalizedId) : [...currentIds,
normalizedId];
  return {
    favoriteIds: saveFavoriteProductIds(nextIds),
    isFavorite: !isFavorite,
  };
};
```

У застосунку також реалізовано світлу і темну тему оформлення. Поточне значення теми зберігається в localStorage, а до кореневого елемента документа додається атрибут data-theme. Завдяки цьому стилі можуть змінювати зовнішній вигляд сторінок без перезавантаження застосунку. Такий підхід також використовується для сторінок каталогу, checkout, адміністративної панелі, профілю та обраного.

Для зворотного зв'язку з користувачем використано toast-повідомлення. Вони показують результат дії: успішний вхід, додавання товару в кошик, помилку підключення до сервера, видалення автомобіля з гаража, збереження товару в адмін-панелі або помилку валідації. Повідомлення мають короткий час життя, після чого автоматично зникають.

Сторінка профілю користувача дозволяє переглядати і змінювати особисті дані: ім'я, прізвище, телефон, місто, адресу доставки та відділення. Ці дані використовуються повторно під час оформлення замовлення, щоб користувачеві не потрібно було щоразу вводити однакову інформацію. Також у профілі відображаються короткі показники: кількість товарів у кошику, кількість обраних позицій і посилання на історію замовлень. На рисунку 4.8 наведено сторінку профілю користувача. У ній користувач може переглянут

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		60

свої контактні дані, оновити інформацію для доставки та перейти до пов’язаних розділів сайту.

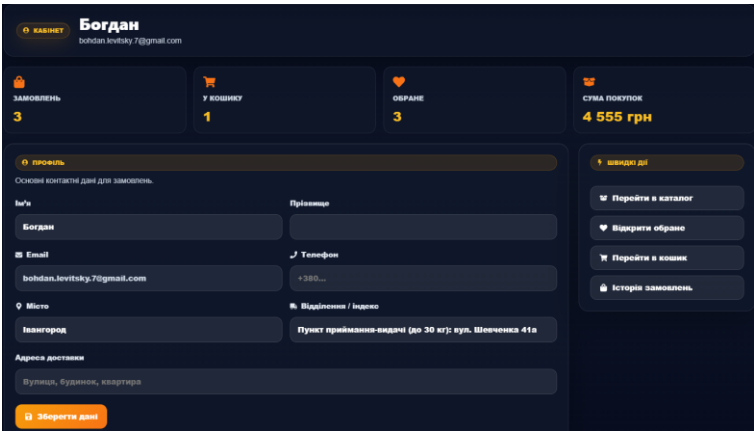


Рисунок 4.8 – Кабінет користувача з контактними даними та статистикою.

Для замовлень реалізовано не лише створення, а й історію покупок. Сторінка замовлення поточного користувача, показує статус, суму, доставку, оплату та перелік товарів. Якщо статус замовлення дозволяє, користувач може скасувати його. На серверній стороні така дія виконується в межах транзакції, оскільки потрібно змінити статус замовлення і повернути товарні залишки на склад [12, 14]. На рисунку 4.9 наведено сторінку історії замовлень. Вона показує список попередніх покупок, поточний статус кожного замовлення, суму, спосіб доставки, спосіб оплати та перелік товарів у замовленні.

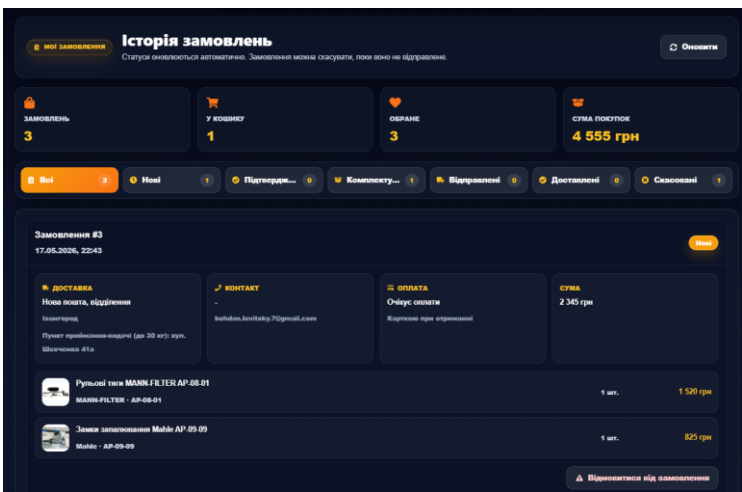


Рисунок 4.9 – Сторінка перегляду попередніх замовлень користувача.

На сторінці товару реалізовано рейтинг і відгуки. Користувач може поставити оцінку товару, а також залишити текстовий відгук. На серверній стороні для цього використовуються таблиці `product_ratings` і `product_reviews`, а в каталозі середні оцінки враховуються під час сортування популярних або рейтингових позицій. Для рейтингу передбачено обмеження значення від 1 до 5, що підтримує коректність даних у PostgreSQL [12, 15].

На рисунку 4.10 наведено блок рейтингу та відгуків на сторінці товару. Цей блок дозволяє користувачеві оцінити товар, переглянути середню оцінку та прочитати відгуки інших покупців.

Рисунок 4.10 – Відображення оцінки товару та користувацьких відгуків.

У кошику та checkout реалізовано промокоди. Користувач може ввести код, після чого система перевіряє мінімальну суму замовлення та обчислює знижку. Активний промокод зберігається тимчасово на клієнтській стороні, а під час створення замовлення сервер отримує вже розраховане значення знижки. Це допомагає продемонструвати типову для інтернет-магазинів бізнес-логіку без ускладнення основної моделі даних.

Для товарів реалізовано галерею зображень. У картці товару можна переглядати основне зображення та додаткові фото, а в адміністративній панелі передбачено можливість редагування зображення товару. Зображення товарів частково підготовлені за допомогою допоміжного сервісу SerpApi, однак у

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						62
Змн.	Арк.	№ докум.	Підпис	Дат		

роботі сайту вони використовуються як звичайні локальні або збережені посилання на медіафайли [25].

4.5. Висновки по розділу

У четвертому розділі описано програмну реалізацію вебзастосунку Auto Parts. Розглянуто структуру проєкту, підготовку клієнтської та серверної частин, використання прм-скриптів, підключення до PostgreSQL і збереження конфігурації через змінні середовища.

Описано реалізацію клієнтської частини: навігацію між сторінками, каталог, картку товару, гараж користувача, кошик, обране, checkout і адміністративну панель. Наведено фрагменти коду, які демонструють вибір сторінки, пошук і перевірку сумісності товару з автомобілем.

На серверній стороні розглянуто авторизацію, хешування паролів, Google-вхід, роботу з товарами, автомобілями, гаражем, доставкою, оплатою та замовленнями. Окремо описано використання транзакцій PostgreSQL для операцій, які змінюють складські залишки та створюють замовлення.

Реалізовані модулі разом забезпечують основну функціональність Auto Parts: перегляд і пошук автозапчастин, перевірку сумісності з автомобілем, роботу з кошиком та обраним, авторизацію користувачів, адміністративне керування каталогом, доставку Новою поштою, підтримку способів оплати та обробку замовлень

Практичним результатом реалізації є вебзастосунок, у якому основні дії користувача та адміністратора пов'язані між собою через спільну базу даних і REST API. Клієнтська частина відповідає за відображення інтерфейсу, зручну взаємодію зі сторінками та передавання запитів, а серверна частина виконує перевірку даних. Такий розподіл дозволяє підтримувати проєкт у зрозумілій структурі, окремо вдосконалювати інтерфейс і серверну логіку, а також використовувати реалізовані модулі як цілісну основу для експлуатації інтернет-магазину автозапчастин.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						63
Змн.	Арк.	№ докум.	Підпис	Дат		

РОЗДІЛ 5. ТЕСТУВАННЯ ТА ВПРОВАДЖЕННЯ ВЕБЗАСТОСУНКУ AUTO PARTS

5.1. Стратегія та середовище тестування

Після реалізації основних модулів вебзастосунку Auto Parts було виконано перевірку працездатності системи. Метою тестування є підтвердження того, що користувач може виконувати основні дії інтернет-магазину автозапчастин: переглядати каталог, шукати товари, перевіряти сумісність із автомобілем, додавати товари до кошика, оформляти замовлення, використовувати обране, керувати профілем і переглядати історію покупок. Для адміністратора перевірялися сценарії керування товарами, користувачами та замовленнями.

Тестування Auto Parts поділено на кілька напрямів: функціональне тестування, інтеграційне тестування, перевірку коректності обробки помилок, базову перевірку безпеки, оцінку якості клієнтської частини та перевірку production-збірки. Такий підхід дає змогу оцінити не лише окремі сторінки інтерфейсу, а й взаємодію клієнтської частини з сервером, базою даних і зовнішніми сервісами.

Для оцінки якості програмного продукту було враховано характеристики, які використовуються в моделях якості програмного забезпечення: функціональна придатність, надійність, зручність використання, супроводжуваність і безпека [32]. Для вебзастосунку Auto Parts ці характеристики мають практичне значення, оскільки система працює з обліковими записами користувачів, персональними даними, замовленнями, зовнішніми API та операціями, пов'язаними з оплатою. Окрему увагу було приділено перевірці авторизації, хешування паролів, обмеженню доступу до адміністративних дій і коректному поверненню HTTP-відповідей у разі помилок [1, 3, 18, 19, 20, 37].

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		64

Тестування виконувалося у локальному середовищі розроблення. Клієнтська частина запускала як React/Vite-застосунок, серверна частина працювала на Node.js та Express, а збереження даних виконувалося у PostgreSQL. Такий підхід дозволяє перевірити взаємодію між основними частинами системи до перенесення проєкту на production-сервер. Таблиця 5.1 – Середовище тестування вебзастосунку Auto Parts.

Таблиця 5.1

Середовище тестування вебзастосунку

Компонент	Використане середовище або інструмент	Призначення
Операційна система	Windows, локальне середовище розробника	Запуск клієнтської та серверної частин
Node.js	v24.14.1	Виконання серверної частини та npm-скриптів
npm	v11.11.0	Керування залежностями та запуск команд
Клієнтська частина	React, Vite	Інтерфейс користувача та production-збірка [4, 6, 36]
Серверна частина	Node.js, Express	REST API, бізнес-логіка та обробка запитів [8, 10, 11]
База даних	PostgreSQL	Збереження користувачів, товарів, гаража й замовлень
Статичний аналіз	ESLint	Перевірка JavaScript-коду на типові помилки [30]
Контроль версій	Git	Збереження історії змін і підготовка до впровадження [31]

Загальний процес тестування наведено на рисунку 5.1. Він починається з визначення цілей перевірки та підготовки тестових даних. Після цього виконуються функціональні й інтеграційні перевірки, статичний аналіз коду та production-збірка. Якщо під час тестування виявляються помилки, вони фіксуються, виправляються, а відповідний сценарій перевіряється повторно.



Рисунок 5.1 – Процес тестування вебзастосунку Auto Parts.

5.2. Функціональне тестування користувацьких сценаріїв

Функціональне тестування спрямоване на перевірку того, чи відповідає реалізована система поставленим вимогам. Для Auto Parts основними є сценарії звичайного покупця та адміністратора. Користувач взаємодіє з головною сторінкою, каталогом, карткою товару, гаражем, кошиком, обраним, профілем і checkout. Адміністратор виконує керування каталогом товарів, переглядає замовлення та змінює їхні статуси.

Під час тестування застосовано підхід “чорної скриньки”, коли перевіряється результат дій користувача без детального аналізу внутрішньої реалізації кожної функції. Такий підхід є доцільним для перевірки інтерфейсу, форм введення, переходів між сторінками та реакції системи на правильні або неправильні дані. Для сценаріїв, пов’язаних із базою даних і зовнішніми API, додатково враховувалася інтеграційна поведінка, оскільки результат залежить не лише від клієнтської частини, а й від серверної логіки.

У таблиці 5.2 наведено основні функціональні сценарії тестування. Для кожного сценарію визначено дії користувача, очікуваний результат і статус перевірки.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						66
Змн.	Арк.	№ докум.	Підпис	Дат		

Таблиця 5.2

Функціональні сценарії тестування

№	Сценарій	Дії під час тестування	Очікуваний результат	Статус
T-01	Завантаження головної сторінки	Відкрити сайт Auto Parts	Відображається головна сторінка, навігація, блоки товарів і перехід до каталогу	Успішно
T-02	Пошук товару в каталозі	Ввести назву або частину назви автозапчастини	Список товарів фільтрується відповідно до пошукового запиту	Успішно
T-03	Фільтрація за категоріями	Обрати категорію або підкатегорію товарів	У каталозі залишаються товари вибраної групи	Успішно
T-04	Сортування товарів	Змінити тип сортування за ціною, популярністю або рейтингом	Порядок товарів змінюється відповідно до вибраного критерію	Успішно
T-05	Перегляд картки товару	Відкрити детальну сторінку товару	Відображаються характеристики, ціна, фото, сумісність, рейтинг і кнопки дій	Успішно
T-06	Перевірка сумісності	Додати автомобіль у гараж і відкрити сумісний або несумісний товар	Система показує повідомлення про сумісність товару з автомобілем	Успішно
T-07	Додавання до кошика	Натиснути кнопку додавання товару до кошика	Товар з'являється в кошику, кількість товарів у шапці сайту оновлюється	Успішно
T-08	Зміна кількості в кошику	Змінити кількість товару або видалити позицію	Загальна сума перераховується, кошик зберігає актуальний стан	Успішно
T-09	Обране	Додати товар в обране та перейти на сторінку обраного	Товар зберігається у списку обраного й доступний після оновлення сторінки	Успішно
T-10	Реєстрація та вхід	Заповнити форму реєстрації або авторизації	Користувач входить у систему, пароль не зберігається у відкритому вигляді [18, 19, 33]	Успішно
T-11	Google-вхід	Виконати авторизацію через Google Identity	Сервер перевіряє ID-токен і створює або знаходить користувача [21, 22]	Успішно
T-12	Редагування профілю	Змінити телефон, місто або адресу доставки	Дані профілю оновлюються і використовуються під час checkout	Успішно

Арк.

БР.ІП - 05.00.00.000 ПЗ

67

Змн. Арк. № докум. Підпис Дат

Продовження таблиці 5.2

T-13	Гараж користувача	Додати або видалити автомобіль	Автомобіль зберігається в гаражі або видаляється зі списку	Успішно
T-14	Оформлення замовлення	Заповнити дані доставки, обрати оплату і підтвердити замовлення	Створюється замовлення, кошик очищується, складські залишки оновлюються	Успішно
T-15	Історія замовлень	Перейти на сторінку попередніх замовлень	Користувач бачить перелік власних замовлень, статуси, суму й товари	Успішно
T-16	Скасування замовлення	Скасувати замовлення зі статусом, який дозволяє скасування	Статус змінюється, кількість товарів повертається на склад	Успішно
T-17	Адміністративне керування товарами	Додати, змінити або видалити товар в адмін-панелі	Дані каталогу оновлюються після дії адміністратора	Успішно
T-18	Відгуки та рейтинг	Додати оцінку або текстовий відгук на сторінці товару	Оцінка враховується у середньому рейтингу, відгук відображається у списку	Частково успішно

За результатами функціонального тестування 17 із 18 підготовлених сценаріїв виконалися відповідно до очікуваної поведінки. Сценарій перевірки відгуків і рейтингу позначено як частково успішний, оскільки основне додавання оцінки працює, але повідомлення для користувача під час введення некоректного тексту відгуку потребує уточнення. Частина сценаріїв перевірялася повторно після зміни даних у кошику, профілі або замовленнях, оскільки саме ці модулі найчастіше змінюють стан системи.

На рисунку 5.2 наведено сторінку каталогу після виконання пошуку або фільтрації. Цей скріншот підтверджує роботу одного з найважливіших сценаріїв користувача: швидкий пошук потрібної автозапчастини за назвою, категорією або сумісністю з автомобілем.

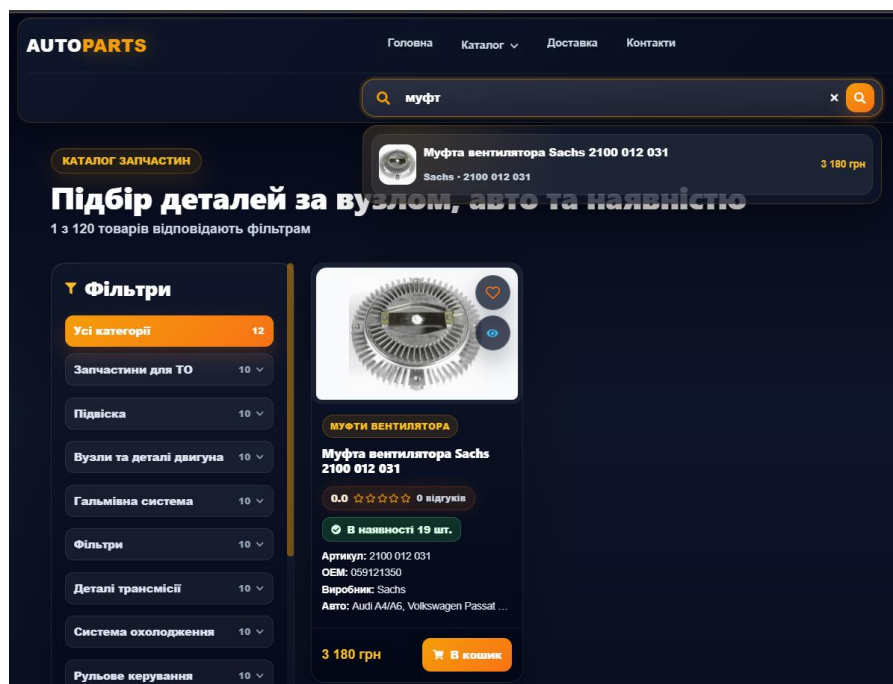


Рисунок 5.2 – Перевірка пошуку та фільтрації товарів у каталозі Auto Parts.

На рисунку 5.3 наведено адміністративну панель під час перевірки керування товарами або замовленнями. Такий скріншот демонструє, що адміністратор може працювати з даними каталогу та контролювати виконання замовлень.

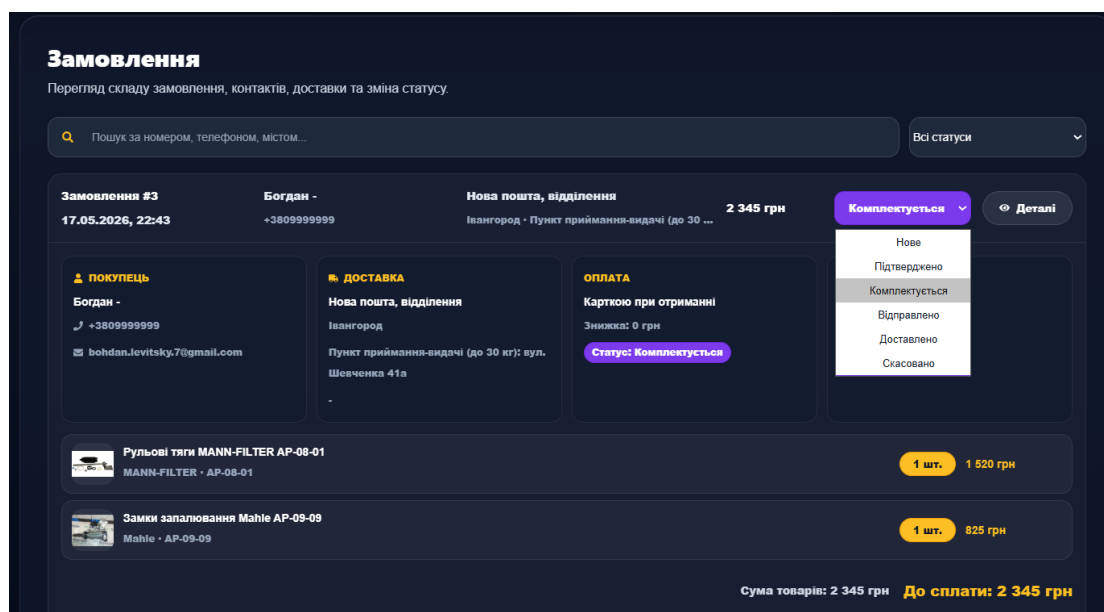


Рисунок 5.3 – Перевірка адміністративних функцій вебзастосунку Auto Parts.

5.3. Інтеграційне тестування та перевірка обробки помилок

Інтеграційне тестування було спрямоване на перевірку взаємодії між клієнтською частиною, Express API, базою даних PostgreSQL та зовнішніми сервісами. Для вебзастосунку Auto Parts це особливо важливо, оскільки більшість користувацьких дій не обмежується лише інтерфейсом. Наприклад, додавання товару до кошика виконується на клієнтській стороні, але оформлення замовлення вже потребує серверної перевірки, створення записів у базі даних і зміни складських залишків. Подібно до цього, вибір міста та відділення доставки залежить від API Нової пошти [23].

Основою взаємодії між клієнтом і сервером є HTTP-запити. Для коректної роботи REST API важливо не лише повертати дані у JSON-форматі, а й використовувати відповідні HTTP-статуси: 200 для успішного отримання даних, 201 для створення ресурсу, 400 для помилок вхідних даних, 401 або 403 для обмеження доступу, 404 для відсутнього ресурсу та 500 для неочікуваних серверних помилок [1, 3, 37]. Обробка помилок у Express повинна бути зрозумілою для клієнтської частини, оскільки саме на основі відповіді сервера інтерфейс показує повідомлення користувачу [11].

У таблиці 5.3 наведено інтеграційні сценарії, за допомогою яких перевірялася взаємодія між клієнтською частиною, сервером, базою даних і зовнішніми сервісами.

Таблиця 5.3

Перевірка інтеграційних сценаріїв

№	Компоненти інтеграції	Перевірена дія	Очікуваний результат	Статус
I-01	React-клієнт – Express API	Завантаження списку товарів через <code>/products</code>	Клієнт отримує масив товарів і відображає каталог	Успішно
I-02	React-клієнт – Express API	Пошук через <code>/products/search</code>	Сервер повертає товари, що відповідають запиту	Успішно
I-03	Express API – PostgreSQL	Отримання товару за ідентифікатором	Сервер повертає товар або <code>404</code> , якщо його не існує	Успішно

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дат		

Продовження таблиці 5.3

I-04	Express API – PostgreSQL	Створення замовлення через <code>/checkout</code>	Замовлення та його позиції зберігаються в базі даних	Успішно
I-05	Express API – PostgreSQL	Скасування замовлення	Статус змінюється, товарні залишки повертаються на склад	Успішно
I-06	Express API – Google Identity	Авторизація через Google ID token	Сервер перевіряє токен і повертає дані користувача	Успішно
I-07	Express API – Нова пошта	Отримання міст і відділень	Користувач може обрати місто та відділення доставки	Успішно
I-08	Express API – PostgreSQL	Збереження способу оплати	Система зберігає спосіб і статус оплати в даних замовлення	Успішно

Під час перевірки інтеграційних сценаріїв окремо розглядалися помилкові ситуації. До них належать спроба входу з неправильним паролем, реєстрація з некоректною електронною адресою, створення замовлення з порожнім кошиком, звернення до неіснуючого товару, спроба редагування товару без адміністративних прав і помилка підключення до зовнішнього сервісу. У таких випадках система має повертати зрозуміле повідомлення.

Окрема увага приділялася операціям, які змінюють кілька пов'язаних таблиць. Наприклад, під час створення замовлення потрібно не тільки зберегти основну інформацію про покупку, а й додати позиції замовлення, зменшити залишки товарів і очистити кошик користувача. Якщо одна з цих операцій не виконується, замовлення не повинно залишатися у частково створеному стані. Саме тому для таких дій використовуються транзакції PostgreSQL, описані в попередніх розділах роботи.

5.4. Перевірка якості, безпеки, production-збірки та впровадження системи

Перевірка якості коду виконувалася за допомогою ESLint. Статичний аналіз дозволяє знайти частину помилок ще до запуску застосунку: неправильне

використання змінних, порушення правил React Hooks, зайві або некоректні імпорти, потенційні проблеми стилю коду [30]. Для бакалаврського проєкту така перевірка є важливою, оскільки вона показує, що код можна підтримувати й розвивати без значного ризику випадкових помилок. Лістинг 5.1 демонструє результат запуску статичного аналізу клієнтської частини.

Лістинг 5.1 – Результат виконання статичного аналізу коду.

```
> diplom-site@0.0.0 lint
> eslint .
```

Відсутність повідомлень про помилки після виконання команди `npm run lint` означає, що перевірка завершилася успішно. Це не гарантує відсутності всіх можливих дефектів, але підтверджує, що код проходить налаштовані правила статичного аналізу.

Окремо було виконано `production`-збірку клієнтської частини. Ця перевірка потрібна для того, щоб переконатися, що застосунок може бути зібраний у статичні файли для подальшого розгортання. Vite під час збирання трансформує модулі, формує оптимізовані CSS і JavaScript-файли та показує їхній розмір після `gzip`-стиснення [36]. Лістинг 5.2 демонструє результат виконання `production`-збірки.

Лістинг 5.2 – Результат виконання `production`-збірки клієнтської частини.

```
> diplom-site@0.0.0 build
> vite build
vite v8.0.10 building client environment for production...
transforming...
✓ 65 modules transformed.
rendering chunks...
computing gzip size...
dist/index.html          0.46 kB | gzip: 0.29 kB
dist/assets/index-CeFL8G5J.css 243.41 kB | gzip: 37.49 kB
dist/assets/index-CS5QX6A8.js 495.98 kB | gzip: 145.48 kB
```

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дат		

✓ built in 558ms

Результат збирання показує, що клієнтська частина успішно компілюється і може бути розміщена як набір статичних файлів. Основний JavaScript-файл після gzip-стиснення має розмір 145,48 kB, а CSS-файл – 37,49 kB. Для подальшого вдосконалення продуктивності можна використовувати Lighthouse та Core Web Vitals, які дозволяють оцінити швидкість завантаження, інтерактивність і стабільність відображення сторінки [38].

Перевірка безпеки виконувалася на рівні основних ризиків, характерних для вебзастосунків. Зокрема, було перевірено, що паролі користувачів не зберігаються у відкритому вигляді, а перед записом у базу даних хешуються за допомогою bcrypt. Під час входу користувача сервер порівнює введений пароль із хешем, а не з відкритим текстом. Такий підхід відповідає рекомендаціям щодо зберігання паролів і автентифікації [18, 19, 33]. Також перевірялася логіка доступу до адміністративної панелі. Адміністративні дії мають бути доступні лише користувачам із відповідною роллю, оскільки вони дозволяють змінювати товари, переглядати користувачів і керувати замовленнями. Спроба виконати такі дії без належних прав повинна завершуватися помилкою доступу. Це відповідає загальним рекомендаціям OWASP щодо контролю доступу та зменшення ризиків несанкціонованих дій [19, 20].

Для форм введення перевірялися типові некоректні дані: порожні обов'язкові поля, неправильний формат електронної адреси, відсутність міста доставки, некоректна кількість товару, спроба оформити замовлення з порожнім кошиком. У таких випадках система не повинна створювати замовлення або змінювати дані, а користувач має отримати зрозуміле повідомлення. Для зручності сприйняття у клієнтській частині використовуються toast-повідомлення.

Інтерфейс також перевірявся з погляду зручності використання. Основні сторінки мають зрозумілу навігацію, кнопки дій, повідомлення про результат операції та адаптацію до різних розмірів екрана. Для повнішої оцінки доступності в майбутньому можна додатково перевірити контрастність, фокус

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		73

клавіатури, підписи до форм і відповідність рекомендаціям WCAG [29]. У таблиці 5.4 узагальнено результати перевірки якості, безпеки та технічної готовності веб-застосунку.

Таблиця 5.4

Узагальнені результати перевірки якості

Напрямок перевірки	Кількість перевірок або показник	Результат
Функціональні сценарії	18 сценаріїв	17 виконано успішно, 1 частково успішно
Інтеграційні сценарії	8 сценаріїв	8 виконано успішно
Статичний аналіз коду	<code>npm run lint</code>	Помилки не виявлено
Production-збірка	<code>npm run build</code>	Збірку створено успішно
Кількість трансформованих модулів	56 модулів	Збірка завершена
Перевірка основних HTTP-відповідей	200, 201, 400, 401/403, 404, 500	Логіка обробки помилок передбачена
Базова перевірка безпеки	Паролі, ролі, валідація форм	Основні ризики враховано

Впровадження вебзастосунку Auto Parts передбачає підготовку клієнтської частини, серверної частини, бази даних і змінних середовища. Перед розгортанням потрібно переконатися, що production-збірка клієнтської частини створюється без помилок, сервер запускається з правильними параметрами підключення до бази даних, а ключі зовнішніх сервісів не зберігаються безпосередньо у вихідному коді. Для цього використовуються змінні середовища та файл конфігурації `.env`, який не повинен потрапляти до відкритого репозиторію.

Процес впровадження можна поділити на кілька послідовних етапів. Спочатку готується серверне середовище з Node.js, npm і PostgreSQL. Далі створюється база даних, виконуються необхідні SQL-скрипти або початкове наповнення каталогу товарів. Після цього налаштовуються параметри підключення до Google Identity та Нової пошти. Клієнтська частина збирається командою `npm run build`, а серверна частина запускається через `npm start` або

інший процес-менеджер, який забезпечує постійну роботу сервера [31, 36]. У таблиці 5.5 наведено основні етапи впровадження вебзастосунку Auto Parts в експлуатацію.

Таблиця 5.5

Етапи впровадження вебзастосунку

№	Етап	Зміст робіт	Очікуваний результат
1	Підготовка середовища	Встановлення Node.js, npm, PostgreSQL	Середовище готове до запуску проєкту
2	Налаштування бази даних	Створення БД, таблиць і початкових даних	Сервер має доступ до актуальних даних
3	Налаштування <code>.env</code>	Додавання параметрів БД і ключів зовнішніх API	Конфігурація відокремлена від коду
4	Збірка клієнта	Виконання <code>npm run build</code>	Створено каталог <code>dist</code> зі статичними файлами
5	Запуск сервера	Виконання <code>npm start</code> у серверній частині	REST API доступний для клієнта
6	Перевірка після розгортання	Авторизація, каталог, checkout, адмін-панель	Основні сценарії працюють у робочому середовищі
7	Супровід	Резервне копіювання БД, оновлення залежностей, контроль логів	Система підтримується після запуску

Після впровадження важливо організувати супровід системи. Для інтернет-магазину автозапчастин це означає регулярне оновлення каталогу, перевірку коректності залишків, контроль замовлень, резервне копіювання бази даних і своєчасне оновлення залежностей. У разі розширення системи можна додати автоматизовані unit- та end-to-end-тести, щоб перевіряти критичні сценарії після кожної зміни коду.

5.5. Висновки по розділу

У п'ятому розділі проведено оцінку готовності вебзастосунку Auto Parts до використання. Було визначено стратегію тестування, описано середовище перевірки, сформовано функціональні та інтеграційні сценарії, наведено результати статичного аналізу коду. Отримані результати показують, що

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		75

реалізовані модулі працюють узгоджено: клієнтська частина взаємодіє з REST API, сервер обробляє запити і працює з PostgreSQL, а зовнішні сервіси використовуються для авторизації, доставки, оплати.

Проведена перевірка підтвердила працездатність основних сценаріїв користувача та адміністратора. Система дозволяє переглядати і шукати автозапчастини, перевіряти сумісність із автомобілем, працювати з кошиком і обраним, оформлювати замовлення, переглядати історію. Один із перевірених сценаріїв потребує незначного уточнення повідомлення про некоректні дані, однак це не впливає на роботу основних модулів сайту.

Разом з тим тестування показує напрями подальшого вдосконалення. До них належать розширення автоматизованих тестів, підключення регулярної перевірки продуктивності через Lighthouse, поглиблена перевірка доступності інтерфейсу за WCAG, налаштування моніторингу серверних помилок і створення резервного копіювання бази даних. Виконання цих заходів дозволить підвищити надійність під час реальної експлуатації та підготувати систему до збільшення кількості користувачів і замовлень [29, 31, 38].

					БР.ІІІ - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		76

ВИСНОВКИ

У бакалаврській роботі було розроблено вебзастосунок Auto Parts, призначений для продажу автозапчастин через інтернет. Розроблена система поєднує каталог товарів, пошук, перевірку сумісності автозапчастин з автомобілем користувача, кошик, оформлення замовлення, доставку, профіль користувача, історію покупок та адміністративне керування даними.

У процесі виконання роботи було проаналізовано предметну область онлайн-продажу автозапчастин. Встановлено, що для такого типу систем важливими є не лише наявність каталогу й можливість оформлення замовлення, а й точний підбір деталей під конкретний автомобіль, зручна фільтрація, зрозуміла картка товару, підтримка доставки та коректна робота із залишками товарів на складі.

На основі аналізу предметної області було визначено основні ролі користувачів системи: гість, зареєстрований користувач та адміністратор. Для користувача передбачено перегляд каталогу, пошук і фільтрацію товарів, роботу з гаражем автомобілів, кошиком, обраними товарами, оформленням замовлення, профілем та історією покупок. Для адміністратора реалізовано можливість керування товарами, користувачами, автомобільними відповідностями та замовленнями.

У роботі сформовано функціональні та нефункціональні вимоги до вебзастосунку. До основних функціональних вимог належать авторизація користувачів, каталог автозапчастин, пошук, фільтрація, сортування, картка товару, перевірка сумісності, кошик, checkout, доставка, підтримка способів оплати, рейтинг, відгуки та адміністративна панель. До нефункціональних вимог віднесено зручність інтерфейсу, безпеку, продуктивність, підтримуваність, розширюваність і коректну обробку помилок.

Для реалізації системи було обрано клієнт-серверну архітектуру. Клієнтську частину створено з використанням React та Vite, що дозволило

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						77
Змн.	Арк.	№ докум.	Підпис	Дат		

побудувати інтерфейс із повторно використовуваних компонентів. Серверну частину реалізовано за допомогою Node.js та Express. Для збереження даних використано PostgreSQL, де зберігаються користувачі, товари, автомобілі, замовлення, відгуки, рейтинги та інші дані, необхідні для роботи системи.

У процесі проектування було визначено структуру основних модулів вебзастосунку, логіку взаємодії клієнтської частини із сервером, структуру бази даних і сценарій оформлення замовлення. Особливу увагу приділено цілісності даних під час створення та скасування замовлень, оскільки ці операції пов'язані зі зміною декількох сутностей: замовлення, позицій замовлення та складських залишків.

У програмній реалізації створено головну сторінку, каталог, сторінку товару, кошик, checkout, сторінку обраних товарів, гараж користувача, профіль, історію замовлень, сторінки авторизації та адміністративну панель. Реалізовано авторизацію через email і пароль, а також вхід через Google. Паролі користувачів зберігаються у вигляді хешів, що підвищує рівень захисту облікових записів. Для оформлення доставки використано інтеграцію з API Нової пошти.

Реалізовано додаткові функції, спрямовані на підвищення зручності використання системи, зокрема гараж автомобілів, список обраних товарів, рейтинг і відгуки, адаптивний інтерфейс та підтримку різних тем оформлення. Такі можливості роблять вебзастосунок більш наближеним до реального інтернет-магазину, а не лише до демонстраційного каталогу товарів.

Для перевірки працездатності системи виконано функціональне та інтеграційне тестування. Було перевірено основні сценарії користувача й адміністратора: завантаження сторінок, пошук, фільтрацію, перегляд товару, роботу гаража, додавання товарів до кошика, оформлення замовлення, роботу обраного, авторизацію, профіль, історію замовлень, скасування замовлення та керування товарами в адміністративній панелі. Із 18 функціональних сценаріїв 17 виконано успішно, а один сценарій позначено як частково успішний, тому рівень успішності становить 94,4%.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
						78
Змн.	Арк.	№ докум.	Підпис	Дат		

Також було виконано перевірку якості клієнтської частини. Статичний аналіз коду командою `npm run lint` не виявив помилок, а `production`-збірка командою `npm run build` завершилася успішно. Це підтверджує, що клієнтська частина може бути підготовлена до розгортання та подальшого використання.

Практичним результатом бакалаврської роботи є робочий вебзастосунок `Auto Parts`, який може використовуватися як основа для подальшого розвитку інтернет-магазину автозапчастин. У майбутньому систему можна розширити автоматизованими тестами, покращеним механізмом рекомендацій товарів, аналітикою продажів, резервним копіюванням бази даних і повноцінним моніторингом серверних помилок.

Отже, поставлену мету бакалаврської роботи досягнуто. Було проаналізовано предметну область, сформовано вимоги, спроектовано архітектуру та базу даних, реалізовано клієнтську і серверну частини вебзастосунку, проведено тестування та визначено напрями подальшого розвитку. Розроблений вебзастосунок `Auto Parts` демонструє практичне застосування сучасних вебтехнологій для створення функціонального інтернет-магазину автозапчастин.

					БР.ІП - 05.00.00.000 ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дат		79

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

- [1] RFC 9110: HTTP Semantics [Електронний ресурс]. – Специфікація протоколу HTTP. URL: <https://www.rfc-editor.org/rfc/rfc9110>
- [2] Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures [Електронний ресурс]. – Дисертаційна робота про архітектурні стилі програмних систем. URL: <https://ics.uci.edu/~fielding/pubs/dissertation/top.htm>
- [3] RFC 9457: Problem Details for HTTP APIs [Електронний ресурс]. – Специфікація формату помилок HTTP API. URL: <https://www.rfc-editor.org/rfc/rfc9457>
- [4] React Documentation [Електронний ресурс]. – Офіційна документація бібліотеки React. URL: <https://react.dev/reference/react>
- [5] React Hooks Reference [Електронний ресурс]. – Довідкова документація щодо використання React Hooks. URL: <https://react.dev/reference/react/hooks>
- [6] Vite Documentation [Електронний ресурс]. – Офіційний посібник із використання Vite. URL: <https://vite.dev/guide/>
- [7] Why Vite. Vite Documentation [Електронний ресурс]. – Опис призначення та переваг інструмента Vite. URL: <https://vite.dev/guide/why.html>
- [8] Node.js Documentation [Електронний ресурс]. – Офіційна API-документація платформи Node.js. URL: <https://nodejs.org/docs/latest/api/>
- [9] About Node.js [Електронний ресурс]. – Офіційний опис платформи Node.js. URL: <https://nodejs.org/en/about>
- [10] Express.js Routing [Електронний ресурс]. – Офіційний посібник із маршрутизації в Express.js. URL: <https://expressjs.com/en/guide/routing.html>
- [11] Express.js Error Handling [Електронний ресурс]. – Офіційний посібник з обробки помилок в Express.js. URL: <https://expressjs.com/en/guide/error-handling.html>

[12] PostgreSQL Documentation [Електронний ресурс]. – Офіційна документація системи керування базами даних PostgreSQL. URL: <https://www.postgresql.org/docs/current/>

[13] PostgreSQL Tutorial [Електронний ресурс]. – Навчальний розділ документації PostgreSQL. URL: <https://www.postgresql.org/docs/current/tutorial.html>

[14] PostgreSQL Transaction Isolation [Електронний ресурс]. – Документація щодо ізоляції транзакцій у PostgreSQL. URL: <https://www.postgresql.org/docs/current/transaction-iso.html>

[15] PostgreSQL Constraints [Електронний ресурс]. – Документація щодо обмежень цілісності даних у PostgreSQL. URL: <https://www.postgresql.org/docs/current/ddl-constraints.html>

[16] node-postgres Documentation: Pooling [Електронний ресурс]. – Документація щодо пулу підключень до PostgreSQL. URL: <https://node-postgres.com/features/pooling>

[17] node-postgres API: Pool [Електронний ресурс]. – API-довідка бібліотеки node-postgres. URL: <https://node-postgres.com/apis/pool>

[18] OWASP Password Storage Cheat Sheet [Електронний ресурс]. – Рекомендації щодо безпечного зберігання паролів. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html

[19] OWASP Authentication Cheat Sheet [Електронний ресурс]. – Рекомендації

Щодо автентифікації користувачів. URL:

https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html

[20] OWASP Top 10:2021 [Електронний ресурс]. – Перелік основних ризиків безпеки вебзастосунків. URL: <https://owasp.org/Top10/>

[21] Google Identity: Verify the Google ID token [Електронний ресурс]. – Офіційний посібник із перевірки Google ID token. URL: <https://developers.google.com/identity/gsi/web/guides/verify-google-id-token>

[22] Google Identity Services for Web [Електронний ресурс]. – Офіційна документація Google Identity Services для вебзастосунків. URL: <https://developers.google.com/identity/gsi/web>

[23] Nova Post API Methods [Електронний ресурс]. – Офіційна документація методів API Нової пошти. URL: <https://api-portal.novapost.com/en/api-methods/>

[24] SerpApi Documentation [Електронний ресурс]. – Документація сервісу пошуку та отримання зображень. URL: <https://serpapi.com/>

[25] MDN Web Docs: Fetch API [Електронний ресурс]. – Довідкова документація браузерного Fetch API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

[26] MDN Web Docs: Web Storage API [Електронний ресурс]. – Довідкова документація Web Storage API. URL: https://developer.mozilla.org/en-US/docs/Web/API/Web_Storage_API

[27] MDN Web Docs: CORS [Електронний ресурс]. – Довідкова документація щодо Cross-Origin Resource Sharing. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS>

[28] W3C Web Content Accessibility Guidelines 2.2 [Електронний ресурс]. – Рекомендації з доступності вебконтенту. URL: <https://www.w3.org/TR/WCAG22/>

[29] ESLint Documentation [Електронний ресурс]. – Офіційна документація інструмента статичного аналізу коду. URL: <https://eslint.org/docs/latest/>

[30] Git Documentation [Електронний ресурс]. – Офіційна документація системи контролю версій Git. URL: <https://git-scm.com/doc>

[31] ISO/IEC 25010: Systems and software Quality Requirements and Evaluation [Електронний ресурс]. – Опис моделі якості програмного

забезпечення. URL: <https://iso25000.com/index.php/en/iso-25000-standards/iso-25010>

[32] bcrypt. npm package documentation [Електронний ресурс]. – Документація бібліотеки для хешування паролів. URL: <https://www.npmjs.com/package/bcrypt>

[33] dotenv. npm package documentation [Електронний ресурс]. – Документація бібліотеки для роботи зі змінними середовища. URL: <https://www.npmjs.com/package/dotenv>

[34] npm Documentation [Електронний ресурс]. – Офіційна документація менеджера пакетів npm. URL: <https://docs.npmjs.com/>

[35] Vite Documentation: Building for Production [Електронний ресурс]. – Посібник зі створення production-збірки. URL: <https://vite.dev/guide/build>

[36] MDN Web Docs: HTTP response status codes [Електронний ресурс]. – Довідкова документація щодо HTTP-статусів відповіді. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Reference/Status>

[37] web.dev: Core Web Vitals [Електронний ресурс]. – Опис метрик продуктивності вебсторінок. URL: <https://web.dev/articles/vitals>

[38] Lighthouse Documentation [Електронний ресурс]. – Офіційна документація інструмента перевірки якості вебсторінок. URL: <https://developer.chrome.com/docs/lighthouse/overview>

БІБЛІОГРАФІЧНА ДОВІДКА

Тема бакалаврської роботи: “Проектування та розробка
веб-застосунку для автоматизації продажу автомобільних запчастин”

Обсяг пояснювальної записки: 73 аркуші

Дата закінчення роботи червня 2026р.

Підпис _____