

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

«Вебзастосунок продажу вживаних автомобілів»

Виконав студент групи ПЗп-22-1
ПОЛЩУК Ілля Андрійович

Керівник роботи:
ВАКАЛЮК Тетяна Анатоліївна

Рецензент:
ЧИЖМОТРЯ Олена Геннадіївна

Житомир – 2026

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна Вакалюк

«16» липня 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ПОЛІЩУК Ілля Андрійович**

Керівник роботи: **ВАКАЛЮК Тетяна Анатоліївна**

Тема роботи: «Вебзастосунок продажу вживаних автомобілів»,

затверджена Наказом закладу вищої освіти від **«16» липня 2026 р., №396/с**

Вихідні дані для роботи: Вебзастосунок продажу вживаних автомобілів

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Вакалюк Т.А.	16.07.2026р.	16.07.2026р.
2	Вакалюк Т.А.	16.07.2026р.	16.07.2026р.
3	Вакалюк Т.А.	16.07.2026р.	16.07.2026р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з реалізованого вебзастосунку продажу вживаних автомобілів та пояснювальної записки. Пояснювальна записка містить 92 сторінок, 33 ілюстрацій, 27 таблиць, 81 сторінку додатків та 40 використаних джерел.

Метою випускної кваліфікаційної роботи є проєктування та реалізація вебзастосунку для продажу вживаних автомобілів, який забезпечує перегляд оголошень, розширений пошук і сортування транспортних засобів, перегляд історії зміни ціни, розміщення власних оголошень, обмін повідомленнями між користувачами, створення заявок на придбання автомобілів, відновлення пароля, керування профілем та адміністрування вмісту системи.

У процесі виконання роботи було проаналізовано предметну область продажу вживаних автомобілів, розглянуто існуючі вебплатформи відповідного призначення, визначено функціональні та нефункціональні вимоги до системи, спроектовано структуру бази даних, розроблено алгоритми роботи основних модулів і реалізовано програмний комплекс.

Архітектура розробленої системи побудована за клієнт-серверним принципом. Клієнтська частина реалізована засобами HTML, CSS та JavaScript і забезпечує взаємодію користувача з вебзастосунком через браузер. Серверна частина реалізована на основі Node.js та відповідає за обробку запитів, автентифікацію користувачів, перевірку прав доступу, роботу з автомобілями, заявками, повідомленнями, відгуками, обраними оголошеннями, історією переглядів, історією зміни ціни та токенами відновлення пароля. Для зберігання даних використано систему керування базами даних MySQL.

Розроблений вебзастосунок підтримує роботу різних категорій користувачів: гостей, зареєстрованих користувачів, власників оголошень та адміністратора. Гість може переглядати оголошення та використовувати

					ДУ «Житомирська політехніка».26.121.18.000 – ПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата				
Розроб.		Поліщук І. А.			Вебзастосунок продажу вживаних автомобілів	Літ.	Арк.	Аркушів
Перевір.							3	182
Керівник		Вакалюк Т. А.				ФІКТ, Гр. ІПЗ-22-1		
Н. контр.		Тичина О. П.						
Зав. каф.		Вакалюк Т. А.						

розширену фільтрацію. Авторизований користувач отримує можливість створювати, редагувати й видаляти власні оголошення, керувати фотографіями, додавати автомобілі до обраного, залишати відгуки, надсилати заявки на придбання автомобіля, вести листування з продавцем, редагувати профіль і відновлювати пароль. Адміністратор має доступ до керування користувачами й оголошеннями, що забезпечує контроль інформаційного наповнення платформи.

Практичне значення отриманих результатів полягає у можливості використання розробленого вебзастосунку як прототипу онлайн-платформи для організації процесу купівлі та продажу вживаних автомобілів. Система може бути розширена шляхом додавання нових функцій, зокрема перевірки транспортних засобів, розширеної модерації, рекомендаційного пошуку або інтеграції з платіжними сервісами.

КЛЮЧОВІ СЛОВА: ВЕБЗАСТОСУНОК, АВТОМОБІЛЬ, ОГОЛОШЕННЯ, ФІЛЬТРАЦІЯ, JWT, MYSQL, КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		4

ABSTRACT

The bachelor's final qualification work consists of an implemented web application for the sale of used cars and an explanatory note. The explanatory note contains 92 pages, 33 illustrations, 27 tables, 81 pages of appendices and 40 sources used.

The purpose of the final qualification work is to design and implement a web application for the sale of used cars, which provides viewing of ads, advanced search and sorting of vehicles, viewing the history of price changes, placing one's own ads, exchanging messages between users, creating applications for the purchase of cars, recovering the password, managing the profile and administering the system content.

In the process of performing the work, the subject area of the sale of used cars was analyzed, existing web platforms of the corresponding purpose were considered, functional and non-functional requirements for the system were determined, the database structure was designed, the algorithms of the main modules were developed and the software package was implemented.

The architecture of the developed system is built on the client-server principle. The client part is implemented using HTML, CSS and JavaScript and provides user interaction with the web application via a browser. The server part is implemented based on Node.js and is responsible for processing requests, user authentication, checking access rights, working with cars, applications, messages, reviews, selected ads, viewing history, price change history and password recovery tokens. The MySQL database management system is used to store data.

The developed web application supports the work of different categories of users: guests, registered users, ad owners and the administrator. A guest can view ads and use advanced filtering. An authorized user gets the opportunity to create, edit and delete their own ads, manage photos, add cars to favorites, leave reviews, send applications for purchasing a car, correspond with the seller, edit the profile and recover the password. The administrator has access to user and ad management, which provides control over the information content of the platform.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				5
Змн.	Арк.	№ докум.	Підпис	Дата		

The practical significance of the results obtained lies in the possibility of using the developed web application as a prototype of an online platform for organizing the process of buying and selling used cars. The system can be expanded by adding new functions, in particular vehicle verification, advanced moderation, recommendation search or integration with payment services.

KEYWORDS: WEB APPLICATION, CAR, ADVERTISEMENT, FILTERING, JWT, MYSQL, CLIENT-SERVER ARCHITECTURE.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				6
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ВСТУП.....	10
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБЗАСТОСУНКУ.....	12
1.1. Постановка задачі	12
1.2. Аналіз аналогів програмного продукту	16
1.3. Вибір архітектури веб-застосунку	21
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	26
Висновки до першого розділу	31
РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ ПРОДАЖУ ВЖИВАНИХ АВТО.....	33
2.1. Визначення варіантів використання та об'єктно-орієнтованої структури системи	33
2.2. Розробка бази даних системи	51
2.3. Проектування та реалізація алгоритмів роботи системи	58
2.4. Реалізація програмного комплексу	66
Висновки до другого розділу	77
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ЗАСТОСУНКОМ 78	
3.1. Порядок встановлення та налаштування параметрів системи.....	78
3.2. Структура інтерфейсу веб-застосунку	81
3.3. Тестування роботи програмного комплексу.....	90
Висновки до третього розділу	93
ВИСНОВКИ	95
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ	98
ДОДАТКИ	101

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API - Application Programming Interface, прикладний програмний інтерфейс, що забезпечує взаємодію між клієнтською та серверною частинами програмного комплексу.

БД - база даних, структуроване сховище інформації, яке використовується для збереження даних про користувачів, автомобілі, оголошення, повідомлення, заявки, відгуки та інші об'єкти системи.

СКБД - система керування базами даних, програмне забезпечення для створення, зберігання, обробки та адміністрування баз даних.

MySQL - реляційна система керування базами даних, що використовується у програмному комплексі для збереження інформації вебзастосунку.

HTML - HyperText Markup Language, мова гіпертекстової розмітки, що використовується для формування структури вебсторінок.

CSS - Cascading Style Sheets, каскадні таблиці стилів, що використовуються для оформлення зовнішнього вигляду сторінок вебзастосунку.

JWT - JSON Web Token, формат токена, що використовується для автентифікації користувачів та перевірки їхніх прав доступу до захищених ресурсів системи.

HTTP - HyperText Transfer Protocol, протокол передавання гіпертекстових даних, що використовується для обміну інформацією між клієнтом і сервером.

REST - Representational State Transfer, архітектурний підхід до організації взаємодії між клієнтською та серверною частинами системи через HTTP-запити.

URL - Uniform Resource Locator, уніфікований покажчик ресурсу, що визначає адресу вебресурсу або API-маршруту.

JSON - JavaScript Object Notation, текстовий формат обміну даними, який використовується для передавання інформації між клієнтом і сервером.

CRUD - Create, Read, Update, Delete, базовий набір операцій створення, читання, оновлення та видалення даних.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		8

UML - Unified Modeling Language, уніфікована мова моделювання, що використовується для візуального опису структури та поведінки програмної системи.

UI - User Interface, користувацький інтерфейс, через який користувач взаємодіє із вебзастосунком.

UX - User Experience, користувацький досвід, що характеризує зручність і логіку взаємодії користувача з програмним продуктом.

ID - унікальний ідентифікатор запису в базі даних.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				9
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

Актуальність теми зумовлена стрімким розвитком цифрових технологій та зростанням популярності онлайн-платформ для купівлі та продажу товарів. Зокрема, вебсервіси з продажу вживаних автомобілів стають невід'ємною частиною сучасного ринку, забезпечуючи швидку, зручну та прозору взаємодію між продавцями та покупцями транспортних засобів.

Щодо веб-платформ для продажу вживаних автомобілів, то вони виступають ключовим інструментом автоматизації процесу розміщення оголошень, пошуку та підбору транспортних засобів. Такі системи надають користувачам можливість швидко отримувати доступ до актуальних пропозицій, переглядати характеристики автомобілів, використовувати фільтри для пошуку за брендом, ціною та містом, а також переглядати детальні описи та фотографії транспортних засобів.

Окрім цього, подібні платформи забезпечують зручний інтерфейс для взаємодії користувача із системою, включаючи можливість додавання оголошень, збереження автомобілів у «обране» та управління власним профілем. Це значно покращує користувацький досвід та підвищує ефективність процесу купівлі-продажу автомобілів. Вебсистема також є важливим інструментом для адміністрації платформи, оскільки дозволяє керувати користувачами, оголошеннями та контролювати вміст системи.

Метою випускної кваліфікаційної роботи є проєктування архітектури та практична реалізація веб-платформи для продажу вживаних автомобілів.

Окреслена мета передбачає виконання таких завдань: опрацювання теоретичних та практичних аспектів створення веб-платформ для електронної комерції; визначення структури та архітектури системи; аналіз існуючих онлайн-сервісів з продажу автомобілів та визначення їх переваг і недоліків; обґрунтування вибору технологічного стеку для реалізації клієнтської та серверної частин системи; проєктування бази даних та логіки роботи системи; реалізація веб-платформи для продажу вживаних автомобілів.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				10
Змн.	Арк.	№ докум.	Підпис	Дата		

Об'єктом дослідження є методи та засоби розробки веборієнтованих інформаційних систем у сфері електронної комерції. Предметом дослідження є використання сучасних веб-технологій для створення платформи з продажу вживаних автомобілів.

У ході виконання роботи були використані методи об'єктно-орієнтованого проєктування, моделювання баз даних, веб-програмування та розробки користувацьких інтерфейсів.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				11
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РЕАЛІЗАЦІЇ ВЕБЗАСТОСУНКУ

1.1. Постановка задачі

Веб-застосунок продажу вживаних автомобілів є інформаційною системою, призначеною для автоматизації основних процесів, пов'язаних із розміщенням оголошень про продаж транспортних засобів, пошуком автомобілів потенційними покупцями та організацією взаємодії між користувачами системи. Основною метою розробки є створення сучасної веб-платформи, яка забезпечить зручний пошук автомобілів, ефективне управління оголошеннями, підтримку комунікації між продавцями та покупцями, а також надасть адміністративні інструменти для контролю функціонування системи. Використання веб-технологій дозволяє об'єднати всі основні операції в єдиному середовищі та спростити процес пошуку, вибору і продажу вживаних автомобілів.

Розроблювана система повинна підтримувати роботу трьох категорій користувачів: гостей, зареєстрованих користувачів та адміністраторів. Гість, який не має облікового запису або не пройшов авторизацію, повинен мати можливість переглядати список доступних автомобілів, здійснювати пошук і фільтрацію оголошень, переглядати детальну інформацію про транспортні засоби, фотографії та відгуки інших користувачів. Такий набір функцій дає змогу ознайомитися з доступними пропозиціями без необхідності реєстрації. Важливим компонентом системи є механізм пошуку та фільтрації, який повинен забезпечувати відбір автомобілів за основними характеристиками, зокрема моделлю, роком випуску, пробігом, типом пального, типом трансмісії та ціною. Додатково повинна бути реалізована можливість сортування отриманих результатів, що дозволить користувачеві швидше знаходити найбільш відповідні пропозиції.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				12
Змн.	Арк.	№ докум.	Підпис	Дата		

Після реєстрації та авторизації користувач отримує розширений набір можливостей. Авторизований користувач може створювати власні оголошення про продаж автомобілів, редагувати інформацію про них, видаляти оголошення та керувати фотографіями транспортного засобу. Для забезпечення зручного представлення автомобіля система повинна дозволити додавати та видаляти окремі фотографії оголошення. Крім того, користувач може додавати автомобілі до обраного, залишати відгуки, надсилати заявки на придбання автомобіля, переглядати статуси власних заявок, вести листування з власниками транспортних засобів через вбудовану систему повідомлень та редагувати дані власного профілю. Таким чином, після авторизації користувач отримує можливість не лише переглядати інформацію, але й повноцінно взаємодіяти з іншими учасниками системи.

Оголошення повинно містити основні характеристики автомобіля, зокрема модель, рік випуску, пробіг, тип пального, тип трансмісії, ціну та інші необхідні відомості. Під час створення оголошення система повинна виконувати перевірку VIN-коду автомобіля та забезпечувати його унікальність у базі даних. Це необхідно для запобігання дублюванню одного й того самого транспортного засобу в системі. У процесі редагування оголошення користувач також може змінювати встановлену ціну, при цьому система повинна зберігати історію таких змін. Збереження історії зміни ціни дозволяє відстежувати динаміку вартості автомобіля та забезпечує додаткову інформацію для аналізу оголошення. Ще однією функцією для власника є перегляд статистики відвідувань власних оголошень, що дає змогу оцінювати рівень зацікавленості потенційних покупців.

Окреме місце в системі займає механізм заявок на придбання автомобілів. Авторизований користувач, зацікавлений у певному транспортному засобі, повинен мати можливість надіслати заявку його власнику. Після цього покупець може переглядати статус власної заявки та отримувати інформацію про прийняте рішення. Власник автомобіля повинен мати доступ до отриманих заявок і можливість підтверджувати або відхиляти

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				13
Змн.	Арк.	№ докум.	Підпис	Дата		

їх. Для забезпечення безпосередньої взаємодії між учасниками процесу продажу передбачається система обміну повідомленнями. Вона повинна дозволяти продавцям і потенційним покупцям вести листування без використання зовнішніх засобів комунікації. Наявність заявок та повідомлень забезпечує формалізований процес взаємодії та створює можливість для обговорення умов продажу автомобіля.

Користувач також повинен мати можливість залишати відгуки про транспортні засоби, що дозволяє накопичувати додаткову інформацію та формувати оцінку пропозицій на основі досвіду інших користувачів. Водночас система повинна забезпечувати належний контроль доступу до відгуків, оголошень та інших даних. Власник автомобіля може керувати лише власними оголошеннями, тоді як доступ до адміністративних операцій має бути обмежений відповідною роллю. Для відновлення доступу до облікового запису необхідно реалізувати механізм відновлення пароля за одноразовим токеном з обмеженим строком дії. Такий підхід дозволяє відновити пароль у разі його втрати та водночас підвищує рівень безпеки користувацьких облікових записів.

Адміністратор системи має розширені права доступу та відповідає за контроль контенту платформи. Він повинен мати можливість переглядати інформацію про зареєстрованих користувачів та розміщені автомобілі, а також видаляти користувачів і оголошення у разі порушення правил користування системою. Адміністративний модуль є необхідним для забезпечення порядку в інформаційному середовищі та контролю коректності розміщених даних. Реалізація системи ролей і розмежування доступу повинна гарантувати, що кожен користувач може виконувати лише ті операції, які відповідають його правам.

Для реалізації зазначеного функціоналу система повинна містити взаємопов'язані програмні модулі. До них належать модуль реєстрації та авторизації користувачів, модуль керування профілем, модуль роботи з оголошеннями, модуль завантаження та зберігання фотографій, модуль

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				14
Змн.	Арк.	№ докум.	Підпис	Дата		

пошуку та фільтрації, модуль відгуків, модуль обробки заявок на придбання автомобілів, модуль обміну повідомленнями та адміністративний модуль. Також необхідно реалізувати модуль відновлення пароля за одноразовим токеном, перевірку унікальності VIN-коду, ведення історії зміни ціни, статистику переглядів власних оголошень та механізм керування окремими фотографіями оголошення. Взаємодія між цими компонентами повинна забезпечувати цілісну роботу всієї системи та узгоджене виконання основних операцій.

Для реалізації поставленої мети необхідно провести аналіз предметної області та існуючих веб-платформ продажу автомобілів, визначити основні особливості взаємодії користувачів і сформулювати функціональні та нефункціональні вимоги до майбутньої системи. На основі отриманих вимог необхідно розробити архітектуру веб-застосунку та визначити структуру його компонентів, а також спроектувати базу даних для зберігання інформації про користувачів, автомобілі, фотографії, повідомлення, заявки, відгуки та інші необхідні дані. Серверна частина повинна забезпечувати обробку клієнтських запитів, реалізацію бізнес-логіки та взаємодію з базою даних, а клієнтська частина - надавати зручний та адаптивний користувацький інтерфейс.

Важливими технічними завданнями є реалізація механізму надсилання заявок та їх подальшого підтвердження або відхилення власником, організація системи обміну повідомленнями, а також розробка розширеної фільтрації та сортування оголошень. Окрему увагу необхідно приділити перевірці VIN-коду та контролю його унікальності, механізму відновлення пароля з обмеженим строком дії токена, збереженню історії зміни ціни та збору статистики переглядів. Усі ці функції повинні працювати у взаємодії з єдиною базою даних та забезпечувати актуальність інформації.

Не менш важливим завданням є забезпечення захисту даних користувачів та безпечної авторизації в системі. Необхідно реалізувати розмежування доступу відповідно до ролей, перевірку прав користувачів під час виконання операцій та захист конфіденційної інформації. Серверна

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				15
Змн.	Арк.	№ докум.	Підпис	Дата		

частина повинна перевіряти дані, що надходять від клієнта, а функції роботи з оголошеннями, заявками, повідомленнями та профілями повинні бути доступними лише уповноваженим користувачам. Це дозволить зменшити ризик несанкціонованого доступу та забезпечити належний рівень надійності програмного рішення.

Таким чином, результатом виконання поставлених завдань має стати повнофункціональний веб-застосунок продажу вживаних автомобілів, який забезпечить зручну взаємодію між продавцями та покупцями, автоматизує процес розміщення та керування оголошеннями, спростить пошук транспортних засобів і забезпечить необхідні засоби комунікації між користувачами. Реалізація пошуку та фільтрації, заявок, повідомлень, відгуків, обраного, перевірки VIN-коду, історії зміни ціни, статистики переглядів, відновлення пароля та адміністративного контролю дозволить сформувати цілісну інформаційну систему для підтримки основних процесів купівлі-продажу вживаних автомобілів.

1.2. Аналіз аналогів програмного продукту

Стрімкий розвиток інформаційних технологій та широке поширення мережі Інтернет суттєво змінили підходи до купівлі та продажу транспортних засобів. Якщо раніше пошук автомобіля здійснювався переважно через друковані видання, автомобільні ринки або особисті знайомства, то сьогодні більшість угод розпочинається саме з використання спеціалізованих веб-ресурсів.

Онлайн-платформи продажу автомобілів дозволяють потенційним покупцям швидко знаходити необхідні транспортні засоби за заданими параметрами, переглядати технічні характеристики, фотографії та контактні дані продавця. У свою чергу продавці отримують можливість оперативно розміщувати інформацію про власні автомобілі та взаємодіяти з великою кількістю потенційних покупців.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				16
Змн.	Арк.	№ докум.	Підпис	Дата		

Незважаючи на значну кількість існуючих рішень, проблема ефективної взаємодії між продавцем і покупцем залишається актуальною. Більшість платформ зосереджуються на процесі публікації оголошень, тоді як механізми подальшої комунікації між користувачами часто реалізовані недостатньо ефективно або потребують використання сторонніх сервісів зв'язку. Для визначення функціональних вимог до розроблюваної системи було проведено аналіз найбільш популярних платформ продажу автомобілів.

Проведемо аналіз платформи AUTO.RIA. AUTO.RIA є одним із найбільших автомобільних порталів України та займає провідні позиції серед онлайн-сервісів купівлі-продажу транспортних засобів. Платформа надає користувачам можливість розміщення оголошень, пошуку автомобілів, перегляду характеристик транспортних засобів та взаємодії з продавцями [1].

Основною перевагою AUTO.RIA є велика база автомобілів, що постійно оновлюється. Користувачі можуть виконувати пошук за значною кількістю параметрів, серед яких марка, модель, рік випуску, тип двигуна, тип коробки передач, пробіг, регіон продажу та ціновий діапазон. Крім того, сервіс підтримує завантаження великої кількості фотографій, що дозволяє потенційному покупцю отримати повніше уявлення про стан транспортного засобу.

Важливою особливістю AUTO.RIA є можливість перевірки транспортного засобу за VIN-кодом, що підвищує рівень довіри до розміщених оголошень та зменшує ризики для покупців.

Разом з тим система має певні недоліки. Частина функціоналу є платною, а просування оголошень потребує додаткових фінансових витрат.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				17
Змн.	Арк.	№ докум.	Підпис	Дата		

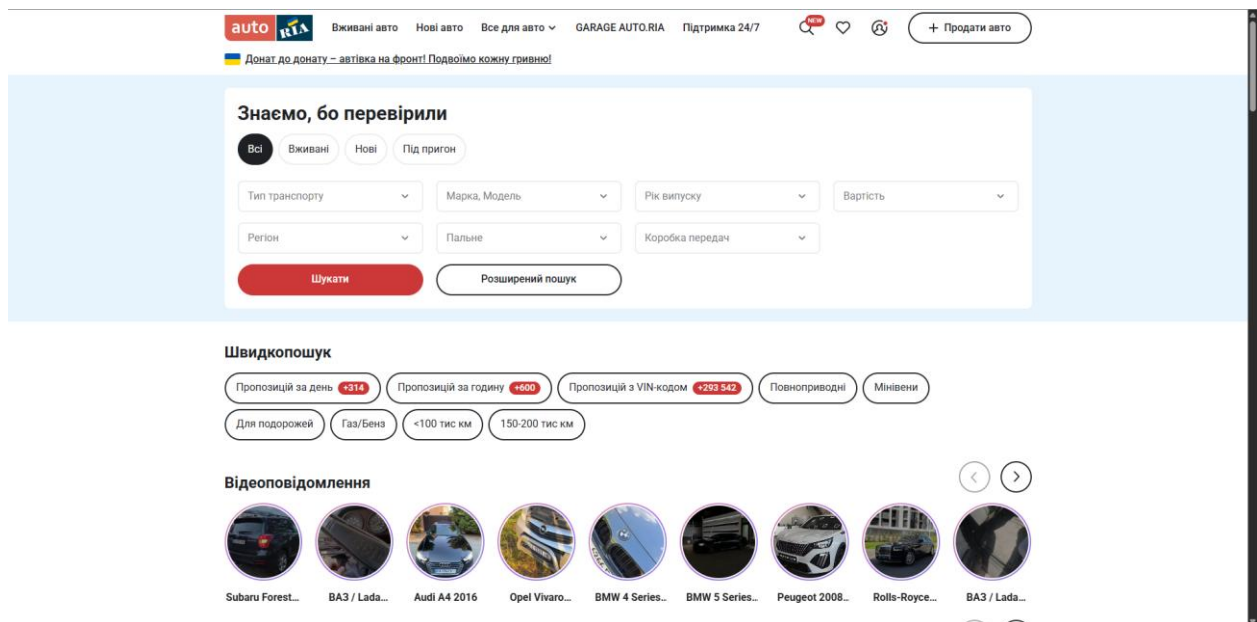


Рис. 1.1. Відображення головної сторінки сайту Auto.ria

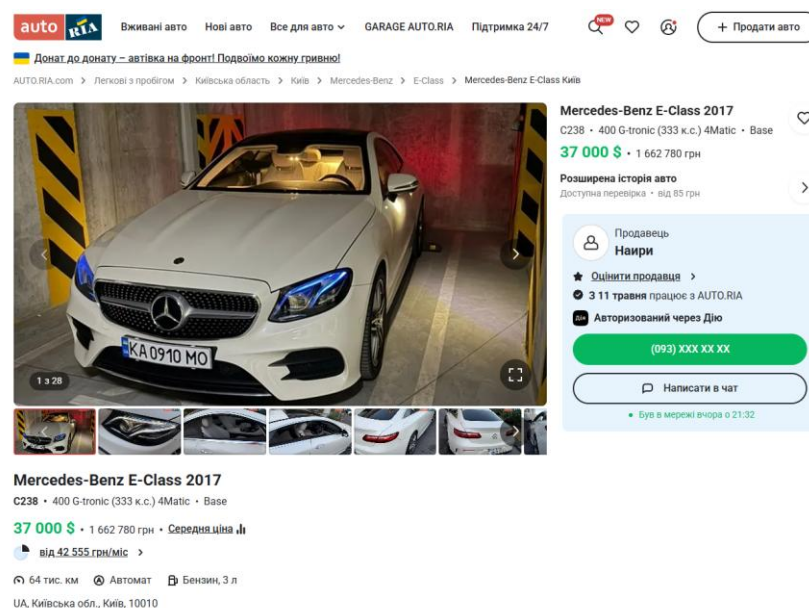


Рис. 1.2. Відображення сторінки оголошення сайту Auto.ria

Проведемо аналіз платформи RST. RST є ще одним популярним українським ресурсом для продажу транспортних засобів. Основною перевагою сервісу є простота інтерфейсу та висока швидкість роботи [2].

Платформа забезпечує базові можливості для пошуку автомобілів, створення оголошень та перегляду характеристик транспортних засобів. Користувачі можуть завантажувати фотографії, редагувати інформацію про автомобілі та підтримувати зв'язок із продавцями.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		18

Перевагою RST є простота використання та невисокі вимоги до швидкості інтернет-з'єднання. Водночас сервіс має менш розвинені інструменти взаємодії між користувачами та поступається AUTO.RIA за кількістю доступних автомобілів.

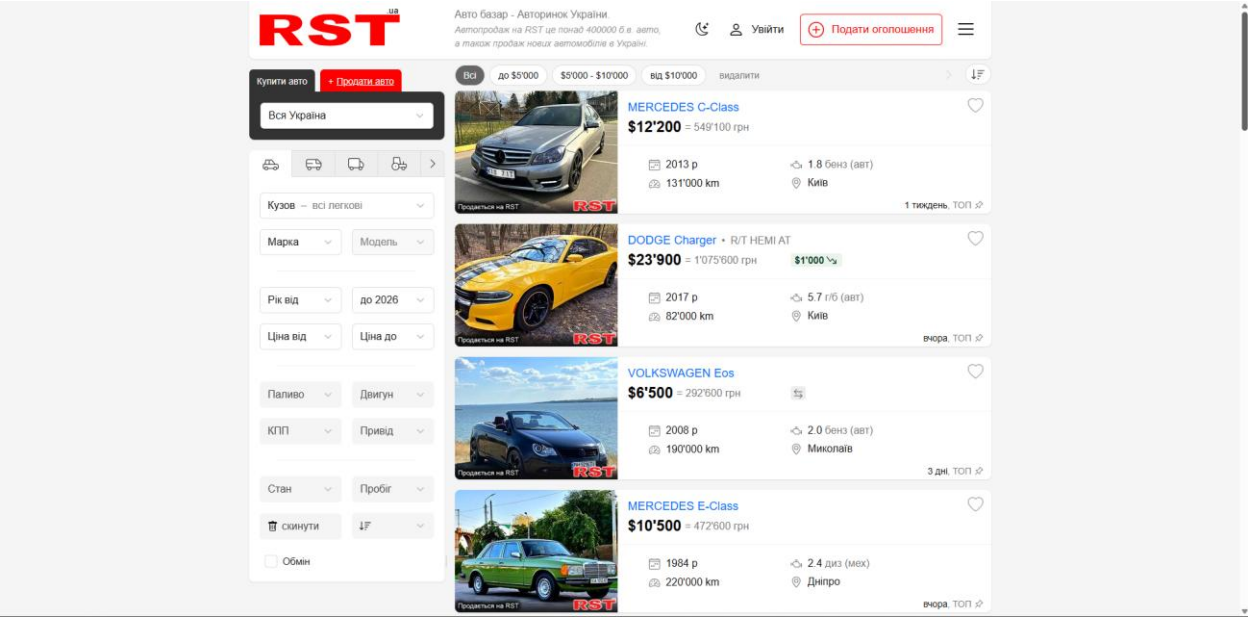


Рис. 1.3. – Відображення головної сторінки сайту RST

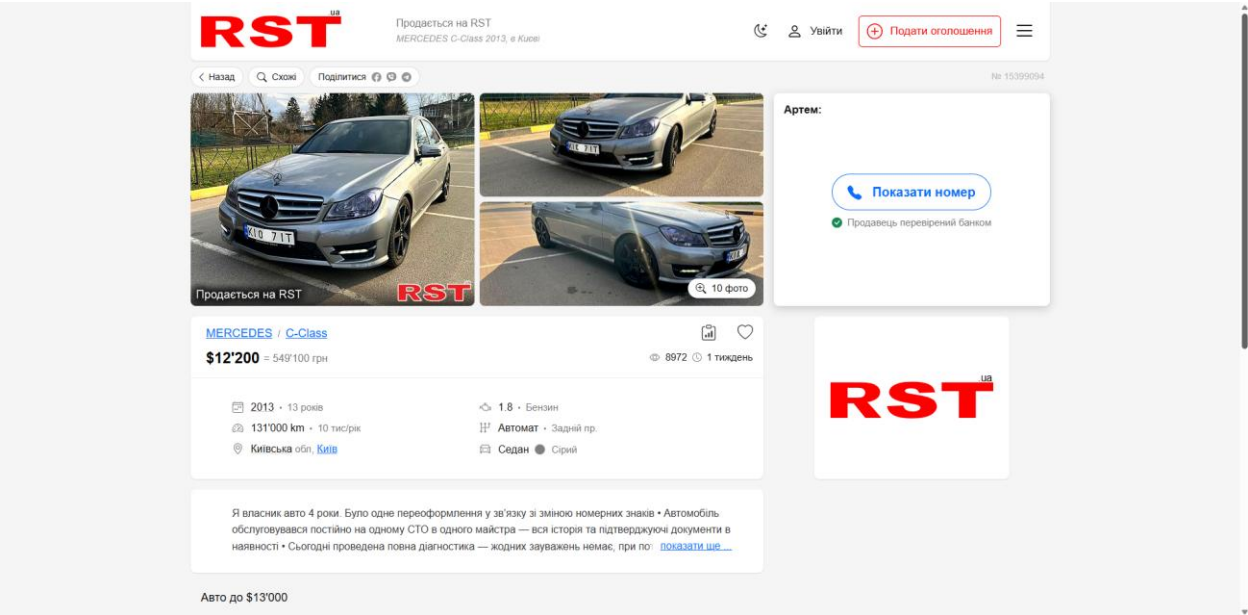


Рис. 1.4. – Відображення сторінки оголошення сайту RST

Для проведення об'єктивної оцінки було виконано порівняння функціональних можливостей існуючих платформ із функціоналом розроблюваної системи.

Виходячи з наявних переваг та недоліків, складемо порівняльну таблицю вищезгаданих програмних аналогів (табл. 1.1).

Таблиця 1.1

Порівняння функціональних можливостей систем

Назва	AUTO.RIA	RST	Власна система
Перегляд автомобілів	+	+	+
Розширена фільтрація	+	частково	+
Сортування оголошень	+	частково	+
Додавання оголошень	+	+	+
Завантаження та керування фото	+	+	+
Перевірка унікальності VIN-коду	+	-	+
Історія зміни ціни	+	-	+
Відгуки користувачів	+	-	+
Надсилання заявки на купівлю	-	-	+
Підтвердження заявки власником	-	-	+
Вбудований чат між користувачами	+	-	+
Особистий кабінет із заявками та статистикою	+	частково	+
Відновлення пароля	+	+	+
Адміністрування користувачів та оголошень	+	+	+

Порівняння показало, що більшість існуючих платформ забезпечують якісну реалізацію основних функцій пошуку та публікації оголошень. Проте функції підтримки процесу прийняття рішення про купівлю реалізовані недостатньо повно.

Зокрема, у більшості систем відсутній механізм офіційного надсилання заявки на придбання автомобіля із подальшим підтвердженням або відхиленням власником транспортного засобу. Також часто відсутня інтегрована система обміну повідомленнями між користувачами без використання сторонніх сервісів.

Особливості розроблюваної системи

На основі проведеного аналізу було визначено перелік функціональних можливостей, які доцільно реалізувати у власній системі. Основними особливостями розроблюваного веб-застосунку є підтримка трьох категорій

користувачів, а саме гостя, зареєстрованого користувача та адміністратора, кожна з яких має відповідний набір функціональних можливостей. Система повинна забезпечувати створення, редагування та видалення оголошень про продаж автомобілів, а також надавати можливість потенційним покупцям надсилати заявки на придбання обраного транспортного засобу. Власник автомобіля, у свою чергу, може переглядати отримані заявки та приймати рішення щодо їх підтвердження або відхилення. Для безпосередньої комунікації між учасниками процесу купівлі-продажу передбачена вбудована система обміну повідомленнями, яка дозволяє користувачам вести листування без використання сторонніх сервісів. Додатковою функціональною можливістю є система відгуків про автомобілі, що дає змогу користувачам ділитися власною думкою та досвідом. Адміністратор має розширені права для контролю зареєстрованих користувачів і розміщених оголошень, а також для підтримки належного стану контенту системи. Крім того, веб-застосунок передбачає зберігання історії дій користувачів, що дає змогу відстежувати основні операції, виконані в межах системи, та може використовуватися для контролю її роботи.

Таким чином проведений аналіз аналогів дозволив визначити основні функціональні вимоги до розроблюваної системи та сформулювати перелік можливостей, які повинні бути реалізовані під час проектування програмного продукту.

1.3. Вибір архітектури веб-застосунку

Архітектура програмного забезпечення є одним із найважливіших етапів проектування інформаційної системи, оскільки визначає структуру програмного продукту, принципи взаємодії між його компонентами та можливості подальшого розвитку системи. Правильно обрана архітектура забезпечує високу продуктивність, надійність, масштабованість та зручність супроводу програмного забезпечення протягом усього життєвого циклу [11].

Під час розробки веб-застосунку продажу вживаних автомобілів було проведено аналіз найбільш поширених архітектурних підходів, які

		Поліщук І. А.			ДУ «Житомирська політехніка». 26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				21
Змн.	Арк.	№ докум.	Підпис	Дата		

використовуються під час створення сучасних веб-систем. Основна увага приділялася монолітній, багаторівневій та мікросервісній архітектурі.

Монолітна архітектура передбачає реалізацію всіх функціональних компонентів системи у межах одного програмного застосунку. У цьому випадку модулі авторизації, керування автомобілями, роботи із заявками, відгуками, повідомленнями та адміністрування працюють як єдина система.

Основною перевагою монолітного підходу є простота розробки та розгортання. Усі компоненти взаємодіють між собою без використання додаткових механізмів передачі даних між сервісами, що позитивно впливає на швидкодію системи.

До основних переваг монолітної архітектури належить простота її реалізації та подальшого супроводу, оскільки всі компоненти системи функціонують у межах єдиного програмного середовища. Взаємодія між окремими модулями відбувається без необхідності використання додаткових мережових механізмів, що забезпечує високу швидкість обміну даними. Об'єднана структура застосунку також спрощує процес його тестування, оскільки всі основні компоненти можуть перевірятися в межах єдиної системи. Крім того, монолітний підхід забезпечує спрощене розгортання програмного продукту, адже для запуску системи не потрібно окремо налаштовувати велику кількість незалежних сервісів. Використання такої архітектури також дозволяє зменшити витрати на підтримку інфраструктури, оскільки для функціонування застосунку достатньо забезпечити роботу єдиного програмного середовища.

Недоліком монолітної архітектури є те, що зі збільшенням розміру проєкту система може ускладнюватися, а внесення змін до окремих модулів може потребувати повторного розгортання всього застосунку.

Багаторівнева архітектура. Одним із найпоширеніших підходів до побудови сучасних веб-систем є багаторівнева архітектура [12]. Дана архітектура передбачає розділення програмного забезпечення на окремі логічні рівні, кожен з яких відповідає за виконання визначеного набору

		Поліщук І. А.			ДУ «Житомирська політехніка». 26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				22
Змн.	Арк.	№ докум.	Підпис	Дата		

функцій. Такий підхід дозволяє структурувати програмний код, розмежувати відповідальність між окремими компонентами системи та спростити її подальшу розробку і супровід. Як правило, у межах багаторівневої архітектури виділяють три основні рівні: рівень представлення (Presentation Layer), рівень бізнес-логіки (Business Logic Layer) та рівень даних (Data Access Layer).

Рівень представлення відповідає за взаємодію користувача із системою та забезпечує відображення необхідної інформації у веб-браузері. Саме на цьому рівні реалізуються елементи користувацького інтерфейсу, форми введення даних, сторінки перегляду інформації та інші компоненти, призначені для безпосередньої взаємодії з користувачем. Рівень бізнес-логіки забезпечує обробку запитів, що надходять від користувача, перевірку коректності введених даних, виконання основних правил функціонування системи та формування відповідних результатів. Він є проміжною ланкою між інтерфейсом користувача та рівнем роботи з даними, забезпечуючи виконання необхідних операцій відповідно до визначеної логіки веб-застосунку. Рівень даних відповідає за безпосередню взаємодію з базою даних та виконання операцій зі збереження, редагування, видалення й отримання інформації. Такий розподіл дозволяє ізолювати роботу з даними від інших компонентів системи та забезпечити більш упорядковану структуру програмного забезпечення.

До основних переваг багаторівневої архітектури належать висока модульність системи, спрощення внесення змін і вдосконалення окремих компонентів, а також покращена підтримуваність програмного коду. Розподіл функціональності між рівнями дає можливість змінювати або розширювати окремі частини програмного продукту з мінімальним впливом на інші компоненти. Крім того, такий підхід створює можливість незалежного розвитку окремих частин системи, що особливо важливо під час її масштабування та подальшого розширення функціональних можливостей. Багаторівнева архітектура є поширеним підходом під час розробки сучасних

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				23
Змн.	Арк.	№ докум.	Підпис	Дата		

веб-застосунків, оскільки забезпечує чітке розмежування відповідальності між компонентами та сприяє створенню структурованого, зручного для супроводу і розвитку програмного забезпечення.

Мікросервісна архітектура передбачає поділ програмного забезпечення на окремі незалежні сервіси, кожен з яких відповідає за виконання певної функції системи. У межах веб-застосунку продажу автомобілів у якості окремих сервісів можуть бути виділені сервіс авторизації користувачів, сервіс роботи з автомобілями та оголошеннями, сервіс обміну повідомленнями, сервіс відгуків, сервіс обробки заявок на придбання автомобілів та адміністративний сервіс. Кожен із таких компонентів може функціонувати незалежно від інших і взаємодіяти з ними за допомогою програмних інтерфейсів API. Такий підхід забезпечує чітке розмежування функціональних компонентів системи та дозволяє змінювати або масштабувати окремі сервіси без необхідності модифікації всього програмного продукту [13].

Основною перевагою мікросервісної архітектури є можливість незалежного масштабування окремих компонентів відповідно до рівня навантаження. Наприклад, сервіс обміну повідомленнями або роботи з оголошеннями може потребувати значно більше ресурсів порівняно з іншими частинами системи, тому його можна масштабувати окремо. Крім того, розробка та супровід окремих сервісів можуть виконуватися різними командами програмістів, що дозволяє паралельно працювати над декількома функціональними частинами системи. Однак використання мікросервісної архітектури має і певні недоліки. Такий підхід потребує додаткових витрат на серверну інфраструктуру, налаштування взаємодії між сервісами, забезпечення їхньої надійності та моніторинг працездатності окремих компонентів. Також ускладнюється процес розгортання, тестування та діагностики системи, оскільки замість одного застосунку необхідно забезпечувати стабільну роботу декількох незалежних сервісів. У зв'язку з цим для невеликих і середніх веб-застосунків використання мікросервісної

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				24
Змн.	Арк.	№ докум.	Підпис	Дата		

архітектури може бути надлишковим і призводити до невиправданого ускладнення програмної та інфраструктурної частини проєкту.

Для визначення найбільш доцільного підходу до побудови розроблюваного веб-застосунку було проведено порівняння розглянутих архітектурних рішень за основними критеріями, зокрема складністю реалізації та супроводу, можливістю масштабування, зручністю тестування, вимогами до інфраструктури та доцільністю використання для системи відповідного масштабу. Результати проведеного порівняння наведено у таблиці (табл 1.2).

На основі наведеного порівняння можна визначити найбільш доцільний архітектурний підхід для реалізації веб-застосунку продажу вживаних автомобілів з урахуванням його функціональних вимог, масштабу та необхідності забезпечення простоти розробки й подальшого супроводу.

Таблиця 1.2

Порівняння архітектурних рішень

Критерій	Монолітна	Багаторівнева	Мікросервісна
Простота реалізації	Висока	Висока	Низька
Масштабованість	Середня	Висока	Дуже висока
Простота супроводу	Висока	Висока	Середня
Вартість розробки	Низька	Середня	Висока
Вартість підтримки	Низька	Середня	Висока
Продуктивність	Висока	Висока	Середня
Доцільність для побудови проєкту	Висока	Висока	Низька

На основі проведеного аналізу для реалізації веб-застосунку продажу вживаних автомобілів було обрано багаторівневу клієнт-серверну архітектуру монолітного типу. Обрана архітектура складається з трьох основних рівнів:

- клієнтської частини;
- серверної частини;
- бази даних.

Клієнтська частина реалізована за допомогою HTML, CSS та JavaScript і забезпечує взаємодію користувача із системою через веб-браузер. Серверна частина реалізована на платформі Node.js та відповідає за обробку HTTP-

запитів, перевірку прав доступу, реалізацію бізнес-логіки та взаємодію з базою даних. Зберігання інформації здійснюється за допомогою СКБД MySQL, у якій розміщуються дані про користувачів, автомобілі, заявки на придбання, повідомлення, фотографії та відгуки.

Обрана архітектура забезпечує достатню продуктивність, простоту розробки та супроводу, а також дозволяє реалізувати всі функціональні вимоги до системи. Крім того, така структура дозволяє у майбутньому розширювати функціональні можливості веб-застосунку без необхідності повного перепроєктування програмного забезпечення.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Сучасні веб-застосунки потребують використання надійних інструментальних засобів, які забезпечують ефективну розробку, супровід та масштабування програмного забезпечення. Вибір технологій для реалізації програмного продукту безпосередньо впливає на його продуктивність, стабільність роботи, безпеку та зручність використання.

Під час розробки веб-застосунку продажу вживаних автомобілів було обрано комплекс сучасних програмних засобів та технологій, які дозволяють реалізувати функціональні вимоги системи, забезпечити надійне зберігання інформації та підтримати взаємодію між користувачами.

До основних інструментальних засобів, використаних у процесі розробки, належать: HTML, CSS, JavaScript, Node.js, MySQL, JWT, bcrypt, Multer.

Використання HTML. HTML (HyperText Markup Language) є стандартною мовою розмітки гіпертекстових документів, яка використовується для формування структури та вмісту веб-сторінок [14;15]. У розроблюваному веб-застосунку HTML використовується як основа клієнтської частини для побудови структури основних сторінок і компонентів користувацького інтерфейсу. Зокрема, за допомогою HTML реалізовано

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		26

головну сторінку веб-застосунку, сторінки реєстрації та авторизації користувачів, сторінки перегляду інформації про автомобілі, форми створення та редагування оголошень, інтерфейс роботи з повідомленнями, а також елементи адміністративної панелі [17]. Застосування HTML дозволяє логічно структурувати інформацію та забезпечити її коректне відображення у веб-браузері. Важливою перевагою цієї технології є універсальність, простота використання та підтримка сучасними веб-браузерами.

Крім того, HTML забезпечує зручну взаємодію з іншими технологіями веб-розробки, зокрема CSS для стилізації та JavaScript для реалізації динамічної поведінки інтерфейсу. Таким чином, використання HTML стало основою для побудови структурованого та функціонального користувацького інтерфейсу розробленого веб-застосунку.

Використання CSS. Для оформлення зовнішнього вигляду та забезпечення зручності користування веб-застосунком використовується CSS (Cascading Style Sheets) - каскадні таблиці стилів [16]. Дана технологія призначена для визначення візуального представлення HTML-документів і дозволяє відокремити структуру веб-сторінки від її оформлення. У розробленій системі CSS використовується для стилізації окремих сторінок та елементів інтерфейсу, оформлення навігаційного меню, таблиць, форм і карток автомобілів, а також для відображення повідомлень та інших компонентів взаємодії користувача із системою.

Важливою особливістю використання CSS є можливість створення адаптивного інтерфейсу, який забезпечує коректне відображення веб-застосунку на пристроях із різними розмірами екранів. Це дозволяє зберігати зручність використання системи як на персональних комп'ютерах, так і на мобільних пристроях. Використання CSS також сприяє забезпеченню єдиного візуального стилю всіх сторінок веб-застосунку, що покращує сприйняття інформації та загальну зручність роботи користувачів.

JavaScript є основною мовою програмування клієнтської частини розробленого веб-застосунку та використовується для реалізації динамічної та

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				27
Змн.	Арк.	№ докум.	Підпис	Дата		

інтерактивної поведінки веб-сторінок. На відміну від статичної розмітки, застосування JavaScript дозволяє обробляти дії користувача, виконувати необхідні операції без постійного перезавантаження сторінки та забезпечувати оперативне оновлення інформації [19;21]. У розробленій системі JavaScript використовується для перевірки коректності введених користувачем даних, формування та надсилання запитів до сервера, реалізації механізмів пошуку і фільтрації автомобілів, обробки отриманих від сервера відповідей, а також динамічного оновлення вмісту веб-сторінок.

Крім того, за допомогою JavaScript реалізовано відображення повідомлень про результати виконання операцій та інших змін стану системи. Застосування JavaScript дозволяє підвищити інтерактивність веб-застосунку, зменшити кількість непотрібних перезавантажень сторінок і забезпечити більш швидку та комфортну взаємодію користувача із системою [22].

Для реалізації серверної частини розробленого веб-застосунку використовується середовище виконання JavaScript Node.js, яке дозволяє виконувати програмний код на стороні сервера. Застосування Node.js забезпечує обробку HTTP-запитів і виконання основної серверної логіки системи, а також дозволяє ефективно працювати з великою кількістю одночасних запитів[23]. У межах розробленого веб-застосунку серверна частина відповідає за авторизацію користувачів, перевірку їхніх прав доступу, взаємодію з базою даних, обробку заявок на придбання автомобілів, повідомлень між користувачами та відгуків, а також керування інформацією про транспортні засоби та оголошення. Сервер забезпечує обмін даними між клієнтською частиною та базою даних, виконує необхідні перевірки та реалізує основні правила функціонування системи.

Однією з важливих переваг Node.js є можливість використання JavaScript як для клієнтської, так і для серверної частини веб-застосунку. Це дозволяє застосовувати єдиний програмний стек під час розробки та спрощує підтримку й подальше розширення програмного продукту. Таким чином, використання Node.js є доцільним для реалізації серверної частини

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				28
Змн.	Арк.	№ докум.	Підпис	Дата		

розробленої системи та забезпечення її взаємодії з користувачами і базою даних.

Для зберігання та обробки інформації у розробленому веб-застосунку використовується реляційна система управління базами даних MySQL. Її вибір зумовлений високою продуктивністю, надійністю зберігання інформації, підтримкою стандарту SQL, можливістю роботи з великими обсягами даних та відносною простотою адміністрування. Реляційна модель дозволяє представити інформацію у вигляді взаємопов'язаних таблиць і забезпечити чітку структуру даних [27]. У базі даних зберігається інформація про зареєстрованих користувачів, автомобілі та їхні характеристики, фотографії транспортних засобів, заявки на придбання, повідомлення між користувачами, відгуки, ролі користувачів та інші службові дані, необхідні для функціонування системи. Для кожної сутності передбачено відповідні таблиці та зв'язки між ними, що забезпечує узгодженість і цілісність інформації.

Особливістю розробленої бази даних є використання окремої таблиці Messages, яка призначена для збереження повідомлень між продавцями та потенційними покупцями в межах конкретного оголошення про продаж автомобіля. Це дозволяє зберігати історію листування та забезпечувати її зв'язок із відповідним транспортним засобом. Використання реляційної моделі даних також сприяє зменшенню дублювання інформації, спрощує її обробку та забезпечує надійне зберігання даних, необхідних для роботи веб-застосунку.

Для реалізації механізму автентифікації та авторизації користувачів у розробленому веб-застосунку використовується технологія JWT (JSON Web Token). Після успішного проходження процедури входу сервер формує спеціальний токен, який містить необхідну інформацію про користувача, зокрема його ідентифікатор та роль у системі. Отриманий токен використовується під час подальших запитів до захищених ресурсів, що дозволяє серверу ідентифікувати користувача та визначати наявність відповідних прав доступу [30]. Використання JWT спрощує реалізацію

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				29
Змн.	Арк.	№ докум.	Підпис	Дата		

механізму авторизації та дозволяє забезпечити розмежування доступу відповідно до ролей користувачів. Крім того, такий підхід дає можливість зменшити необхідність постійного зберігання інформації про стан авторизації на сервері, що позитивно впливає на ефективність роботи системи. Таким чином, застосування JWT забезпечує надійну ідентифікацію користувачів та спрощує контроль доступу до функціональних можливостей веб-застосунку [31].

Одним із важливих аспектів розробки веб-застосунку є забезпечення безпеки персональних даних користувачів, зокрема захисту їхніх паролів. Для цього використовується бібліотека `bcrypt`, призначена для безпечного хешування паролів. Під час реєстрації введений користувачем пароль не зберігається у відкритому вигляді, а попередньо перетворюється на хешоване значення, яке після цього записується до бази даних [33]. Під час авторизації введений пароль порівнюється з відповідним хешем, що дозволяє перевірити правильність облікових даних без збереження самого пароля. Такий підхід значно підвищує рівень захисту облікових записів, оскільки навіть у разі отримання несанкціонованого доступу до бази даних збережені значення не містять паролів користувачів у відкритому вигляді. Використання `bcrypt` таким чином забезпечує додатковий рівень захисту конфіденційної інформації та підвищує безпеку системи автентифікації [34].

Оскільки розроблений веб-застосунок передбачає можливість додавання фотографій автомобілів до оголошень, для реалізації завантаження файлів використовується бібліотека `Multer`. Вона призначена для обробки даних, що надходять від користувача у форматі `multipart/form-data`, та забезпечує зручний механізм приймання файлів серверною частиною [35]. У межах розробленої системи `Multer` використовується для отримання фотографій від користувачів, перевірки параметрів завантажених файлів і їх подальшого збереження на сервері. Інформація про завантажені файли може бути пов'язана з відповідним оголошенням та збережена в базі даних. Використання даної бібліотеки спрощує реалізацію функціоналу роботи із

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				30
Змн.	Арк.	№ докум.	Підпис	Дата		

зображеннями автомобілів та дозволяє організувати їх завантаження і зберігання у структурованому вигляді.

Висновки до першого розділу

У першому розділі кваліфікаційної роботи було проведено аналіз предметної області, що стосується розробки веб-застосунків для організації процесу купівлі та продажу вживаних автомобілів. Дослідження показало, що використання сучасних інформаційних технологій значно спрощує взаємодію між продавцями та покупцями, автоматизує процес розміщення оголошень, пошуку транспортних засобів і супроводу угоди. Водночас існуючі програмні продукти не завжди забезпечують повний цикл взаємодії користувачів у межах однієї системи, що обумовлює актуальність розробки власного веб-застосунку.

У процесі постановки задачі було визначено основну мету роботи та сформовано перелік функціональних завдань, які необхідно реалізувати під час розробки програмного забезпечення. Встановлено, що система повинна підтримувати роботу декількох категорій користувачів із різними правами доступу: гостей, зареєстрованих користувачів та адміністраторів. Для кожної категорії визначено функціональні можливості, що забезпечують зручну та безпечну роботу із системою.

Проведений аналіз існуючих аналогів дозволив дослідити сучасні підходи до реалізації веб-платформ продажу автомобілів, визначити їх переваги та недоліки. Було встановлено, що більшість наявних систем забезпечують ефективний пошук транспортних засобів та можливість публікації оголошень, проте недостатньо автоматизують процес взаємодії між продавцем і покупцем. На основі проведеного порівняльного аналізу сформовано перелік функціональних можливостей, які доцільно реалізувати у власному програмному продукті.

Під час аналізу можливих архітектурних рішень було розглянуто різні підходи до побудови веб-застосунків. За результатами порівняння встановлено, що для розроблюваної системи найбільш доцільним є

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				31
Змн.	Арк.	№ докум.	Підпис	Дата		

використання монолітної клієнт-серверної архітектури. Обрана архітектура забезпечує високу продуктивність, простоту реалізації, зручність супроводу програмного забезпечення та можливість його подальшого розвитку.

Також було виконано обґрунтування вибору інструментальних засобів реалізації програмного продукту. Для створення клієнтської частини застосовано HTML, CSS та JavaScript, що забезпечують формування структури веб-сторінок, їх оформлення та інтерактивну взаємодію з користувачем. Серверна частина реалізується на основі середовища Node.js, а для зберігання інформації використовується система управління базами даних MySQL. Для реалізації механізмів автентифікації та захисту персональних даних використано технологію JWT і бібліотеку bcrypt, а для роботи із завантаженням фотографій автомобілів - бібліотеку Multer.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				32
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКУ ПРОДАЖУ ВЖИВАНИХ АВТО

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

Проектування програмного забезпечення є одним із найважливіших етапів створення інформаційної системи, оскільки саме на цьому етапі визначаються функціональні можливості майбутнього програмного продукту, вимоги до його роботи, структура взаємодії між окремими компонентами та принципи організації даних. Від правильності прийнятих проєктних рішень залежить зручність використання системи, її продуктивність, можливість подальшого розвитку та супроводу [3;4].

Основною метою розроблюваного веб-застосунку є створення інформаційної системи, яка забезпечує повний цикл взаємодії між продавцем та потенційним покупцем автомобіля. На відміну від багатьох існуючих сервісів, система не обмежується лише розміщенням оголошень, а надає користувачам можливість вести листування, надсилати заявки на придбання автомобілів, залишати відгуки та керувати власними оголошеннями.

Під час проектування були визначені категорії користувачів системи, функціональні та нефункціональні вимоги, вимоги до швидкодії, а також сформовані основні сценарії використання програмного продукту. Крім того, була побудована об'єктно-орієнтована модель системи, яка описує взаємозв'язки між основними сутностями програмного комплексу [6;8].

Функціональні вимоги визначають перелік можливостей, які повинна забезпечувати інформаційна система для виконання поставлених перед нею завдань. Формування функціональних вимог до розроблюваного веб-застосунку здійснювалося на основі аналізу предметної області, дослідження існуючих платформ продажу автомобілів та особливостей взаємодії між потенційними покупцями і продавцями. Для забезпечення належного розмежування доступу в системі передбачено три основні категорії користувачів: гість, зареєстрований користувач та адміністратор. Кожна з цих

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				33
Змн.	Арк.	№ докум.	Підпис	Дата		

категорій має відповідний набір функціональних можливостей залежно від рівня доступу до інформації та операцій системи [10].

Гість є користувачем, який не пройшов процедуру автентифікації в системі, тому його можливості обмежуються переглядом загальнодоступної інформації. Такий користувач може переглядати перелік доступних автомобілів та детальну інформацію про кожен транспортний засіб, включаючи його характеристики, фотографії та відомості про продавця. Для пошуку необхідного автомобіля гість може використовувати відповідні засоби пошуку за назвою, маркою або моделлю, а також застосовувати фільтрацію за такими параметрами, як марка, місто, ціна, рік випуску, пробіг, тип пального та коробка передач. Водночас гість не має можливості створювати або редагувати оголошення, залишати відгуки, обмінюватися повідомленнями з іншими користувачами чи надсилати заявки на придбання автомобілів. Для використання зазначених функцій необхідно пройти процедуру реєстрації та авторизації.

Після успішної реєстрації та входу до системи користувач отримує розширений набір функціональних можливостей. Авторизований користувач може переглядати інформацію про автомобілі, використовувати пошук і фільтрацію, а також створювати власні оголошення про продаж транспортних засобів. Для власних оголошень користувач має можливість редагувати інформацію про автомобіль, видаляти оголошення та завантажувати фотографії. Крім того, система дозволяє додавати цікаві автомобілі до списку обраних і переглядати сформований перелік, а також отримувати доступ до історії переглянутих транспортних засобів. Це забезпечує зручність повторного повернення до раніше переглянутих пропозицій.

Авторизований користувач також отримує можливість безпосередньої взаємодії з іншими учасниками системи. Він може надсилати повідомлення власникам автомобілів, отримувати повідомлення від інших користувачів та надсилати заявки на придбання транспортних засобів. Після відповідної взаємодії користувач може залишати відгуки про продавців відповідно до

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				34
Змн.	Арк.	№ докум.	Підпис	Дата		

передбачених системою правил. Крім того, він має можливість переглядати власні оголошення та контролювати їх поточний статус. Таким чином, зареєстрований користувач може виступати як потенційним покупцем, так і продавцем автомобіля в межах одного облікового запису.

Якщо зареєстрований користувач створив оголошення про продаж транспортного засобу, він додатково отримує функціональні можливості власника автомобіля. Зокрема, власник може переглядати заявки, отримані від потенційних покупців, приймати рішення щодо їх підтвердження або відхилення, а також переглядати листування, пов'язане з конкретним автомобілем. Для забезпечення взаємодії з потенційними покупцями власник може вести з ними обмін повідомленнями та обговорювати умови придбання транспортного засобу. Після успішного продажу автомобіля власник також може змінити відповідний статус оголошення. Такий механізм дозволяє організувати основні етапи взаємодії між продавцем і покупцем без використання сторонніх сервісів.

Окремий рівень функціональних можливостей передбачено для адміністратора системи. Адміністратор має розширений доступ до інформаційної системи та виконує функції контролю її роботи й інформаційного наповнення. Він може переглядати всі розміщені оголошення незалежно від їхніх власників, а також отримувати доступ до списку зареєстрованих користувачів. У разі виявлення порушень правил користування системою адміністратор має можливість видаляти відповідні оголошення або облікові записи користувачів. Також він здійснює загальний контроль інформаційного наповнення веб-застосунку, що дозволяє підтримувати актуальність і належний стан даних.

Таким чином, реалізація визначених функціональних вимог забезпечує розмежування можливостей між різними категоріями користувачів та дозволяє організувати повний цикл взаємодії з веб-застосунком. Гість отримує доступ до перегляду та пошуку автомобілів, зареєстрований користувач — до функцій купівлі, продажу та комунікації, а адміністратор — до засобів

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				35
Змн.	Арк.	№ докум.	Підпис	Дата		

контролю й керування системою. Такий розподіл функціональних можливостей забезпечує відповідність програмного продукту поставленим вимогам та створює основу для безпечної й зручної роботи користувачів із веб-застосунком.

На основі проведеного аналізу можна зробити висновок, що функціональні можливості системи охоплюють усі основні процеси, пов'язані з купівлею та продажем автомобілів. Реалізований функціонал забезпечує не лише публікацію оголошень, а й повноцінну взаємодію між користувачами шляхом обміну повідомленнями, надсилання заявок на придбання автомобілів, ведення історії переглядів, формування списку обраних автомобілів та залишення відгуків про продавців.

Розмежування функціональних можливостей між різними категоріями користувачів забезпечує належний рівень безпеки, запобігає несанкціонованому доступу до окремих функцій системи та створює зручні умови для роботи кожної категорії користувачів відповідно до їхніх повноважень.

Окрім функціональних можливостей, інформаційна система повинна відповідати визначеним нефункціональним вимогам, які характеризують якість її роботи, продуктивність, надійність, безпеку, масштабованість та зручність використання. Якщо функціональні вимоги визначають перелік операцій, які повинна виконувати система, то нефункціональні вимоги встановлюють характеристики та умови, яких необхідно дотримуватися під час експлуатації програмного продукту [5].

Під час проєктування веб-застосунку продажу вживаних автомобілів було визначено основні нефункціональні вимоги, дотримання яких забезпечує стабільну роботу системи та комфортну взаємодію з нею для всіх категорій користувачів.

Для коректного функціонування розробленого веб-застосунку необхідний певний набір апаратних і програмних ресурсів.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				36
Змн.	Арк.	№ докум.	Підпис	Дата		

Серверна частина системи повинна функціонувати на комп'ютері з процесором із тактовою частотою від 2 ГГц, оперативною пам'яттю обсягом не менше 4 ГБ та щонайменше 10 ГБ вільного дискового простору. Як операційна система може використовуватися Windows із попередньо встановленими середовищем Node.js та системою управління базами даних MySQL.

Для роботи клієнтської частини достатньо персонального комп'ютера або мобільного пристрою із сучасним веб-браузером, доступом до мережі Інтернет та обсягом оперативної пам'яті не менше 2 ГБ. Зазначені вимоги є достатніми для забезпечення роботи основних функцій розробленого веб-застосунку та не потребують використання високопродуктивного обладнання.

Веб-застосунок повинен забезпечувати швидку обробку запитів користувачів незалежно від кількості зареєстрованих користувачів та розміщених оголошень.

Основні операції, зокрема перегляд інформації, пошук автомобілів, фільтрація та робота з оголошеннями, повинні виконуватися без помітних для користувача затримок. Особлива увага приділяється оптимізації операцій пошуку автомобілів, фільтрації та отримання інформації з бази даних. Для підвищення швидкодії в базі даних передбачено використання індексів для полів, які найчастіше беруть участь у пошуку та сортуванні, зокрема для марки автомобіля, міста продажу, вартості автомобіля, статусу заявки та параметрів листування між користувачами. Це дозволяє зменшити час виконання запитів та підвищити загальну продуктивність системи.

Веб-застосунок повинен забезпечувати стабільну роботу під час одночасного використання декількома користувачами та гарантувати цілісність інформації, що зберігається в системі. Для цього в структурі бази даних використовуються зовнішні ключі, які встановлюють зв'язки між відповідними таблицями та запобігають появі неузгоджених записів. Додатково застосовується механізм каскадного видалення записів ON DELETE CASCADE, завдяки якому пов'язані дані автоматично видаляються

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				37
Змн.	Арк.	№ докум.	Підпис	Дата		

після видалення відповідного основного об'єкта. Це дозволяє уникнути накопичення некоректних або непотрібних записів у базі даних. Наявність часових міток створення та оновлення записів також забезпечує можливість контролю актуальності інформації та відстеження основних змін у системі.

Однією з основних нефункціональних вимог до інформаційної системи є забезпечення безпеки персональних даних користувачів та захисту їхніх облікових записів [36]. Для реалізації автентифікації використовується технологія JWT (JSON Web Token), яка забезпечує ідентифікацію авторизованого користувача під час звернення до захищених ресурсів системи. Паролі користувачів не зберігаються у відкритому вигляді. Перед записом до бази даних вони проходять хешування за допомогою бібліотеки bcrypt, що підвищує рівень захисту облікових записів у разі несанкціонованого доступу до даних [37].

Додатковим механізмом забезпечення безпеки є розмежування прав доступу відповідно до ролі користувача. Гість має можливість лише переглядати доступну інформацію, зареєстрований користувач отримує доступ до персонального функціоналу, пов'язаного з придбанням і продажем автомобілів, а адміністратор має розширені права для керування інформаційним наповненням системи. Такий підхід дозволяє обмежити доступ до операцій відповідно до повноважень користувача та зменшити ризик несанкціонованої зміни даних.

Архітектура веб-застосунку повинна забезпечувати можливість його подальшого розвитку без необхідності повного перепроєктування програмного продукту. Використання окремих модулів серверної частини, логічне розділення функцій та реляційна структура бази даних створюють можливість для поступового розширення функціональності системи. У подальшому без значних змін до основної архітектури можуть бути реалізовані додаткові можливості, зокрема система сповіщень про нові оголошення, онлайн-оплата або рекомендаційний механізм підбору

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				38
Змн.	Арк.	№ докум.	Підпис	Дата		

автомобілів. Таким чином, розроблена структура дозволяє адаптувати веб-застосунок до зміни вимог і збільшення кількості користувачів.

Інтерфейс веб-застосунку повинен бути інтуїтивно зрозумілим та зручним для користувачів із різним рівнем підготовки. Навігація між основними сторінками повинна здійснюватися за мінімальну кількість переходів, а найбільш важливі функції мають бути доступними без тривалого пошуку відповідних елементів керування. Особливу увагу під час розробки приділено оформленню карток автомобілів, форм створення та редагування оголошень, сторінки листування та особистого кабінету користувача. Зручний інтерфейс повинен сприяти швидкому виконанню основних операцій та забезпечувати зрозуміле представлення інформації.

Програмний продукт повинен коректно працювати в сучасних веб-браузерах, які підтримують стандарти HTML5, CSS3 та JavaScript. Оскільки розроблений застосунок є веб-орієнтованим, для його використання не потрібно встановлювати спеціальне програмне забезпечення на клієнтський пристрій. Користувачеві достатньо мати сучасний веб-браузер та доступ до мережі Інтернет, що забезпечує можливість використання системи на різних типах пристроїв.

Однією з важливих характеристик веб-застосунку є швидкість реакції на дії користувача. Висока продуктивність системи безпосередньо впливає на зручність її використання та загальне сприйняття програмного продукту. Тому під час проєктування було визначено орієнтовні часові характеристики виконання основних операцій, які повинні забезпечувати комфортну взаємодію користувача із системою.

До таких операцій належать відкриття головної сторінки, авторизація та реєстрація користувача, створення посилання для відновлення пароля, зміна пароля за одноразовим токеном, виконання розширеного пошуку та фільтрації, сортування оголошень, відкриття сторінки автомобіля, перегляд і збільшення фотографій, додавання та редагування автомобіля, видалення окремої фотографії, додавання автомобіля до обраного, надсилання повідомлень,

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				39
Змн.	Арк.	№ докум.	Підпис	Дата		

завантаження діалогу, надсилання заявки на придбання, підтвердження або відхилення заявки, залишення відгуку та завантаження адміністративної панелі. Максимально допустимий час виконання зазначених операцій визначено відповідно до їх складності та наведено в таблиці (табл. 2.1).

Таблиця 2.1

Часові характеристики виконання основних операцій системи

Операція	Максимальний час виконання
Відкриття головної сторінки	до 3 с
Авторизація користувача	до 2 с
Реєстрація нового користувача	до 3 с
Створення посилання для відновлення пароля	до 2 с
Зміна пароля за одноразовим токеном	до 2 с
Розширений пошук і фільтрація оголошень	до 2 с
Сортування оголошень	до 2 с
Відкриття сторінки автомобіля	до 2 с
Перегляд галереї та збільшення фото	до 1 с
Додавання нового автомобіля	до 3 с
Редагування автомобіля із записом історії ціни	до 3 с
Видалення окремої фотографії оголошення	до 2 с
Додавання автомобіля до обраних	до 1 с
Надсилання повідомлення	до 1 с
Завантаження діалогу з продавцем або покупцем	до 2 с
Надсилання заявки на придбання	до 2 с
Підтвердження або відхилення заявки	до 2 с
Залишення відгуку	до 2 с
Завантаження адміністративної панелі	до 2 с

Наведені в таблиці 2.1 часові характеристики є орієнтовними та можуть залежати від швидкості мережевого з'єднання, поточного навантаження на сервер, обсягу даних і продуктивності обладнання користувача. Водночас використання оптимізованих SQL-запитів, індексації бази даних та відповідної організації серверної частини дозволяє забезпечити виконання більшості основних операцій у межах встановлених часових значень. Це сприяє забезпеченню достатньої швидкодії системи та комфортної взаємодії користувачів із веб-застосунком.

Після визначення функціональних та нефункціональних вимог до веб-застосунку було сформовано основні варіанти використання системи, які описують взаємодію користувачів із програмним продуктом. Варіанти

використання дають змогу визначити основні сценарії роботи системи, встановити взаємозв'язок між користувачами та функціональними можливостями, а також уточнити порядок виконання основних операцій.

У розробленому веб-застосунку передбачено чотири типи учасників взаємодії: гість, зареєстрований користувач, власник оголошення та адміністратор. Гість має доступ до загальнодоступних функцій системи, зокрема реєстрації, авторизації, відновлення пароля, перегляду оголошень та використання пошуку і фільтрації. Після проходження авторизації користувач отримує доступ до розширених можливостей, пов'язаних зі створенням та керуванням власними оголошеннями, обраними автомобілями, заявками, повідомленнями та відгуками. Власник оголошення є зареєстрованим користувачем, який розмістив автомобіль у системі, тому він додатково отримує можливість керувати отриманими заявками, фотографіями та інформацією про власні оголошення. Адміністратор має розширені права та здійснює контроль користувачів і розміщених у системі автомобілів.

На рисунку 2.1 наведено діаграму варіантів використання веб-застосунку продажу вживаних автомобілів. Діаграма відображає основні функціональні можливості системи та взаємозв'язки між ними і відповідними категоріями користувачів.



Рисунок 2.1 – Діаграма варіантів використання веб-застосунку продажу вживаних автомобілів

Представлена діаграма відображає основні функціональні можливості розробленої системи та взаємозв'язки між ними і відповідними акторами. Як видно з рисунка 2.1, функціональність системи охоплює процеси реєстрації та авторизації користувачів, перегляду і пошуку оголошень, створення та

редагування інформації про автомобілі, роботи із заявками на придбання, обміну повідомленнями та відгуками. Окремо виділено функції власника оголошення та адміністратора, які забезпечують керування відповідними даними системи.

Для більш детального опису основних сценаріїв взаємодії користувачів із системою нижче наведено окремі прецеденти. Для кожного прецеденту визначено його назву, відповідного актора та короткий опис процесу виконання.

Опис прецеденту «Реєстрація користувача». Одним із базових сценаріїв роботи із системою є реєстрація нового користувача. Даний прецедент передбачає створення облікового запису та введення користувачем необхідної інформації. Основні характеристики прецеденту «Реєстрація користувача» наведено в таблиці (табл. 2.2).

Таблиця 2.2

Опис прецеденту «Реєстрація користувача»

Назва прецеденту	Реєстрація користувача
Актори	Гість
Короткий опис	Користувач створює новий обліковий запис шляхом введення особистих даних, електронної пошти та пароля. Після успішної перевірки введених даних інформація зберігається у базі даних, а користувач отримує можливість авторизуватися в системі.

Таким чином, виконання даного прецеденту забезпечує створення нового облікового запису та надає користувачу можливість перейти до наступного етапу взаємодії із системою - авторизації.

Опис прецеденту «Авторизація користувача». Після успішної реєстрації користувач може виконати вхід до системи. Авторизація необхідна для отримання доступу до функціональних можливостей, які доступні лише зареєстрованим користувачам. Опис основних характеристик прецеденту «Авторизація користувача» наведено в таблиці (табл. 2.3).

Опис прецеденту «Авторизація користувача»

Назва прецеденту	Авторизація користувача
Актори	Гість
Короткий опис	Користувач вводить адресу електронної пошти та пароль. Після перевірки облікових даних система створює JWT-токен автентифікації та надає доступ до функцій зареєстрованого користувача.

У результаті виконання прецеденту користувач проходить автентифікацію та отримує доступ до функцій відповідно до своєї ролі. Використання JWT-токена забезпечує подальшу ідентифікацію користувача під час роботи із захищеними ресурсами системи.

Опис прецеденту «Додавання автомобіля». Важливою функціональною можливістю веб-застосунку є створення користувачем власного оголошення про продаж автомобіля. Для цього авторизований користувач заповнює відповідну форму та вказує основну інформацію про транспортний засіб. Опис прецеденту «Додавання автомобіля» наведено в таблиці (табл. 2.4).

Опис прецеденту «Додавання автомобіля»

Назва прецеденту	Додавання автомобіля
Актори	Зареєстрований користувач
Короткий опис	Користувач заповнює форму створення оголошення, зазначає технічні характеристики автомобіля, його опис, ціну, місце знаходження та завантажує фотографії. Після підтвердження інформація зберігається у базі даних і оголошення стає доступним для перегляду іншими користувачами.

Таким чином, даний прецедент забезпечує основний механізм розміщення пропозицій про продаж автомобілів у системі.

Опис прецеденту «Редагування інформації про авто». Після створення оголошення власник автомобіля може змінювати його інформаційне наповнення відповідно до актуального стану транспортного засобу або умов продажу. Основні характеристики прецеденту «Редагування автомобіля» наведено в таблиці (табл. 2.5).

Таблиця 2.5

Опис прецеденту «Редагування інформації про авто»

Назва прецеденту	Редагування автомобіля
Актори	Зареєстрований користувач (Власник автомобіля)
Короткий опис	Власник оголошення може змінювати інформацію про автомобіль, оновлювати фотографії, змінювати ціну, опис та інші характеристики. Після збереження змін інформація автоматично оновлюється у базі даних.

Отже, прецедент забезпечує підтримання актуальності інформації про автомобіль та дозволяє власнику оперативно змінювати параметри власного оголошення.

Опис прецеденту «Надсилання заявки на придбання автомобіля». Для організації процесу купівлі-продажу зареєстрований користувач має можливість сформулювати заявку на придбання обраного автомобіля. Опис відповідного прецеденту наведено в таблиці (табл. 2.6).

Таблиця 2.6

Опис прецеденту «Надсилання заявки на придбання автомобіля»

Назва прецеденту	Надсилання заявки на придбання автомобіля
Актори	Зареєстрований користувач
Короткий опис	Користувач формує заявку на придбання обраного автомобіля. Після підтвердження запит зберігається у базі даних зі статусом «Очікує розгляду» та стає доступним власнику автомобіля.

Виконання даного прецеденту дозволяє формалізувати намір користувача придбати автомобіль та передати заявку його власнику для подальшого розгляду.

Опис прецеденту «Підтвердження або відхилення заявки». Отримані заявки повинні бути доступними власнику відповідного автомобіля для подальшого розгляду. Власник може прийняти заявку або відхилити її залежно від результату взаємодії з потенційним покупцем. Основні характеристики прецеденту наведено в таблиці (табл. 2.7).

Таблиця 2.7

Опис прецеденту «Підтвердження або відхилення заявки»

Назва прецеденту	Підтвердження або відхилення заявки
Актори	Зареєстрований користувач (Власник автомобіля)
Короткий опис	Власник автомобіля переглядає отримані заявки та приймає рішення щодо їх підтвердження або відхилення. Після виконання операції статус заявки змінюється у базі даних, а покупець отримує актуальну інформацію про результат розгляду.

Таким чином, даний прецедент забезпечує можливість власнику контролювати отримані заявки та визначати подальший статус взаємодії з потенційним покупцем.

Опис прецеденту «Обмін повідомленнями». Для забезпечення безпосередньої комунікації між продавцями та потенційними покупцями у веб-застосунку передбачено систему обміну повідомленнями. Опис прецеденту «Обмін повідомленнями» наведено в таблиці (табл. 2.8).

Таблиця 2.8

Опис прецеденту «Обмін повідомленнями»

Назва прецеденту	Обмін повідомленнями
Актори	Зареєстрований користувач, Зареєстрований користувач (Власник автомобіля)
Короткий опис	Користувачі можуть надсилати один одному повідомлення щодо конкретного автомобіля. Усі повідомлення зберігаються у базі даних із зазначенням відправника, отримувача, автомобіля та часу відправлення, що забезпечує ведення історії листування.

Збереження повідомлень у базі даних дозволяє підтримувати історію листування та пов'язувати комунікацію з конкретним оголошенням про продаж автомобіля.

Опис прецеденту «Залишення відгуку». Додатковим засобом взаємодії користувачів є система відгуків, яка дозволяє оцінювати результати взаємодії між покупцями та продавцями. Основні характеристики прецеденту «Залишення відгуку» наведено в таблиці (табл. 2.9).

Таблиця 2.9

Опис прецеденту «Залишення відгуку»

Назва прецеденту	Залишення відгуку
Актори	Зареєстрований користувач
Короткий опис	Після взаємодії з продавцем користувач може залишити текстовий відгук та оцінку за п'ятибальною шкалою. Відгук зберігається у базі даних і використовується для формування рейтингу продавця.

Отже, даний прецедент дозволяє накопичувати оцінки та відгуки користувачів, які можуть використовуватися для формування репутації продавців у системі.

Опис прецеденту «Керування користувачами». Адміністративна частина веб-застосунку передбачає можливість контролю зареєстрованих користувачів. Адміністратор може переглядати інформацію про користувачів та видаляти облікові записи у випадку порушення встановлених правил. Опис прецеденту наведено в таблиці (табл. 2.10).

Таблиця 2.10

Опис прецеденту «Керування користувачами»

Назва прецеденту	Керування користувачами
Актори	Адміністратор
Короткий опис	Адміністратор переглядає список зареєстрованих користувачів та, за необхідності, може видалити обліковий запис користувача із системи. Під час видалення також автоматично видаляються пов'язані записи відповідно до налаштованих зовнішніх ключів бази даних.

Застосування такого механізму дозволяє адміністратору контролювати облікові записи та підтримувати належний стан інформаційного наповнення системи.

Опис прецеденту «Керування автомобілями». Окремою адміністративною функцією є контроль оголошень про продаж автомобілів. Адміністратор має можливість переглядати розміщені оголошення та видаляти ті з них, які порушують правила використання системи або містять некоректну інформацію. Основні характеристики даного прецеденту наведено в таблиці (табл. 2.11).

Опис прецеденту «Керування автомобілями»

Назва прецеденту	Керування автомобілями
Актори	Адміністратор
Короткий опис	Адміністратор переглядає всі оголошення, розміщені у системі, та має можливість видаляти автомобілі, що порушують правила користування сервісом або містять некоректну інформацію.

Таким чином, адміністративний контроль дозволяє підтримувати актуальність оголошень та забезпечувати відповідність інформаційного наповнення встановленим правилам.

У результаті сформовані варіанти використання охоплюють основні сценарії роботи веб-застосунку та відображають взаємодію між його користувачами і функціональними можливостями. Наведені в таблицях 2.2 - 2.11 прецеденти деталізують основні операції системи, починаючи від створення та авторизації облікового запису і завершуючи процесами купівлі-продажу, обміну повідомленнями та адміністративного контролю. Це дозволяє формалізувати функціональні вимоги до системи та визначити основні сценарії, які повинні бути реалізовані під час розробки програмного продукту.

Об'єктно-орієнтована структура системи. Під час проектування веб-застосунку було застосовано об'єктно-орієнтований підхід, відповідно до якого система представлена як сукупність взаємопов'язаних сутностей предметної області. Кожна сутність характеризується власними властивостями та взаємозв'язками з іншими компонентами системи. Такий підхід забезпечує логічне структурування програмного забезпечення, спрощує його супровід та створює можливість для подальшого розширення функціональності.

Основу об'єктно-орієнтованої моделі становлять десять основних сутностей: User, Car, CarImage, Favorite, Message, Review, Order, ViewHistory, CarPriceHistory та PasswordResetToken. Їх структура та взаємозв'язки відповідають основним сутностям реляційної бази даних, що

використовується у веб-застосунку. На рисунку наведено діаграму класів розробленої системи (рис 2.2).

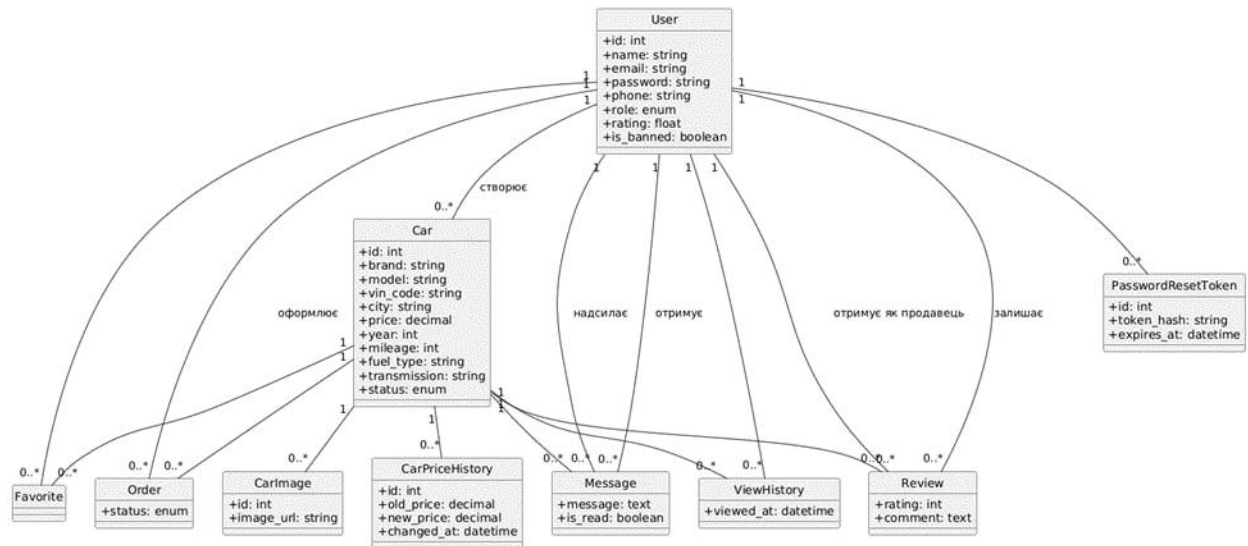


Рисунок 2.2 – Діаграма класів веб-застосунку продажу вживаних автомобілів

Діаграма класів відображає основні сутності системи, їх атрибути та зв'язки між ними. Центральним класом є User, який представляє користувача системи та містить основну інформацію про нього, зокрема ім'я, електронну пошту, пароль, номер телефону, роль, рейтинг та статус облікового запису. Один користувач може створювати декілька оголошень, залишати відгуки, надсилати та отримувати повідомлення, формувати заявки на придбання автомобілів, а також додавати транспортні засоби до списку обраного.

Клас Car відповідає за представлення автомобіля та містить основні характеристики оголошення: марку, модель, VIN-код, місто, ціну, рік випуску, пробіг, тип пального, трансмісію, опис і статус. З кожним автомобілем може бути пов'язано декілька фотографій, записів історії зміни ціни, повідомлень, заявок та переглядів. Для зберігання фотографій використовується клас CarImage, який забезпечує можливість додавання декількох зображень до одного автомобіля.

Клас Favorite реалізує механізм додавання автомобілів до списку обраного, що дозволяє користувачеві швидко отримувати доступ до оголошень, які його зацікавили.

Клас Message призначений для реалізації обміну повідомленнями між користувачами. Він містить текст повідомлення, інформацію про його відправника та отримувача, а також ознаку прочитання. Повідомлення пов'язуються з конкретним автомобілем, що дозволяє організувати листування безпосередньо в контексті відповідного оголошення.

Клас Review використовується для реалізації системи відгуків та оцінювання продавців. Він містить оцінку та текстовий коментар користувача, що можуть використовуватися для формування загального рейтингу продавця.

Клас Order відповідає за заявки на придбання автомобілів. Він пов'язує покупця з конкретним автомобілем та містить поточний статус заявки. Власник оголошення може переглядати отримані заявки та змінювати їх статус шляхом підтвердження або відхилення.

Для реалізації додаткового функціоналу використовуються класи ViewHistory, CarPriceHistory та PasswordResetToken. ViewHistory забезпечує збереження інформації про переглянуті користувачем автомобілі. CarPriceHistory використовується для фіксації змін вартості автомобіля, зберігаючи попередню та нову ціну, а також дату зміни. PasswordResetToken забезпечує безпечне відновлення пароля за допомогою одноразового токена з обмеженим строком дії.

Взаємозв'язки між класами відображають логіку функціонування системи. Один користувач може створювати декілька автомобілів, кожен автомобіль може мати декілька фотографій та записів історії ціни, а користувачі можуть формувати заявки, залишати відгуки, надсилати повідомлення та зберігати автомобілі в обраному. Така структура дозволяє забезпечити цілісність даних і логічно пов'язати між собою основні процеси веб-застосунку.

Отже, розроблена об'єктно-орієнтована модель відображає основні сутності предметної області та забезпечує основу для реалізації функціональності веб-застосунку. Визначені класи та зв'язки між ними узгоджуються зі структурою бази даних і сформованими функціональними

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				50
Змн.	Арк.	№ докум.	Підпис	Дата		

вимогами, що спрощує подальшу реалізацію та супровід програмного продукту.

Проведений аналіз вимог дозволив визначити основні функціональні можливості системи, ролі користувачів та сценарії їх взаємодії. На основі цих вимог було сформовано діаграму варіантів використання та об'єктно-орієнтовану модель системи. Розроблена модель визначає основні сутності, їх властивості та взаємозв'язки і використовується як основа для подальшого проєктування бази даних та реалізації програмного комплексу.

2.2. Розробка бази даних системи

База даних є одним із основних компонентів розробленого веб-застосунку, оскільки забезпечує зберігання інформації про користувачів, автомобілі, заявки на придбання, повідомлення, відгуки, історію переглядів та інші дані, необхідні для роботи системи. Від структури бази даних залежить цілісність інформації, швидкодія програмного забезпечення та можливість подальшого розширення його функціональних можливостей.

Для реалізації сховища даних використано реляційну систему керування базами даних MySQL. Її вибір зумовлений високою продуктивністю, підтримкою зовнішніх ключів, індексуванням, забезпеченням цілісності даних та сумісністю із середовищем Node.js. Структуру бази даних сформовано відповідно до основних процесів предметної області. Вона повинна забезпечувати зберігання інформації про користувачів, оголошення автомобілів, фотографії, заявки на придбання, повідомлення між продавцями та покупцями, відгуки, обрані автомобілі, історію переглядів, зміни ціни та відновлення пароля.

У результаті проєктування створено реляційну базу даних, що складається з десяти основних таблиць, пов'язаних між собою за допомогою зовнішніх ключів. Така структура дозволяє уникнути дублювання інформації та забезпечити цілісність даних. На рисунку наведено ER-діаграму розробленої бази даних (рис. 2.3).

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		51

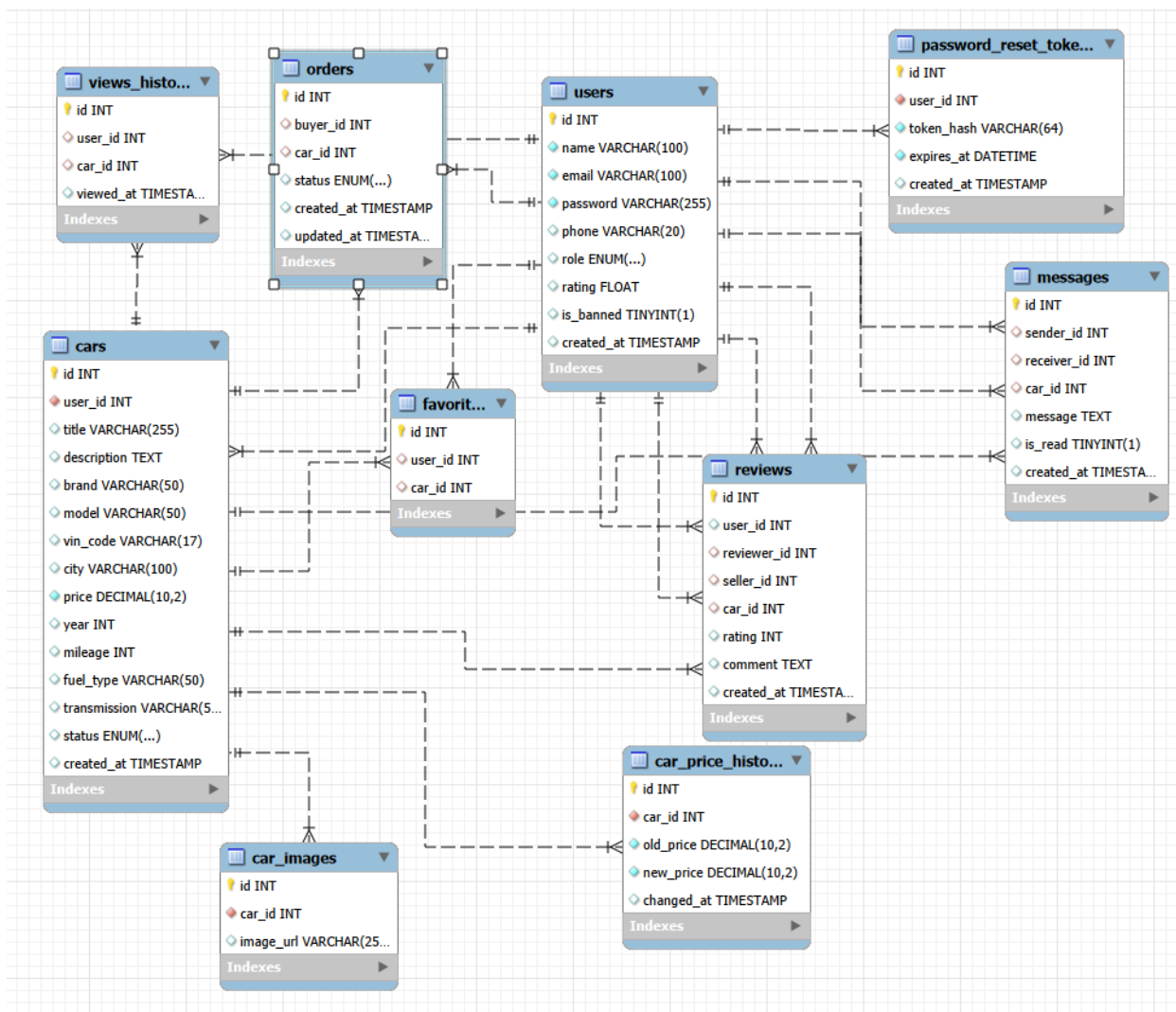


Рисунок 2.3 – ER-діаграма бази даних веб-застосунку продажу автомобілів

До складу бази даних входять таблиці users, cars, car_images, favorites, messages, reviews, orders, views_history, car_price_history та password_reset_tokens. Кожна таблиця відповідає окремій функціональній частині системи та пов'язана з іншими таблицями відповідно до логіки предметної області.

Таблиця users призначена для зберігання інформації про зареєстрованих користувачів системи. Вона є центральною таблицею бази даних, оскільки інші сутності містять зовнішні ключі, пов'язані з користувачами. У ній зберігаються персональні дані, електронна пошта, хешований пароль, номер телефону, роль, рейтинг, ознака блокування та дата реєстрації. Поле id є

первинним ключем, а для email встановлено обмеження унікальності. Структуру таблиці users наведено в таблиці (табл. 2.12).

Таблиця 2.12

Структура таблиці users

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор користувача
name	VARCHAR(100)	Ім'я користувача
email	VARCHAR(100)	Адреса електронної пошти користувача
password	VARCHAR(255)	Хешований пароль користувача
phone	VARCHAR(20)	Контактний номер телефону
role	ENUM('user','admin')	Роль користувача в системі
rating	FLOAT	Середній рейтинг продавця
is_banned	BOOLEAN	Ознака блокування облікового запису
created_at	TIMESTAMP	Дата та час реєстрації користувача

Таблиця cars зберігає основну інформацію про автомобілі, представлені на продаж. Для кожного оголошення передбачено збереження його назви, опису, марки, моделі, VIN-коду, міста, вартості, року випуску, пробігу, типу пального, трансмісії та поточного статусу.

Зв'язок автомобіля з його власником реалізовано за допомогою поля user_id, яке посилається на таблицю users. Поле vin_code використовується для контролю унікальності транспортного засобу. Структуру таблиці cars наведено в таблиці (табл. 2.13).

Таблиця 2.13

Структура таблиці cars

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор автомобіля
user_id	INT	Ідентифікатор власника автомобіля
title	VARCHAR(255)	Заголовок оголошення
description	TEXT	Опис автомобіля
brand	VARCHAR(50)	Марка автомобіля
model	VARCHAR(50)	Модель автомобіля
vin_code	VARCHAR(17)	VIN-код автомобіля
city	VARCHAR(100)	Місто продажу автомобіля
price	DECIMAL(10,2)	Вартість автомобіля
year	INT	Рік випуску

Продовження таблиці 2.13

mileage	INT	Пробіг автомобіля
fuel_type	VARCHAR(50)	Тип пального
transmission	VARCHAR(50)	Тип коробки передач
status	ENUM('active','sold')	Поточний статус оголошення
created_at	TIMESTAMP	Дата створення оголошення

Для зберігання фотографій автомобілів використовується окрема таблиця `car_images`. Вона містить ідентифікатор автомобіля та шлях до відповідного зображення, що дозволяє додавати декілька фотографій до одного оголошення без дублювання основної інформації. Структуру таблиці `car_images` наведено в таблиці (табл. 2.14).

Таблиця 2.14

Структура таблиці `car_images`

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор зображення
car_id	INT	Ідентифікатор автомобіля
image_url	VARCHAR(255)	Шлях до зображення автомобіля

Таблиця `favorites` реалізує функцію додавання автомобілів до списку обраного. Вона пов'язує користувача з обраним автомобілем. Обмеження `UNIQUE(user_id, car_id)` запобігає повторному додаванню одного автомобіля одним користувачем. Структуру таблиці `favorites` наведено в таблиці (табл. 2.15).

Таблиця 2.15

Структура таблиці `favorites`

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор запису
user_id	INT	Ідентифікатор користувача
car_id	INT	Ідентифікатор автомобіля, доданого до обраного

Таблиця `messages` використовується для реалізації внутрішнього обміну повідомленнями між продавцями та потенційними покупцями. У ній зберігаються дані про відправника, отримувача, автомобіль, текст повідомлення, його статус прочитання та дату надсилання. Прив'язка

повідомлення до конкретного автомобіля дозволяє вести окреме листування щодо кожного оголошення. Структуру таблиці messages наведено в таблиці (табл. 2.16).

Таблиця 2.16

Структура таблиці messages

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор повідомлення
sender_id	INT	Ідентифікатор відправника повідомлення
receiver_id	INT	Ідентифікатор отримувача повідомлення
car_id	INT	Ідентифікатор автомобіля, щодо якого ведеться листування
message	TEXT	Текст повідомлення
is_read	BOOLEAN	Ознака прочитання повідомлення
created_at	TIMESTAMP	Дата та час надсилання повідомлення

Таблиця reviews призначена для реалізації системи оцінювання продавців. Вона містить інформацію про автора відгуку, продавця, автомобіль, оцінку, текст коментаря та дату створення. Збережені дані використовуються для формування рейтингу продавця. Структуру таблиці reviews наведено в таблиці (табл. 2.17).

Таблиця 2.17

Структура таблиці reviews

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор відгуку
reviewer_id	INT	Ідентифікатор користувача, який залишив відгук
seller_id	INT	Ідентифікатор продавця
car_id	INT	Ідентифікатор автомобіля, до якого належить відгук
rating	INT	Оцінка продавця від 1 до 5
comment	TEXT	Текст відгуку
created_at	TIMESTAMP	Дата створення відгуку

Таблиця orders забезпечує зберігання заявок на придбання автомобілів. Кожна заявка пов'язана з покупцем і конкретним автомобілем та містить її поточний статус, дату створення й дату останнього оновлення. Статус заявки

може мати значення pending, approved або rejected. Структуру таблиці orders наведено в таблиці (табл. 2.18).

Таблиця 2.18

Структура таблиці orders

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор заявки
buyer_id	INT	Ідентифікатор покупця
car_id	INT	Ідентифікатор автомобіля
status	ENUM('pending','approved','rejected')	Статус заявки на придбання
created_at	TIMESTAMP	Дата створення заявки
updated_at	TIMESTAMP	Дата останнього оновлення заявки

Таблиця views_history призначена для збереження інформації про переглянуті користувачем автомобілі. Кожен запис містить посилання на користувача та автомобіль, а також час перегляду. Це забезпечує можливість реалізації історії переглядів. Структуру таблиці views_history наведено в таблиці (табл. 2.19).

Таблиця 2.19

Структура таблиці views_history

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор запису
user_id	INT	Ідентифікатор користувача
car_id	INT	Ідентифікатор переглянутого автомобіля
viewed_at	TIMESTAMP	Дата та час перегляду автомобіля

Таблиця car_price_history використовується для збереження історії зміни вартості автомобіля. Новий запис створюється під час редагування оголошення, якщо встановлена ціна відрізняється від попередньої. Структуру таблиці car_price_history наведено в таблиці (табл. 2.20).

Таблиця 2.20

Структура таблиці car_price_history

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор запису
car_id	INT	Ідентифікатор автомобіля
old_price	DECIMAL(10,2)	Попередня вартість автомобіля
new_price	DECIMAL(10,2)	Нова вартість автомобіля

Продовження таблиці 2.20

changed_at	TIMESTAMP	Дата та час зміни ціни
------------	-----------	------------------------

Таблиця password_reset_tokens забезпечує функціонування механізму відновлення пароля. Для підвищення безпеки в ній зберігається SHA-256-хеш одноразового токена, а також інформація про користувача та строк його дії. Структуру таблиці password_reset_tokens наведено в таблиці (табл. 2.21).

Таблиця 2.21

Структура таблиці password_reset_tokens

Поле	Тип даних	Призначення
id	INT	Унікальний ідентифікатор токена
user_id	INT	Ідентифікатор користувача
token_hash	VARCHAR(64)	Хеш одноразового токена відновлення
expires_at	DATETIME	Дата та час завершення дії токена
created_at	TIMESTAMP	Дата створення токена

Між таблицями бази даних встановлено зв'язки типу «один до багатьох», що забезпечують логічну цілісність інформації. Один користувач може створити декілька оголошень, тоді як кожне оголошення належить одному користувачу. Аналогічно, один автомобіль може мати декілька фотографій, повідомлень, заявок, записів історії переглядів та записів зміни ціни.

Для реалізації зв'язків використовуються зовнішні ключі та механізм ON DELETE CASCADE. У разі видалення користувача або автомобіля пов'язані записи видаляються автоматично, що дозволяє уникнути появи неузгоджених даних.

Для підвищення продуктивності бази даних створено індекси для полів, які часто використовуються під час пошуку, фільтрації та службових операцій. Зокрема, індексуються ціна, марка, місто та VIN-код автомобіля, статус заявок, історія зміни ціни, токени відновлення пароля та повідомлення користувачів. Основні індекси, використані в розробленій базі даних, наведено в таблиці (табл. 2.22).

Індекси бази даних

Індекс	Таблиця	Призначення
idx_cars_price	cars	Прискорення пошуку автомобілів за ціною
idx_cars_brand	cars	Прискорення пошуку за маркою автомобіля
idx_cars_city	cars	Прискорення пошуку за містом
idx_cars_vin_code	cars	Контроль унікальності VIN-коду
idx_car_price_history_car	car_price_history	Швидкий перегляд історії зміни ціни
idx_password_reset_tokens_user	password_reset_tokens	Пошук токенів відновлення за користувачем
idx_password_reset_tokens_expires	password_reset_tokens	Очищення та перевірка прострочених токенів
idx_orders_status	orders	Прискорення вибірки заявок за статусом
idx_messages_users	messages	Оптимізація пошуку повідомлень між користувачами

Отже, спроектована реляційна база даних забезпечує інформаційне супроводження основних процесів веб-застосунку продажу автомобілів. Її структура охоплює роботу з користувачами, автомобілями, фотографіями, заявками, повідомленнями, відгуками, обраними автомобілями, історією переглядів, зміною ціни та відновленням пароля. Використання зовнішніх ключів, каскадного видалення та індексів забезпечує цілісність даних і належну продуктивність системи та створює основу для подальшої реалізації програмного комплексу.

2.3 Проектування та реалізація алгоритмів роботи системи

Після розроблення структури бази даних наступним етапом проектування стало створення алгоритмів роботи основних функціональних модулів веб-застосунку. Алгоритми визначають послідовність виконання дій користувача та програмної системи під час реалізації основних бізнес-процесів, а також забезпечують взаємодію між клієнтською частиною, сервером і базою даних.

Під час проектування були визначені алгоритми, які забезпечують виконання найбільш важливих функцій системи, а саме авторизацію користувача, відновлення пароля, створення й редагування оголошень про продаж автомобілів, перевірку VIN-коду, збереження історії зміни ціни,

надсилання заявок на придбання транспортного засобу, обмін повідомленнями між користувачами та керування заявками власником автомобіля.

Для наочного представлення послідовності виконання операцій використовувалися UML-діаграми діяльності (Activity Diagram), які відображають логіку роботи системи та взаємодію її окремих компонентів [7;9].

Алгоритм авторизації користувача є одним із ключових етапів роботи інформаційної системи, оскільки саме після успішного входу користувач отримує доступ до персоналізованого функціоналу веб-застосунку.

Під час авторизації користувач вводить адресу електронної пошти та пароль. Сервер перевіряє наявність користувача у базі даних, після чого виконує перевірку правильності пароля за допомогою бібліотеки bcrypt. У разі успішної перевірки створюється JWT-токен, який використовується для подальшої автентифікації запитів.

У випадку помилки введення даних або відсутності користувача система повертає повідомлення про помилку без надання доступу до функціональних можливостей. На рисунку наведено алгоритм авторизації користувача (рис. 2.4).

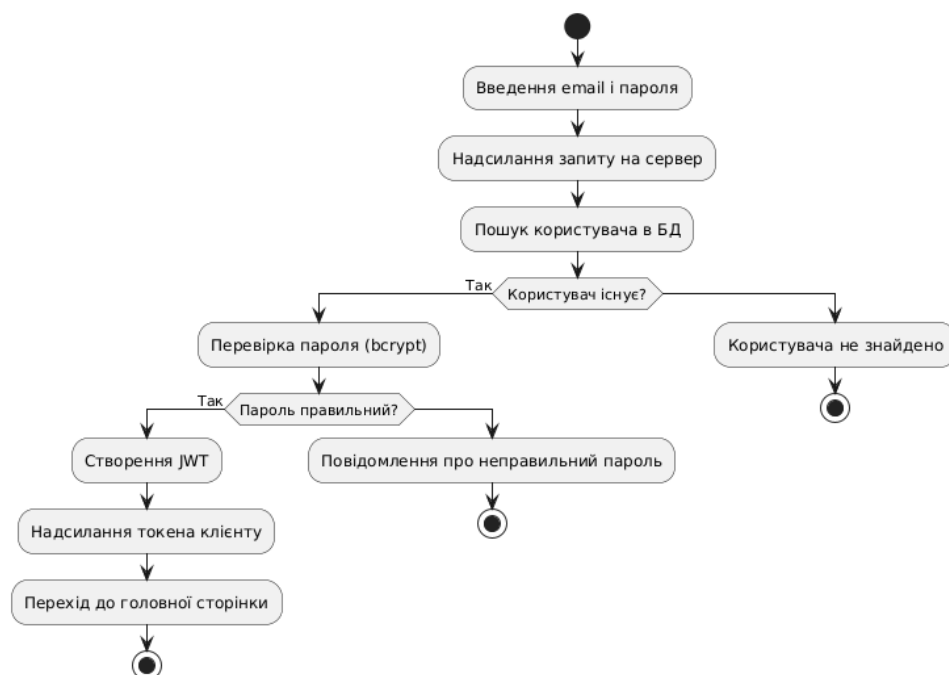


Рис 2.4 Алгоритм авторизації користувача

Після успішної авторизації всі наступні запити користувача супроводжуються JWT-токеном, що дозволяє серверу визначати особу користувача без повторного введення логіна та пароля.

Однією з основних функцій веб-застосунку є можливість створення оголошення про продаж автомобіля. Для додавання нового автомобіля користувач переходить до відповідної форми, заповнює інформацію про транспортний засіб, вказує його характеристики, додає опис, встановлює вартість та завантажує фотографії.

Після відправлення форми сервер виконує перевірку коректності отриманих даних. Якщо всі поля заповнені правильно, інформація записується до таблиці cars, а фотографії — до таблиці car_images. Після успішного завершення операції нове оголошення автоматично з'являється на головній сторінці веб-застосунку. На рисунку наведено алгоритм створення нового оголошення (рис. 2.5).

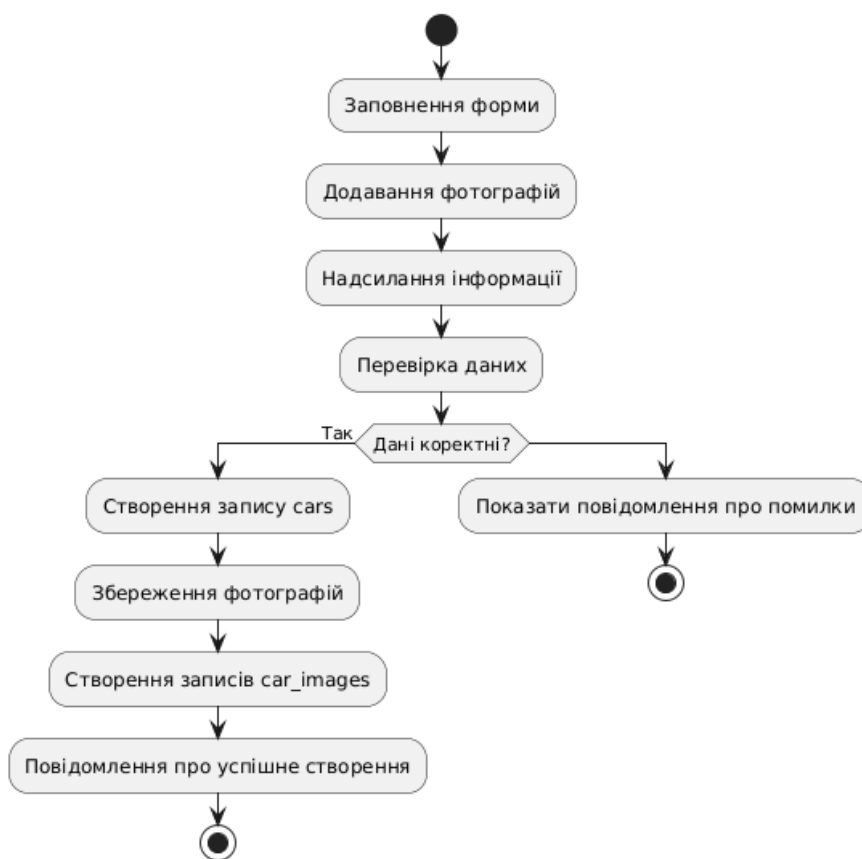


Рис 2.5 Алгоритм створення нового оголошення

Після успішного створення оголошення користувач отримує можливість його редагування, видалення або зміни статусу продажу.

Після перегляду інформації про автомобіль авторизований користувач може надіслати заявку на його придбання. Після натискання відповідної кнопки клієнтська частина формує HTTP-запит, який передається серверу. Сервер перевіряє існування автомобіля, наявність авторизації користувача та створює запис у таблиці orders зі статусом pending. Власник автомобіля може переглядати список отриманих заявок та приймати рішення щодо кожної з них. На рисунку наведено алгоритм створення заявки (рис 2.6).

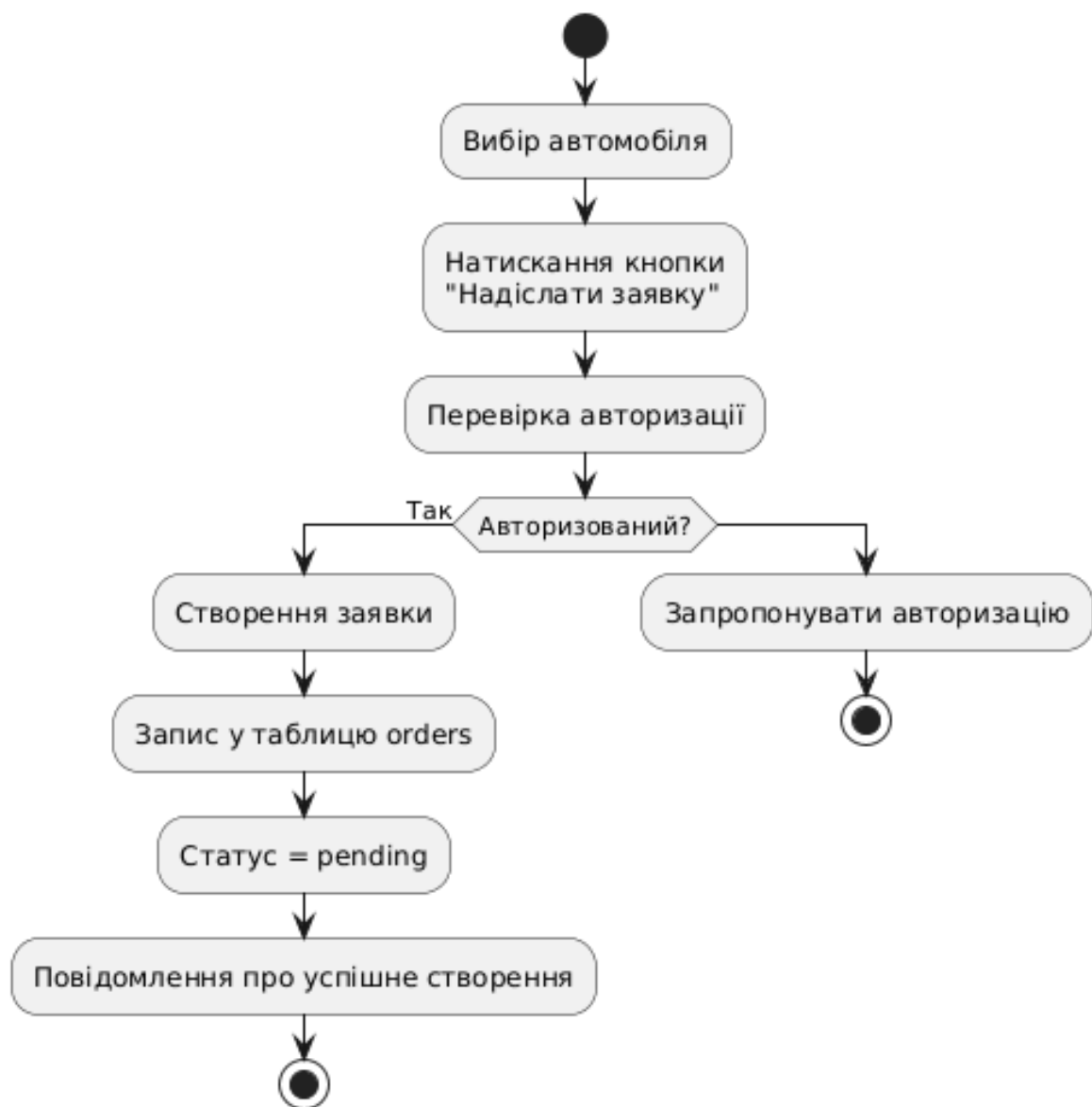


Рис 2.6 Алгоритм створення заявки

Для забезпечення безпосередньої взаємодії між продавцем і покупцем у системі реалізовано внутрішній чат. Після відкриття сторінки автомобіля користувач може перейти до листування з його власником. Введене повідомлення надсилається серверу, після чого перевіряється коректність даних та створюється новий запис у таблиці messages. Під час відкриття діалогу система автоматично отримує всі повідомлення, що належать відповідному автомобілю та конкретній парі користувачів. На рисунку наведено алгоритм обміну повідомленнями (рис. 2.7).



Рис 2.7 Алгоритм обміну повідомленнями

Після отримання заявки на придбання автомобіля власник транспортного засобу може переглянути список усіх отриманих заявок. Після вибору необхідної заявки користувач приймає рішення щодо її підтвердження або відхилення. Якщо заявку підтверджено, сервер змінює її статус на approved, переводить автомобіль у стан sold і автоматично відхиляє інші активні заявки на цей самий автомобіль. Якщо заявку відхилено, змінюється лише статус відповідного запису orders. На рисунку наведено алгоритм опрацювання заявки(рис. 2.8).

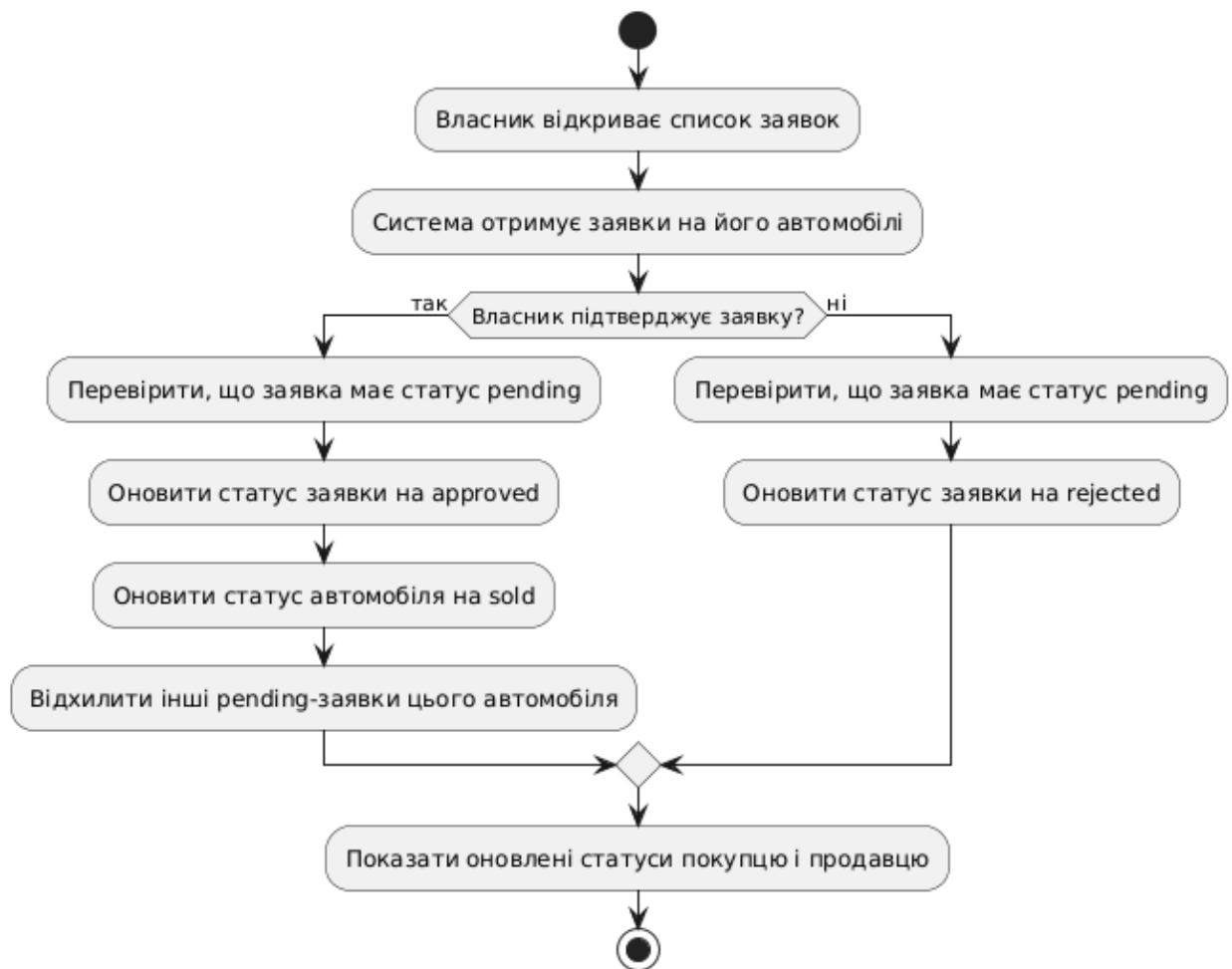


Рис 2.8 Алгоритм опрацювання заявки

Відновлення пароля починається із введення електронної пошти на сторінці forgot-password.html. Сервер перевіряє наявність користувача, генерує випадковий токен, зберігає його хеш у таблиці password_reset_tokens і повертає одноразове посилання на сторінку reset-password.html.

Під час встановлення нового пароля сервер перевіряє строк дії токена, хешує новий пароль за допомогою bcrypt, оновлює запис користувача та видаляє використаний токен. На рисунку наведено алгоритм відновлення пароля (рис. 2.9).

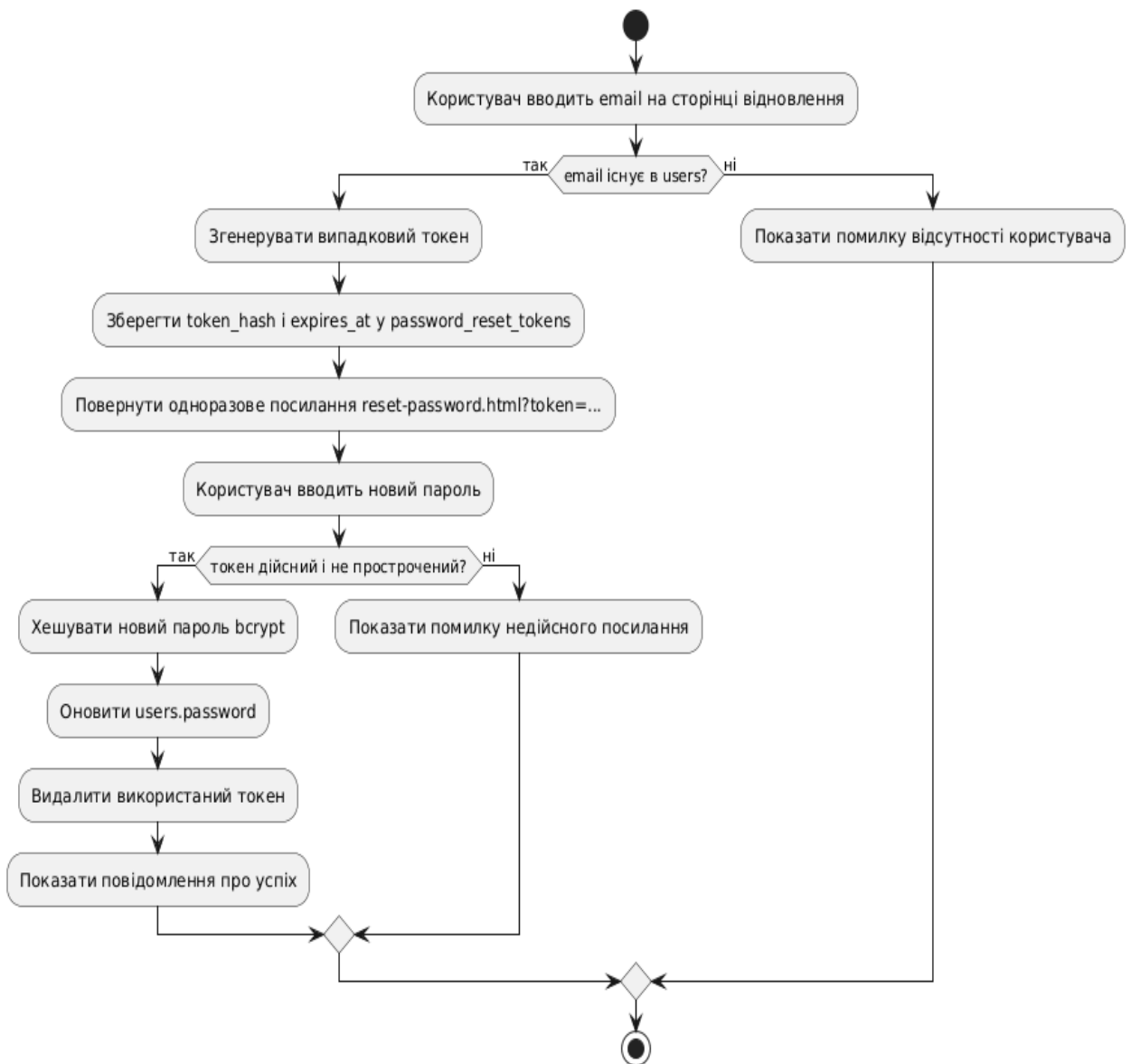


Рисунок 2.9 - Алгоритм відновлення пароля користувача

Під час редагування автомобіля клієнтська частина надсилає форму з оновленими характеристиками, VIN-кодом і новими фотографіями. Сервер перевіряє авторизацію та належність автомобіля поточному користувачу,

нормалізує VIN-код, порівнює попередню і нову ціну. Якщо ціна змінилася, створюється запис у `car_price_history`.

Нові фотографії додаються до `car_images`, а власник може окремо видаляти застарілі зображення. На рисунку наведено алгоритм редагування автомобіля та фіксації зміни ціни (рис. 2.10).



Рисунок 2.10 Алгоритм редагування автомобіля та фіксації зміни ціни

Під час проектування алгоритмів роботи веб-застосунку було визначено послідовність виконання основних функціональних процесів системи. Розроблені алгоритми охоплюють авторизацію та відновлення пароля користувачів, створення і редагування оголошень про продаж автомобілів, перевірку VIN-коду, фіксацію історії зміни ціни, формування заявок на придбання транспортних засобів, обмін повідомленнями між користувачами та опрацювання заявок власниками автомобілів.

Запропоновані алгоритми забезпечують узгоджену взаємодію між клієнтською частиною, сервером та базою даних, а також гарантують коректне виконання основних функцій системи. Представлені UML-діаграми діяльності наочно демонструють логіку виконання операцій та дозволяють простежити послідовність обробки даних на різних етапах роботи веб-застосунку.

Розроблені алгоритми стали основою для програмної реалізації функціональних модулів системи, опис яких наведено в наступному підрозділі.

2.4 Реалізація програмного комплексу

Програмний комплекс реалізовано за клієнт-серверною архітектурою. Серверна частина виконує бізнес-логіку, обробляє HTTP-запити, взаємодіє з базою даних MySQL та перевіряє права доступу користувачів. Клієнтська частина забезпечує відображення інтерфейсу, введення даних і взаємодію із сервером через REST API.

Серверна частина містить каталоги config, controllers, routes, middlewares та uploads. Каталог config містить підключення до бази даних. У controllers реалізовано бізнес-логіку. У routes описано маршрути REST API. У middlewares реалізовано перевірку авторизації, адміністративних прав, валідацію даних і завантаження файлів. Каталог uploads використовується для збереження фотографій автомобілів.

Клієнтська частина складається з HTML-сторінок, CSS-файлів і JavaScript-модулів. Основними сторінками є index.html, login.html, register.html, forgot-password.html, reset-password.html, car.html, add-car.html,

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				66
Змн.	Арк.	№ докум.	Підпис	Дата		

edit-car.html, profile.html та admin.html. Загальна структура програмного комплексу представлена на рисунках (рис. 2.11-2.12).

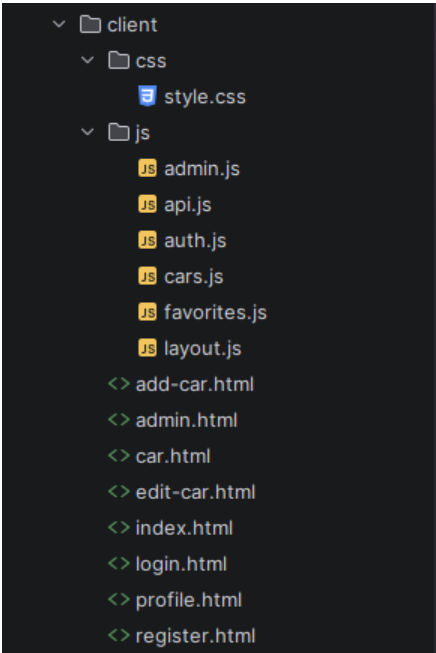


Рисунок 2.11 Загальна структура програмного комплексу

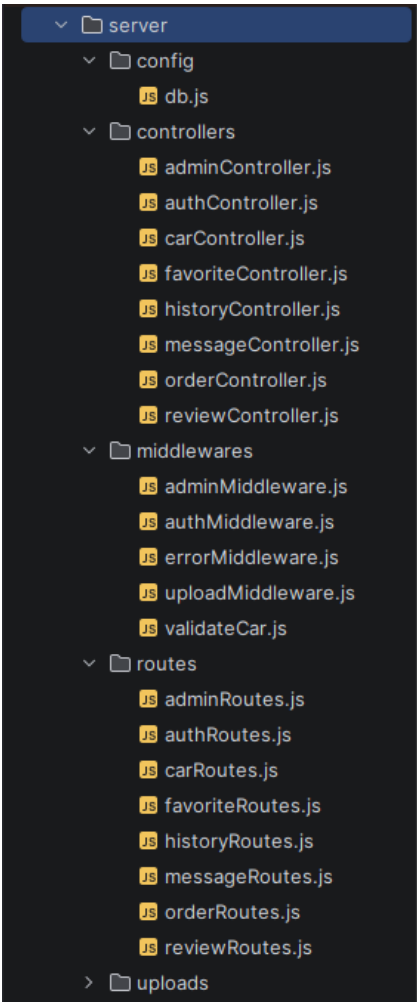


Рисунок 2.12 Загальна структура програмного комплексу

Для забезпечення взаємодії серверної частини програмної системи з базою даних MySQL реалізовано окремий модуль db.js. Для роботи з базою даних використовується бібліотека mysql2/promise, яка забезпечує виконання асинхронних SQL-запитів. Підключення організовано у вигляді пулу з'єднань, що дає змогу повторно використовувати встановлені з'єднання під час виконання декількох запитів та зменшує накладні витрати, пов'язані зі створенням нового з'єднання для кожної операції. Фрагмент реалізації модуля підключення до бази даних наведено в лістингу (лістинг 2.1) [29].

```
const mysql = require('mysql2/promise');
const path = require('path');
require('dotenv').config({ path: path.join(__dirname, '..', '.env') });

const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  waitForConnections: true,
  connectionLimit: 10
});
module.exports = pool;
```

Лістинг 2.1 Фрагмент модуля підключення до бази даних

У наведеному фрагменті параметри підключення зберігаються у файлі середовища .env, що дозволяє не розміщувати конфіденційні дані безпосередньо в програмному коді. Пул підключень використовується всіма контролерами системи.

Модуль db.js також виконує перевірку актуальності структури бази даних під час запуску сервера. Якщо у таблиці cars відсутні службові поля або індекси, система додає їх автоматично. Аналогічно створюються таблиці car_price_history і password_reset_tokens, а також індекси для швидкого доступу до історії цін і токенів відновлення пароля.

Для забезпечення захищеного доступу до функціональних можливостей системи реалізовано механізм автентифікації та авторизації користувачів. Основна логіка розташована у контролері authController.js, а контроль доступу

до захищених маршрутів здійснюється за допомогою middleware. Під час реєстрації система перевіряє коректність отриманих даних та унікальність електронної пошти. Пароль перед збереженням у базі даних проходить хешування за допомогою бібліотеки bcrypt.

```
const hashed = await bcrypt.hash(password, 10);

await db.query(
  'INSERT INTO users (name, email, password, phone, role) VALUES (?, ?, ?, ?, ?)',
  [name, email, hashed, phone, 'user']
);

res.json({ message: 'Registered' });
```

Лістинг 2.2 Фрагмент реєстрації користувача

Застосування хешування забезпечує збереження паролів у базі даних у захищеному вигляді. Після успішної перевірки облікових даних під час авторизації сервер формує JSON Web Token, який використовується для подальшої ідентифікації користувача.

```
const token = jwt.sign(
  {
    id: user.id,
    email: user.email,
    role: user.role
  },
  process.env.JWT_SECRET,
  { expiresIn: '7d' }
);
```

Лістинг 2.3 Формування JWT-токена під час авторизації

До корисного навантаження токена включаються ідентифікатор користувача, електронна пошта та його роль у системі. Строк дії токена встановлено на сім днів, після чого користувач повинен пройти повторну авторизацію.

Перевірка доступу до захищених серверних маршрутів виконується за допомогою authMiddleware. Middleware отримує JWT-токен із заголовка Authorization, перевіряє його коректність за допомогою секретного ключа та передає декодовані дані користувача до наступного етапу обробки запиту [32].

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				69
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const header = req.headers['authorization'];

if (!header) {
  return res.status(401).json({ message: 'No token' });
}

const token = header.split(' ')[1];
const decoded = jwt.verify(token, process.env.JWT_SECRET);
req.user = decoded;
next();

```

Лістинг 2.4 Перевірка JWT-токена

Для операцій, доступних виключно адміністраторам, реалізовано додаткове middleware `adminMiddleware`. Воно перевіряє роль автентифікованого користувача та за відсутності необхідних прав блокує виконання запиту.

```

module.exports = (req, res, next) => {
  if (!req.user || req.user.role !== 'admin') {
    return res.status(403).json({ message: 'Access denied' });
  }
  next();
};

```

Лістинг 2.5 Перевірка адміністративних прав

Таким чином, авторизація реалізована на двох рівнях: спочатку перевіряється факт автентифікації користувача, після чого для адміністративних операцій додатково перевіряється його роль. Це забезпечує розмежування доступу до ресурсів залежно від повноважень користувача.

Функціональність роботи з автомобілями реалізовано у файлі `carController.js`. Відповідний контролер забезпечує отримання переліку доступних автомобілів, пошук і фільтрацію оголошень, перегляд окремого автомобіля, а також створення, редагування та видалення оголошень.

Для отримання списку автомобілів використовується динамічне формування SQL-запиту. Залежно від переданих користувачем параметрів до основного запиту додаються відповідні умови фільтрації за маркою, містом та іншими характеристиками автомобіля. Також передбачено сортування

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				70
Змн.	Арк.	№ докум.	Підпис	Дата		

результатів за ціною, роком випуску, пробігом та датою створення оголошення. Фрагмент реалізації отримання автомобілів із використанням параметрів фільтрації наведено у лістингу (лістинг 2.6).

```
let query = `
  SELECT cars.*, users.name AS seller_name, users.rating AS seller_rating,
  (SELECT image_url FROM car_images
   WHERE car_images.car_id = cars.id
   ORDER BY id ASC LIMIT 1) AS first_image
  FROM cars
  JOIN users ON users.id = cars.user_id
  WHERE cars.status = 'active'
`;
const params = [];

if (req.query.brand) {
  query += ' AND cars.brand LIKE ?';
  params.push(`%${req.query.brand}%`);
}

if (req.query.city) {
  query += ' AND cars.city LIKE ?';
  params.push(`%${req.query.city}%`);
}
```

Лістинг 2.6 Фрагмент отримання автомобілів із фільтрацією

Використання параметризованих SQL-запитів дозволяє безпечно передавати значення фільтрів та зменшує ризик SQL-ін'єкцій. Для роботи з фотографіями застосовується middleware uploadMiddleware, реалізований на основі бібліотеки Multer. Завантажені файли зберігаються у каталозі uploads, а їхні шляхи записуються до таблиці car_images. Фрагмент налаштування завантаження фотографій наведено у лістингу (лістинг 2.7).

```
const storage = multer.diskStorage({
  destination(req, file, cb) {
    cb(null, uploadDir);
  },

  filename(req, file, cb) {
    cb(null, Date.now() + '-' + file.originalname);
  }
});

module.exports = multer({ storage }).array('images', 10);
```

Лістинг 2.7 Фрагмент middleware для завантаження фотографій

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				71
Змн.	Арк.	№ докум.	Підпис	Дата		

Перед створенням або редагуванням оголошення виконується перевірка обов'язкових полів за допомогою `validateCar.js`. Додатково здійснюється нормалізація та перевірка VIN-коду. У базі даних для відповідного поля встановлено унікальний індекс, що не дозволяє створити декілька оголошень з однаковим VIN-кодом. Під час зміни вартості автомобіля попередня та нова ціна зберігаються у таблиці `car_price_history`.

Редагування оголошення також передбачає керування окремими фотографіями. Власник може додавати нові зображення або видаляти наявні. Перед видаленням система перевіряє права доступу користувача, після чого відповідний запис і файл видаляються.

Модуль роботи із заявками на придбання реалізовано у файлі `orderController.js`. Створення заявки виконується в межах транзакції бази даних, що забезпечує цілісність операції та запобігає збереженню неповних даних.

Перед створенням заявки система перевіряє наявність автомобіля, його поточний статус та відсутність дублюючої заявки. Також виконується перевірка, щоб користувач не міг створити заявку на придбання власного автомобіля. Фрагмент реалізації створення заявки наведено у лістингу (лістинг 2.8).

```
const connection = await db.getConnection();
await connection.beginTransaction();

const [cars] = await connection.query(
  'SELECT id, user_id, status FROM cars WHERE id = ?',
  [car_id]
);

if (cars[0].user_id === req.user.id) {
  await connection.rollback();
  return res.status(400).json({ message: 'You cannot order your own car' });
}

const [result] = await connection.query(
  'INSERT INTO orders (buyer_id, car_id, status) VALUES (?, ?, ?)',
  [req.user.id, car_id, 'pending']
);
```

```
await connection.commit();
```

Лістинг 2.8 Фрагмент створення заявки на придбання автомобіля

Після успішного створення заявці встановлюється статус pending. Власник автомобіля може переглянути отриману заявку та підтвердити або відхилити її. У разі підтвердження відповідний автомобіль переводиться у статус sold, що відображає завершення процесу продажу.

Клієнтська частина веб-застосунку реалізована з використанням HTML, CSS та JavaScript. Для уніфікації взаємодії з серверною частиною використовується функція request(), розташована у файлі api.js. Вона формує HTTP-запити, додає JWT-токен до заголовків, визначає формат переданих даних та обробляє відповіді сервера. Реалізацію функції наведено у лістингу (лістинг 2.9) [20].

```
async function request(endpoint, method = 'GET', data = null, isForm = false) {
  const headers = {};
  const token = getToken();

  if (token) {
    headers.Authorization = `Bearer ${token}`;
  }

  if (!isForm) {
    headers['Content-Type'] = 'application/json';
  }

  const response = await fetch(API_URL + endpoint, {
    method,
    headers,
    body: data ? (isForm ? data : JSON.stringify(data)) : null
  });

  const text = await response.text();
  if (!response.ok) {
    throw new Error(text || 'Request failed');
  }
  return JSON.parse(text);
}
```

Лістинг 2.9 Універсальна функція взаємодії з API

Після успішної авторизації отриманий JWT-токен зберігається у localStorage. На основі ролі, що міститься у токени, система визначає

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				73
Змн.	Арк.	№ докум.	Підпис	Дата		

подальший маршрут користувача: адміністратор переходить до адміністративної панелі, а звичайний користувач - на головну сторінку. Фрагмент реалізації авторизації наведено у лістингу (лістинг 2.10).

```
const res = await request('/auth/login', 'POST', {
  email: document.getElementById('email').value,
  password: document.getElementById('password').value
});

if (res.token) {
  localStorage.setItem('token', res.token);
  const payload = JSON.parse(atob(res.token.split('.')[1]));

  if (payload.role === 'admin') {
    window.location.href = 'admin.html';
  } else {
    window.location.href = 'index.html';
  }
}
```

Лістинг 2.10 - Фрагмент авторизації на клієнтській частині

Окремий модуль layout.js відповідає за оновлення навігаційного меню залежно від стану авторизації. Якщо користувач не авторизований, йому відображаються посилання на вхід і реєстрацію. Якщо користувач авторизований, відображаються посилання на профіль, додавання автомобіля та вихід. Для адміністратора додатково додається посилання на адміністративну панель.

Для забезпечення взаємодії клієнтської та серверної частин веб-застосунку реалізовано REST API, маршрути якого розподілено відповідно до основних функціональних модулів системи. Такий підхід забезпечує структурованість програмного коду, спрощує його супровід і дозволяє розмежувати операції авторизації, роботи з автомобілями, заявками, повідомленнями, відгуками та адміністративними функціями. Основні маршрути REST API та їх призначення наведено в таблиці (табл. 2.23).

Таблиця 2.23

Основні маршрути REST API

Модуль	Метод і маршрут	Призначення
auth	POST /auth/register	Реєстрація користувача

Продовження таблиці 2.23

auth	POST /auth/login	Авторизація користувача
auth	POST /auth/password/forgot	Створення посилання для відновлення пароля
auth	POST /auth/password/reset	Зміна пароля за одноразовим токеном
auth	GET /auth/profile	Отримання профілю користувача
auth	PUT /auth/profile	Оновлення профілю та пароля
cars	GET /cars	Отримання списку автомобілів з фільтрацією і сортуванням
cars	GET /cars/user/my	Отримання власних автомобілів зі статистикою переглядів
cars	GET /cars/:id	Отримання деталей автомобіля, фото та історії ціни
cars	POST /cars	Створення нового оголошення
cars	PUT /cars/:id	Редагування автомобіля
cars	DELETE /cars/:id/images/:imageId	Видалення окремого фото автомобіля
cars	DELETE /cars/:id	Видалення власного оголошення
favorites	POST /favorites	Додавання автомобіля до обраного
favorites	GET /favorites	Отримання списку обраних автомобілів
favorites	DELETE /favorites/:carId	Видалення автомобіля з обраного
orders	POST /orders	Створення заявки на купівлю
orders	GET /orders/seller/my	Отримання заявок на автомобілі продавця
orders	GET /orders	Отримання заявок покупця
orders	PUT /orders/:id/approve	Підтвердження заявки та позначення авто як проданого
orders	PUT /orders/:id/reject	Відхилення заявки
messages	POST /messages	Надсилення повідомлення
messages	GET /messages	Отримання повідомлень користувача
messages	GET /messages/seller-conversations	Отримання чатів продавця
messages	GET /messages/conversation/:userId	Отримання конкретного діалогу

messages	PUT /messages/:id/read	Позначення повідомлення як прочитаного
messages	DELETE /messages/:id	Видалення повідомлення
reviews	POST /reviews	Додавання відгуку
reviews	GET /reviews/:carId	Отримання відгуків автомобіля
reviews	GET /reviews/seller/:sellerId	Отримання відгуків продавця
admin	GET /admin/users	Отримання списку користувачів
admin	DELETE /admin/users/:id	Видалення користувача
admin	GET /admin/cars	Отримання всіх оголошень
admin	DELETE /admin/cars/:id	Видалення автомобіля адміністратором

Наведені в таблиці 2.23 маршрути охоплюють основні операції системи: реєстрацію та авторизацію користувачів, керування оголошеннями, роботу з обраними автомобілями, заявками, повідомленнями, відгуками та адміністративними функціями. Використання окремих маршрутів для кожного модуля забезпечує логічну структуру API та стандартизовану взаємодію між клієнтською і серверною частинами.

Компонентна структура програмного комплексу визначає взаємодію основних частин веб-застосунку. Клієнтська частина формує запити до REST API, після чого middleware виконує перевірку доступу, а відповідний контролер обробляє запит і взаємодіє з базою даних. Діаграма компонентів програмного комплексу показана на рисунку (рис. 2.13).

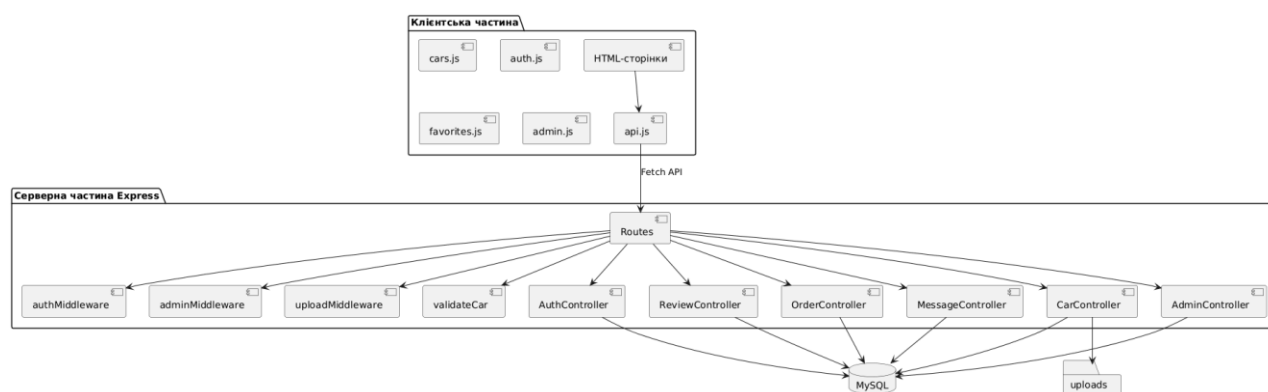


Рисунок 2.13 - Діаграма компонентів програмного комплексу

Розроблена компонентна структура забезпечує розподіл відповідальності між окремими частинами системи, що спрощує її тестування, супровід та подальше розширення. Взаємодія компонентів побудована таким чином, щоб основна бізнес-логіка залишалася на серверній частині, а клієнтський застосунок відповідав переважно за відображення інформації та взаємодію з користувачем.

Висновки до другого розділу

У другому розділі було виконано проєктування веб-застосунку продажу вживаних автомобілів. Визначено основні категорії користувачів, функціональні та нефункціональні вимоги, часові характеристики реакції системи, а також сформовано варіанти використання програмного продукту.

Було розроблено об'єктно-орієнтовану структуру системи, що відображає основні сутності предметної області та їх взаємозв'язки. На основі цієї структури спроектовано реляційну базу даних, яка складається з десяти таблиць і забезпечує зберігання інформації про користувачів, автомобілі, фотографії, повідомлення, відгуки, заявки, обране, історію переглядів, історію зміни ціни та токени відновлення пароля.

Також було спроектовано алгоритми роботи основних процесів системи: авторизації та відновлення пароля користувача, додавання й редагування автомобіля, створення заявки на придбання, обміну повідомленнями, опрацювання заявок власником автомобіля та фіксації історії зміни ціни. Реалізація програмного комплексу виконана за модульним принципом із розподілом на клієнтську та серверну частини. Наведені фрагменти програмного коду демонструють реалізацію авторизації, роботи з автомобілями, повідомленнями, заявками, відгуками, обраним і REST API.

Отримані результати підтверджують, що проєктні рішення відповідають поставленим функціональним вимогам і створюють основу для стабільної роботи веб-застосунку, опису інтерфейсу та тестування, що розглядаються у третьому розділі.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				77
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ВЕБ-ЗАСТОСУНКОМ

3.1. Порядок встановлення та налаштування параметрів системи

Для забезпечення повноцінної роботи розробленого веб-застосунку необхідно виконати попередню підготовку програмного середовища, налаштувати серверну частину, створити та підключити базу даних, а також забезпечити запуск клієнтської частини. Оскільки розроблена система має веб-орієнтовану архітектуру, її функціонування передбачає взаємодію трьох основних складових: клієнтської частини, серверної частини та бази даних. Користувач взаємодіє із системою через веб-браузер, сервер відповідає за оброблення запитів і реалізацію основної бізнес-логіки, а база даних використовується для зберігання інформації, необхідної для функціонування веб-застосунку.

Для розгортання системи на локальному комп'ютері розробника або сервері необхідно підготувати відповідне програмне середовище. Основними компонентами є Node.js для виконання серверного коду, MySQL для організації зберігання даних, npm для керування залежностями програмного проєкту та веб-браузер для роботи користувача з клієнтською частиною. Для редагування та супроводу програмного коду може використовуватися середовище розробки WebStorm або інший сучасний редактор. Основні програмні засоби, необхідні для встановлення та запуску системи, наведено в (таблиці 3.1) [38].

Таблиця 3.1

Таблиця Програмні засоби, необхідні для запуску системи

Компонент	Призначення
Node.js	Запуск серверної частини веб-застосунку
MySQL	Зберігання даних системи
WebStorm	Редагування та супровід програмного коду

Веб-браузер	Робота користувача з клієнтською частиною
npm	Встановлення залежностей серверної частини

Першим етапом встановлення системи є підготовка програмного середовища. Перед запуском веб-застосунку на комп'ютері має бути встановлено Node.js, який забезпечує виконання JavaScript-коду серверної частини поза межами браузера. Разом із Node.js встановлюється менеджер пакетів npm. Він використовується для встановлення бібліотек та інших залежностей, необхідних для функціонування серверної частини веб-застосунку.

Наступним необхідним компонентом є система керування базами даних MySQL. Вона використовується для постійного зберігання інформації, з якою працює веб-застосунок. У межах розробленої системи передбачається використання бази даних з назвою car_marketplace. Перед першим запуском застосунку необхідно створити цю базу даних у середовищі MySQL та виконати SQL-скрипт, який містить команди для створення необхідних таблиць і структури бази даних [28].

Створення бази даних є важливою частиною початкового налаштування, оскільки серверна частина застосунку здійснює звернення до неї під час виконання основних операцій. Зокрема, сервер може отримувати з бази даних інформацію, необхідну для формування відповідей на запити клієнта, додавати нові записи, змінювати наявні дані та виконувати їх пошук. Тому до запуску серверної частини необхідно переконатися, що база даних створена, SQL-скрипт виконаний без помилок, а сервер MySQL доступний для підключення.

Після створення бази даних виконується налаштування параметрів підключення серверної частини до MySQL. Для цього у серверній частині проєкту створюється файл .env, у якому зберігаються необхідні параметри

середовища. До них належать дані, що використовуються для встановлення з'єднання з базою даних, а також інші параметри, необхідні для запуску серверної частини. Використання окремого файлу змінних середовища дає змогу не вказувати такі параметри безпосередньо в програмному коді та спрощує налаштування застосунку під час його перенесення на інше середовище.

Після налаштування параметрів підключення необхідно встановити залежності проєкту. Для цього у каталозі серверної частини виконується команда `npm install`. Вона забезпечує встановлення бібліотек, зазначених у конфігурації проєкту. Після завершення цієї операції серверна частина отримує всі необхідні програмні компоненти для коректної роботи [25].

Основою серверної частини розробленого веб-застосунку є платформа Node.js. Для організації REST API використовується Express. Сервер приймає HTTP-запити від клієнтської частини, визначає необхідний маршрут та передає запит відповідному обробнику. Для розділення відповідальності між окремими елементами серверної частини використовуються контролери. Вони відповідають за виконання основних операцій, пов'язаних з обробленням отриманих від клієнта запитів та формуванням відповідей [24;26].

Окрім контролерів, у серверній частині передбачено використання `middleware`. Проміжне програмне забезпечення дає змогу виконувати додаткові операції під час оброблення HTTP-запитів до їх передачі відповідним контролерам або перед формуванням остаточної відповіді. Такий підхід дає змогу структурувати серверну частину та винести окремі повторювані операції за межі основної логіки контролерів.

Розроблена система має розподілену структуру, у якій кожна складова виконує визначену функцію. Браузер забезпечує взаємодію з користувачем, клієнтські HTML-, CSS- та JavaScript-компоненти формують інтерфейс і забезпечують надсилання запитів, Node.js та Express відповідають за серверну логіку, а MySQL використовується для зберігання даних. Додаткові

компоненти JWT, bcrypt та multer забезпечують реалізацію автентифікації, захисту паролів і роботи із завантаженими файлами.

Для коректного запуску веб-застосунку необхідно дотримуватися визначеної послідовності налаштування: підготувати програмне середовище, створити базу даних car_marketplace, імпортувати структуру таблиць, налаштувати параметри підключення у файлі .env, встановити залежності за допомогою npm install, запустити серверну частину та забезпечити доступ до клієнтської частини через веб-браузер. Виконання цих дій забезпечує необхідні умови для функціонування веб-застосунку та взаємодії між його основними компонентами. Загальну структуру розгортання та взаємодію основних компонентів веб-застосунку наведено на рисунку (рис. 3.1.)

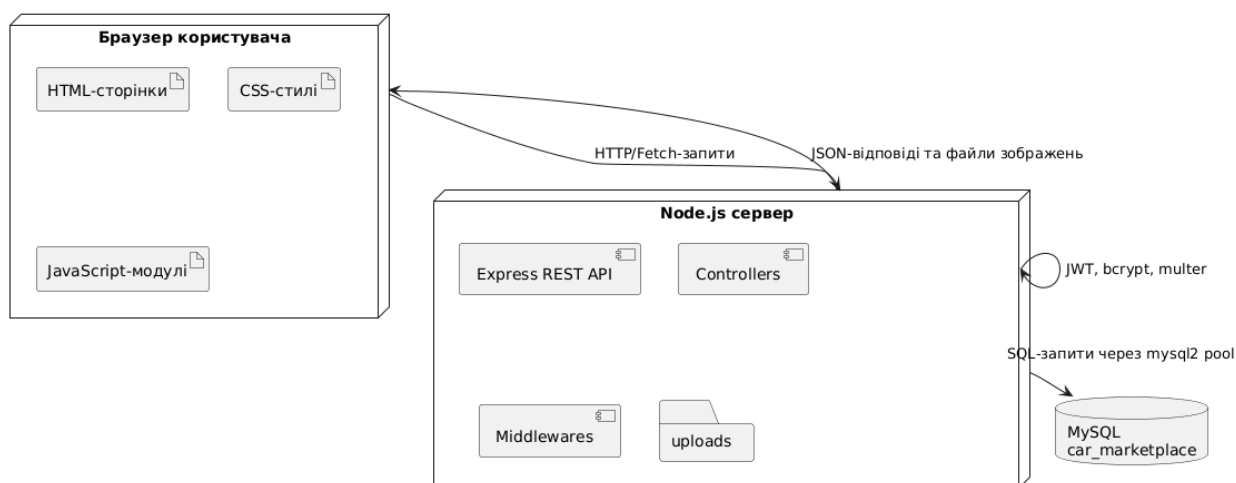


Рисунок 3.1 Діаграма розгортання веб-застосунку

3.2. Структура інтерфейсу веб-застосунку

Інтерфейс веб-застосунку побудований із кількох сторінок, кожна з яких відповідає за окремий функціональний модуль системи. Такий підхід забезпечує логічну навігацію, спрощує роботу користувача та дозволяє розмежувати функції гостей, зареєстрованих користувачів і адміністраторів.

Головна сторінка призначена для перегляду доступних автомобілів. На ній розміщено навігаційне меню, розширену панель фільтрації та список карток автомобілів. Користувач може фільтрувати оголошення за маркою, моделлю, містом, діапазоном ціни, роком випуску, пробігом, типом пального

та типом трансмісії, а також сортувати результати за новизною, ціною, роком випуску або пробігом. Гість може переглядати автомобілі та використовувати фільтри, однак для створення оголошень, додавання автомобіля до обраного, надсилання повідомлень або заявок необхідно авторизуватися, як показано на рисунку (рис. 3.2).

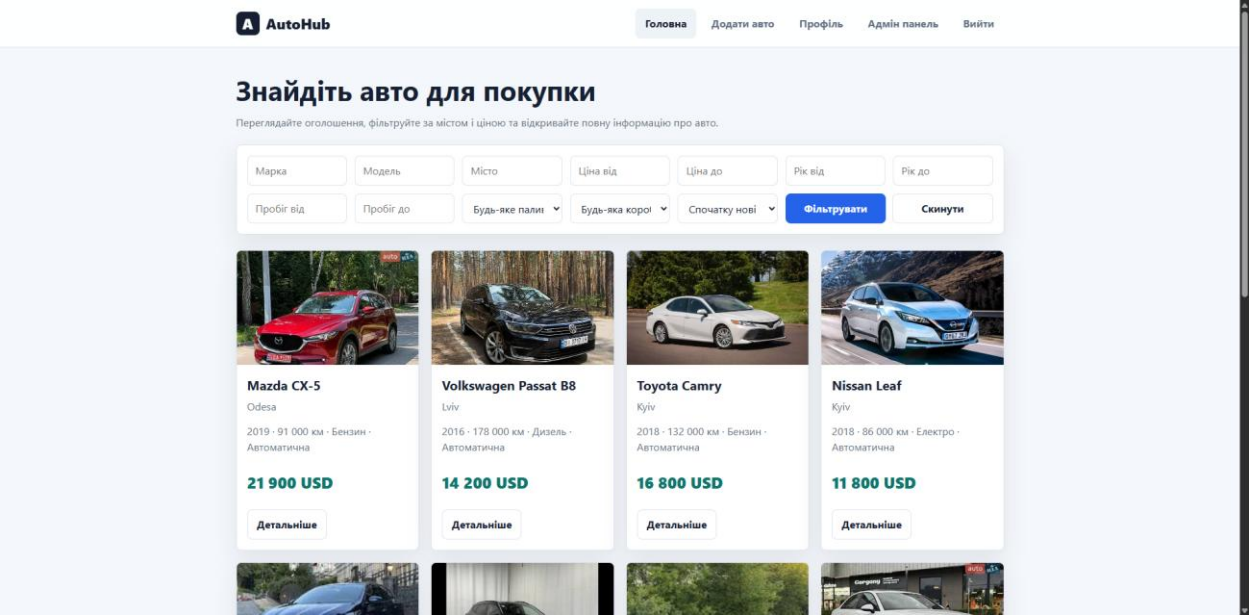


Рисунок 3.2 Головна сторінка веб-застосунку з розширеною фільтрацією

Сторінка реєстрації дозволяє створити новий обліковий запис. Користувач вводить ім'я, електронну пошту, пароль і номер телефону. Вигляд сторінки реєстрації наведено на рисунку (рис 3.3).

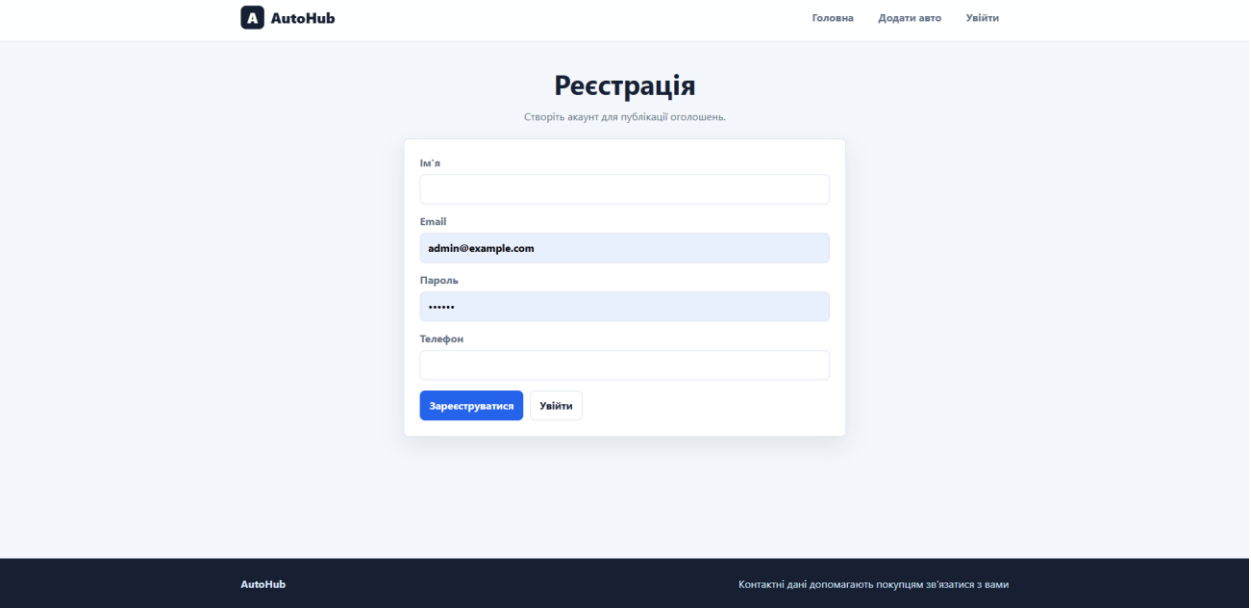


Рисунок 3.3 Сторінка реєстрації користувача

Сторінка авторизації призначена для входу до системи. Після успішного входу токен зберігається у браузері, а користувач перенаправляється на головну сторінку або до адміністративної панелі, якщо його роль дорівнює admin. Вигляд сторінки авторизації наведено на рисунку (рис. 3.4).

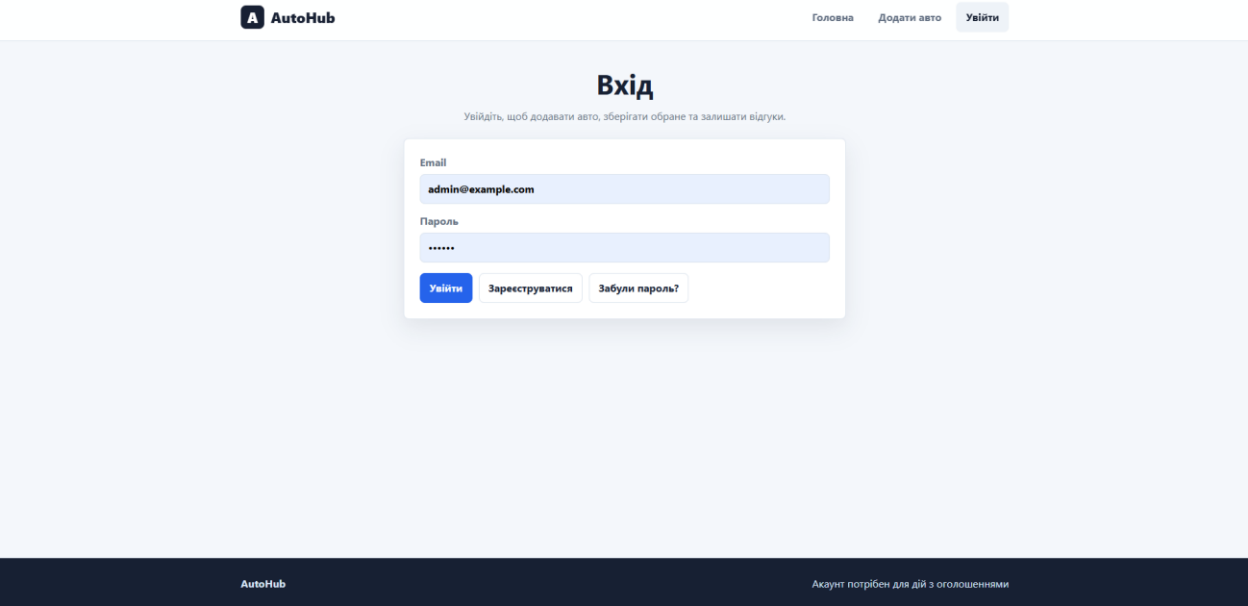


Рисунок 3.4 Сторінка авторизації користувача

Якщо користувач забув пароль, він може перейти на сторінку відновлення, ввести email, отримати одноразове посилання та встановити новий пароль на сторінці reset-password.html., як показано на рисунку (рис. 3.5).

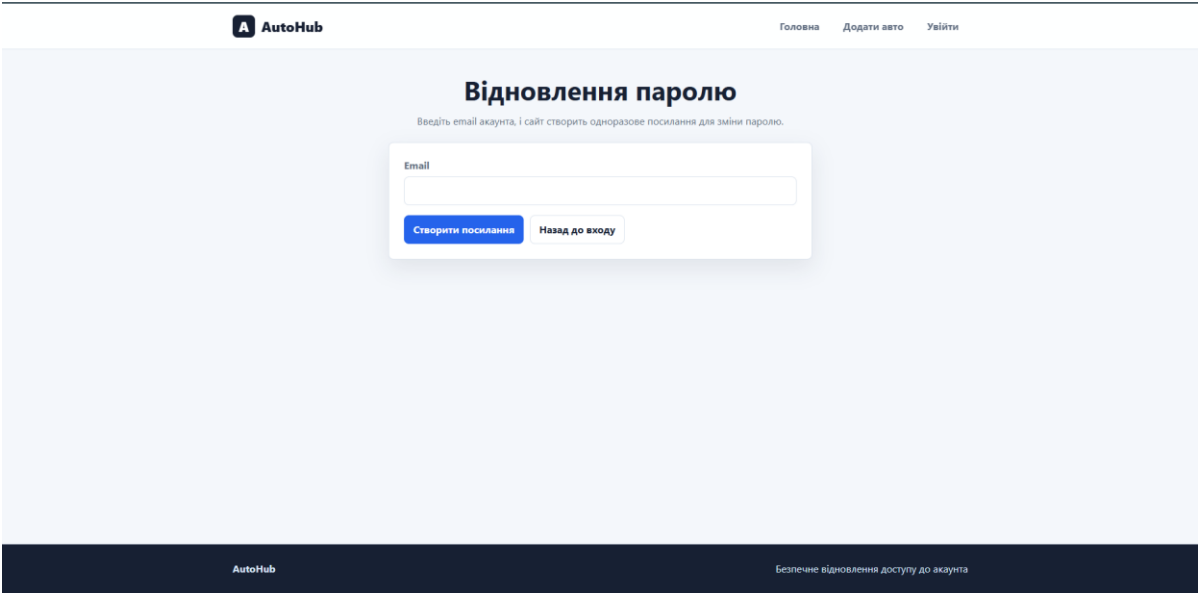


Рисунок 3.5 Сторінка відновлення пароля

Сторінка перегляду автомобіля призначена для відображення детальної інформації про вибране оголошення. У верхній частині сторінки користувач може переглянути основні відомості про транспортний засіб, його фотографії, поточний статус та інформацію про продавця, як показано на рисунку(рис.3.6).

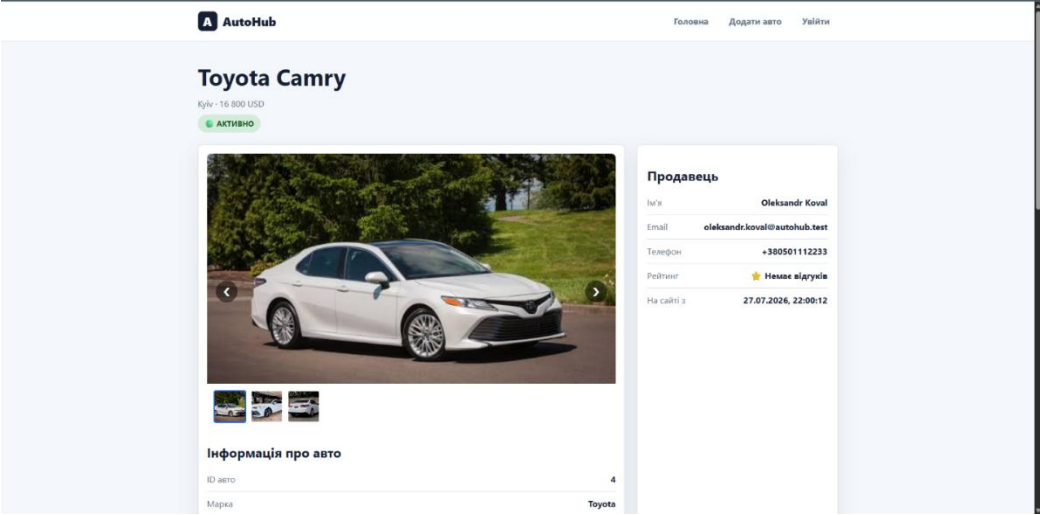


Рисунок 3.6 Сторінка перегляду оголошення про автомобіль

Нижче на сторінці відображаються технічні та експлуатаційні характеристики транспортного засобу. Як показано на рисунку (рис.3.7), користувач може переглянути ідентифікатор автомобіля, марку, модель, VIN-код, місто, ціну, пробіг, рік виготовлення, тип пального, тип трансмісії, статус оголошення та дату його створення. Також передбачено відображення історії зміни ціни та текстового опису автомобіля.

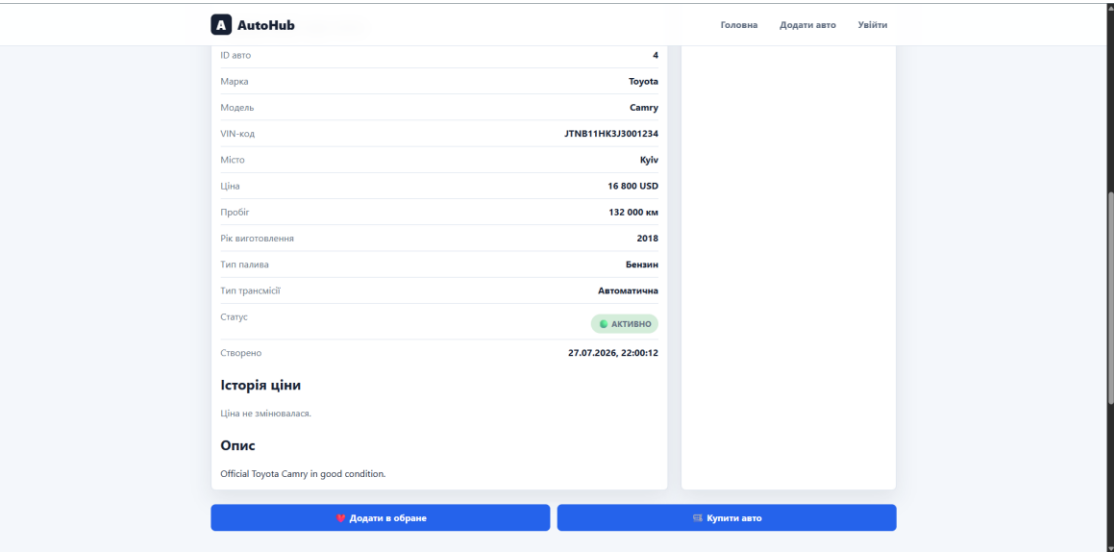


Рисунок 3.7 Детальна інформація про автомобіль

Для взаємодії з оголошенням авторизований користувач може скористатися функціональними кнопками, розміщеними в нижній частині інформаційного блоку. Зокрема, передбачена можливість додати автомобіль до обраного та створити заявку на його купівлю. Якщо автомобіль уже продано, можливість створення нової заявки не відображається.

На сторінці також реалізовано функціональність роботи з відгуками. Користувач може переглянути наявні відгуки та, за необхідності, залишити власний. Форма додавання відгуку містить поле для оцінки автомобіля та текстове поле для введення коментаря. Відповідний елемент інтерфейсу наведено на рисунку (рис. 3.8).

The screenshot displays the 'AutoHub' website interface. At the top, there is a navigation bar with the 'AutoHub' logo and links for 'Головна', 'Додати авто', and 'Увійти'. Below the navigation bar, a car listing is shown with the text 'Ціна не змінювалася.' and 'Опис: Official Toyota Camry in good condition.' Two blue buttons are visible: 'Додати в обране' and 'Купити авто'. Below the listing, the 'Відгуки' (Reviews) section is shown with the text 'Відгуків поки немає.' A form for adding a review is displayed, featuring a rating field labeled 'Оцінка' with a range of '1-5', a text area labeled 'Коментар' with the placeholder 'Ваш коментар', and a blue button labeled 'Надіслати відгук'. The footer of the page includes the 'AutoHub' logo and the text 'Перевіряйте деталі перед покупкою'.

Рисунок 3.8 Форма додавання відгуку про автомобіль

Форма додавання автомобіля містить поля для введення основних характеристик: марки, моделі, VIN-коду, міста, ціни, року випуску, пробігу, типу пального, типу коробки передач, опису та фотографій. Вигляд сторінки додавання нового автомобіля наведено на рисунку (рис.3.9). VIN-код перевіряється на клієнтській і серверній частині, а перша фотографія використовується як обкладинка картки оголошення.

А

AutoHub

Головка

Додати авто

Профіль

Вийти

Додати авто

Заповніть характеристики авто та додайте кілька фотографій. Перше фото буде обкладинкою оголошення.

Марка

Модель

VIN-код

Наприклад WBAJA71070CD12345

Місто

Ціна, USD

Пробіг, км

Рік виготовлення

Тип палива

Оберіть тип палива

Тип трансмісії

Оберіть трансмісію

Опис авто

Опишіть стан, комплектацію та важливі деталі

Фотографії

Вибрати файли

Файл не вибрано

Опублікувати

Назад

Сторінка редагування має подібну структуру, однак поля форми попередньо заповнюються наявними даними автомобіля. Власник може додати нові фотографії, видалити окремі поточні фото, змінити характеристики та ціну. Якщо ціна була змінена, система автоматично зберігає попереднє й нове значення в історії ціни. Вигляд сторінки редагування оголошення наведено на рисунку (рис. 3.10).

Головна

Додати авто

Профіль

Вийти

Оновіть характеристики та опис оголошення.

Марка

Volkswagen

Модель

Passat

VIN-код

Наприклад WBAJA71070CD12345

Місто

Житомир

Ціна, USD

16000,00

Пробіг, км

176000

Рік виготовлення

2018

Тип палива

Бензин

Тип трансмісії

Роботизована

Опис авто

Авто у відмінному стані

Додати нові фото

Вибрати файли | Файл не вибрано

Поточні фото







Видалити

Видалити

Видалити

Оновити

Назад

Особистий кабінет користувача призначений для керування даними профілю, власними оголошеннями, обраними автомобілями, заявками покупця, заявками на власні автомобілі та чатами з покупцями. Користувач може змінити ім'я, телефон і пароль, редагувати або видаляти власні автомобілі, переглядати кількість переглядів оголошення, кількість унікальних переглядачів та дату останнього перегляду. Загальний вигляд особистого кабінету та інформації про профіль і власні оголошення наведено на рисунку (рис. 3.11).

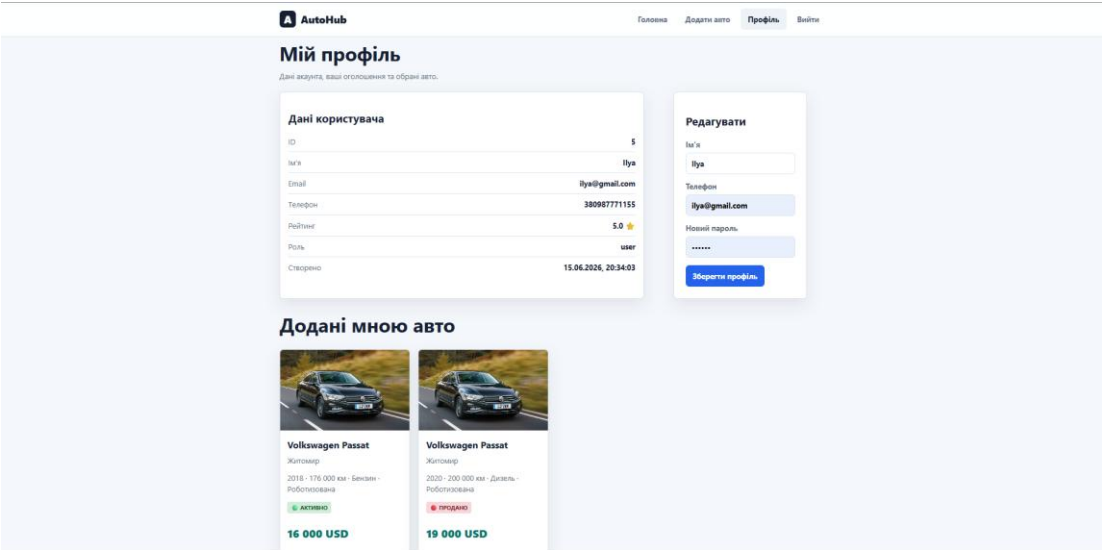


Рисунок 3.11 Особистий кабінет користувача

У розділі особистого кабінету також відображаються обрані автомобілі та заявки, створені користувачем як покупцем. Приклад відображення заявок покупця та їхніх статусів наведено на рисунку (рис. 3.12).

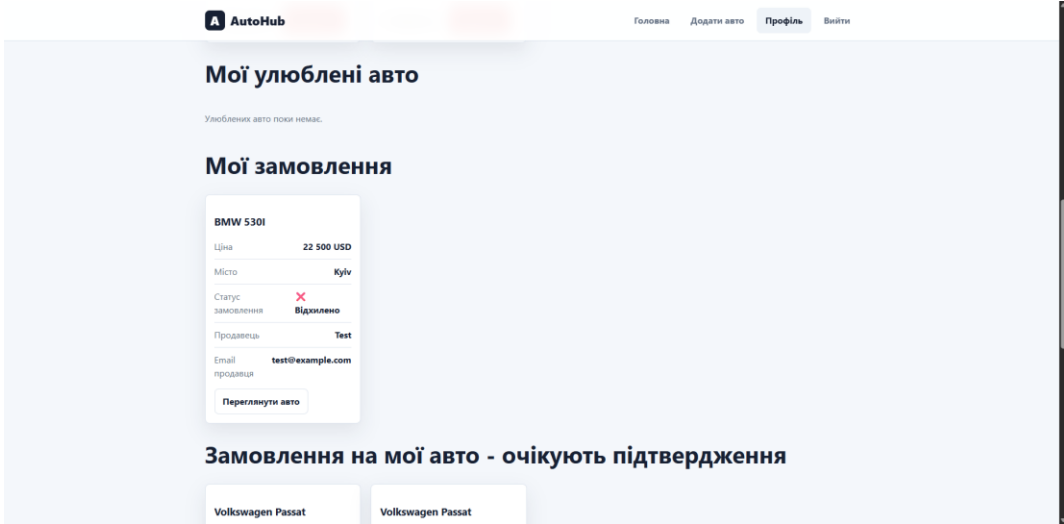


Рисунок 3.12 Заявки користувача на придбання автомобілів

Окремо для власника автомобіля передбачено блок заявок на його оголошення. У ньому відображається інформація про автомобіль, ціну, статус заявки, дані покупця, його електронну пошту та номер телефону. Продавець може переглянути заявки та прийняти рішення щодо їх підтвердження або відхилення. Приклад відображення заявок на власні автомобілі наведено на рисунку (рис. 3.13).

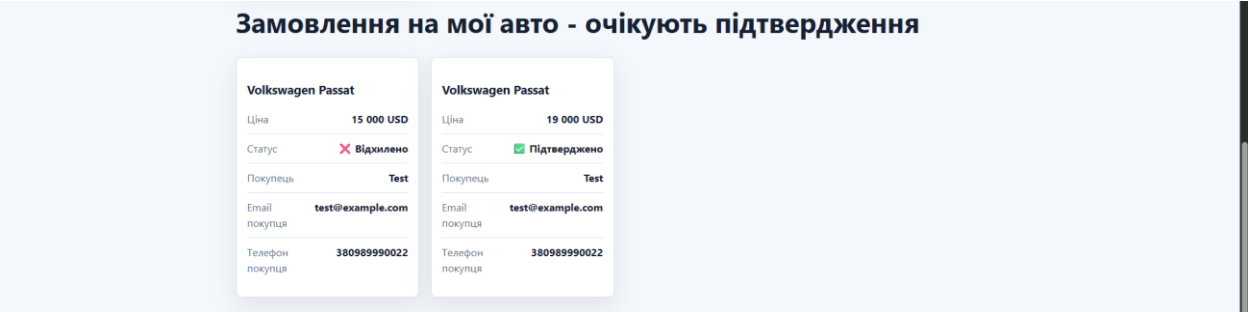


Рисунок 3.13 Заявки на автомобілі користувача

Вбудований чат забезпечує обмін повідомленнями між потенційним покупцем і власником автомобіля. Повідомлення прив'язуються до конкретного оголошення, тому користувачі можуть вести різні діалоги щодо різних автомобілів. Вигляд розділу чатів із покупцями в особистому кабінеті наведено на рисунку (рис. 3.14). Для покупця чат доступний на сторінці автомобіля, а для продавця в особистому кабінеті формується список діалогів із покупцями, останнім повідомленням і кількістю непрочитаних повідомлень.

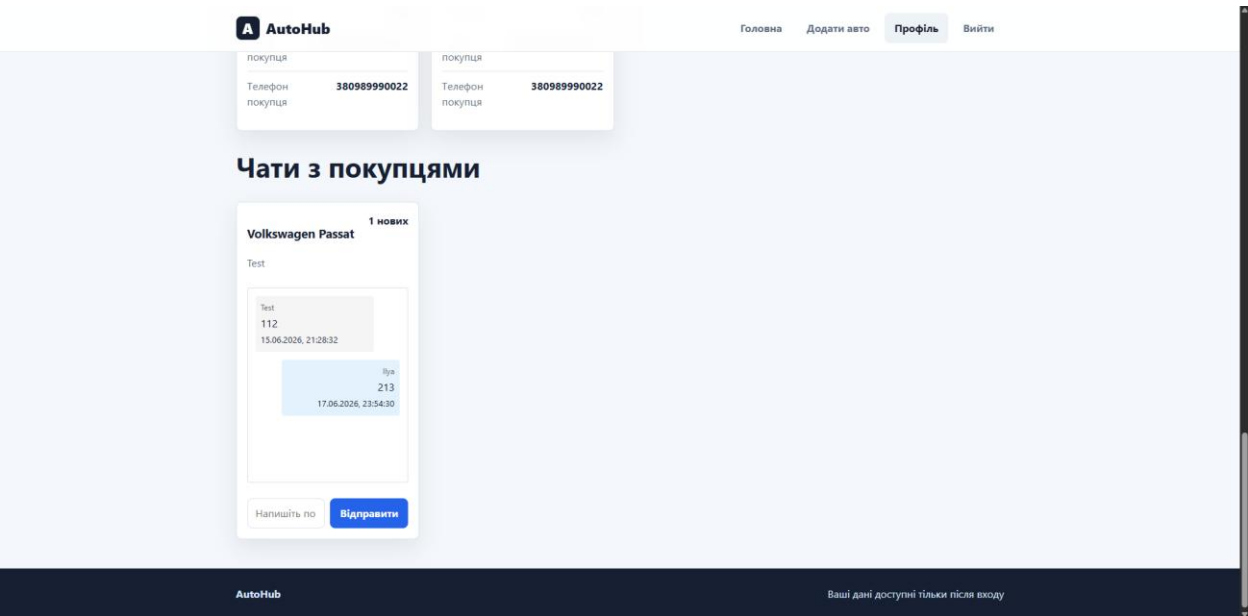


Рисунок 3.14 Інтерфейс чату між покупцем і продавцем

Адміністративна панель доступна лише користувачам із роллю admin та призначена для керування основними даними системи. Вона містить вкладки для перегляду та керування користувачами й автомобілями. Вигляд розділу керування користувачами наведено на рисунку (рис. 3.15).

У картках користувачів відображаються ідентифікатор, email, роль, телефон, рейтинг, ознака блокування та дата створення облікового запису. Адміністратор має можливість видаляти облікові записи користувачів у разі необхідності

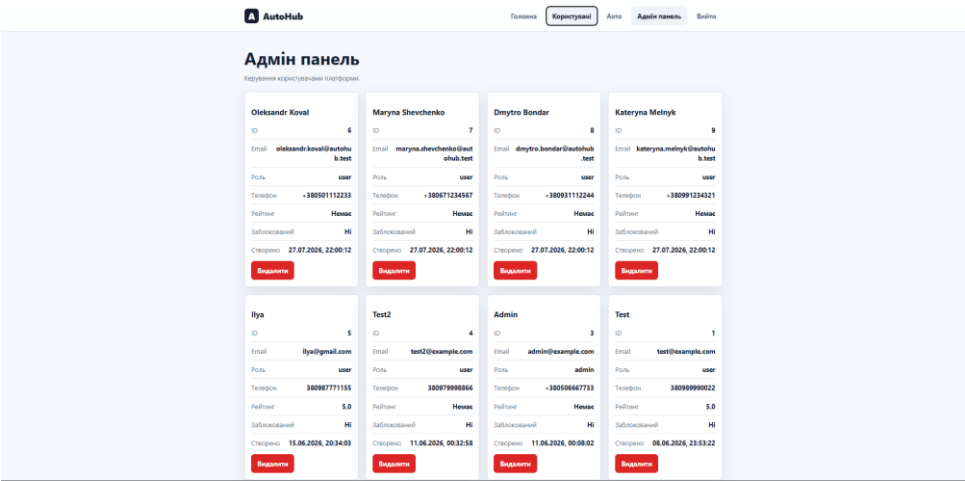


Рисунок 3.15 Розділ керування користувачами адміністративної панелі

У розділі керування автомобілями адміністратор може переглядати основні дані оголошень, інформацію про продавця та його email. Вигляд розділу керування автомобілями наведено на рисунку (рис. 3.16). Адміністратор також може видалити некоректне або небажане оголошення.

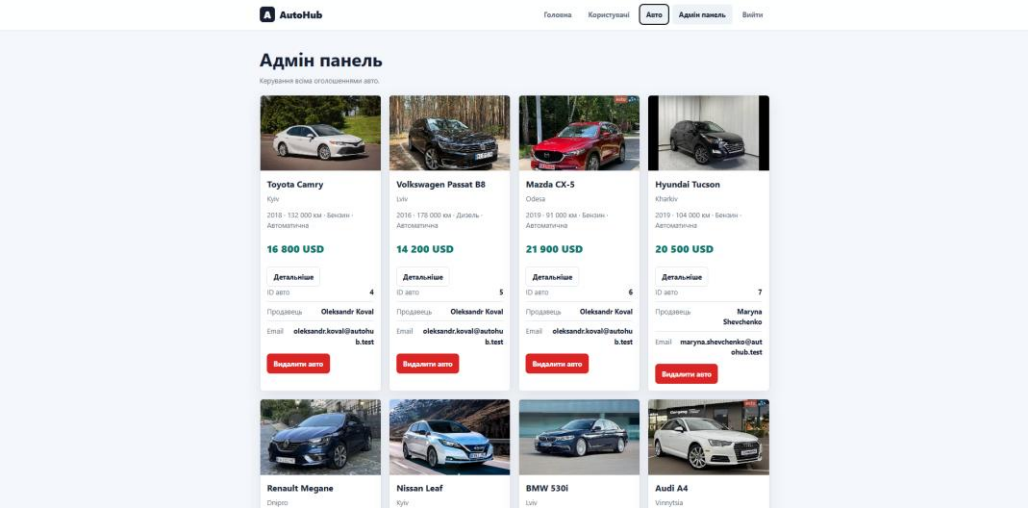


Рисунок 3.16 Розділ керування автомобілями адміністративної панелі

3.3. Тестування роботи програмного комплексу

Тестування програмного комплексу виконувалося з метою перевірки коректності реалізації основних функціональних можливостей розробленої системи, а також виявлення можливих помилок під час взаємодії користувача з веб-застосунком. Основна увага під час тестування приділялася перевірці правильності виконання передбачених сценаріїв роботи та відповідності отриманої поведінки системи встановленим функціональним вимогам.

У процесі тестування розглядалися різні категорії користувачів, оскільки набір доступних їм функцій відрізняється залежно від їхнього статусу та ролі в системі. Перевірялися сценарії роботи гостя, зареєстрованого користувача, власника автомобіля та адміністратора. Такий підхід дав змогу перевірити не лише окремі функції, а й розмежування доступу до них відповідно до ролі користувача.

На першому етапі перевірялися функції, пов'язані зі створенням та використанням облікового запису. Тестувалася реєстрація нового користувача із введенням необхідних персональних даних, а також подальша авторизація за допомогою електронної пошти та пароля. Окремо перевірялася можливість відновлення пароля у випадку втрати користувачем доступу до облікового запису. Перевірялося коректне перенаправлення користувача після авторизації та надання доступу до функцій відповідно до його ролі.

Значну частину тестування становили операції, пов'язані з розміщенням та керуванням оголошеннями про автомобілі. Перевірялася можливість додавання нового автомобіля із заповненням основних характеристик, зокрема марки, моделі, VIN-коду, міста, ціни, року випуску, пробігу, типу пального, типу трансмісії та опису. Окремо перевірялася коректність оброблення VIN-коду на клієнтській і серверній частинах системи.

Також тестувалася можливість додавання фотографій до оголошення та подальше керування фотографіями. Перевірялося завантаження зображень, їх відображення в оголошенні, додавання нових фотографій під час редагування та видалення окремих зображень. Оскільки перша фотографія

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				90
Змн.	Арк.	№ докум.	Підпис	Дата		

використовується як основне зображення оголошення, її відображення також належить до функцій, які підлягали перевірці.

Окремо перевірялися функції пошуку та перегляду оголошень. Тестувалася розширена фільтрація автомобілів за передбаченими параметрами, а також сортування результатів. Перевірялася можливість отримання переліку автомобілів відповідно до заданих користувачем критеріїв. Крім того, тестувалися операції редагування та видалення власних оголошень, доступні користувачу, який є їхнім власником.

Наступним напрямом тестування була перевірка функцій взаємодії між продавцем та потенційним покупцем. Перевірялося надсилання повідомлень через вбудований чат, причому повідомлення повинні бути пов'язані з конкретним оголошенням про автомобіль. Також тестувалося створення заявки на купівлю автомобіля авторизованим користувачем. Для власника автомобіля перевірялася можливість перегляду отриманих заявок та їх подальшого опрацювання.

У межах тестування процесу купівлі також перевірялася фіксація статусу автомобіля sold після відповідної обробки заявки власником. Це дає змогу перевірити узгодженість стану оголошення з результатом операції купівлі та коректність подальшого відображення його статусу в системі.

Окремо перевірялася функція залишення відгуків про автомобілі. Тестувалася можливість введення оцінки та текстового коментаря, а також їх подальше відображення на сторінці відповідного автомобіля. Це дає змогу перевірити взаємодію користувача з функцією оцінювання та збереження відповідних даних у системі.

Також у процесі тестування перевірялися операції з профілем користувача. Зокрема, тестувалася можливість оновлення персональних даних та зміни пароля. Перевірялося, що внесені користувачем зміни коректно обробляються системою та відображаються після повторного завантаження відповідного розділу.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				91
Змн.	Арк.	№ докум.	Підпис	Дата		

Завершальним напрямом тестування була адміністративна панель. Перевірялася доступність адміністративних функцій лише для користувачів із відповідною роллю, а також можливість перегляду та керування користувачами й оголошеннями. Окрему увагу приділено операціям видалення даних, доступним адміністратору.

Таблиця 3.2

Тестові сценарії перевірки програмного комплексу

№	Сценарій	Очікуваний результат	Результат
1	Відкриття головної сторінки	Відображається список активних автомобілів	Успішно
2	Розширена фільтрація	Відображаються авто за моделлю, роком, пробігом, паливом і трансмісією	Успішно
3	Сортування оголошень	Список змінює порядок за ціною, роком або пробігом	Успішно
4	Реєстрація нового користувача	Створюється новий обліковий запис	Успішно
5	Авторизація користувача	Користувач входить до системи, токен зберігається	Успішно
6	Створення посилання відновлення пароля	Генерується одноразове посилання з токеном	Успішно
7	Зміна пароля за токеном	Пароль оновлюється, токен видаляється	Успішно
8	Додавання автомобіля з VIN-кодом	Оголошення зберігається, VIN проходить перевірку	Успішно
9	Спроба додати дубльований VIN	Система повертає повідомлення про вже наявний VIN-код	Успішно
10	Редагування ціни автомобіля	Дані оновлюються, зміна ціни записується в історію	Успішно
11	Видалення окремого фото	Фото зникає зі списку поточних зображень	Успішно
12	Перегляд сторінки авто	Відображаються галерея, VIN, статус, продавець та історія ціни	Успішно
13	Додавання автомобіля до обраного	Автомобіль з'являється у списку обраних	Успішно
14	Надсилання повідомлення продавцю	Повідомлення зберігається та відображається в чаті	Успішно
15	Перегляд чатів продавця	Продавець бачить діалоги і кількість нових повідомлень	Успішно
16	Створення заявки на купівлю	Створюється заявка зі статусом pending	Успішно
17	Підтвердження заявки власником	Заявка стає approved, авто отримує статус sold	Успішно
18	Відхилення заявки власником	Статус заявки змінюється на rejected	Успішно

Продовження таблиці 3.2

19	Додавання відгуку	Відгук зберігається, рейтинг продавця оновлюється	Успішно
20	Оновлення профілю	Ім'я, телефон або пароль користувача оновлюються	Успішно
21	Вхід адміністратора	Адміністратор переходить до admin.html	Успішно
22	Видалення користувача адміністратором	Користувач і пов'язані записи видаляються	Успішно
23	Видалення автомобіля адміністратором	Оголошення видаляється незалежно від власника	Успішно

Окремо було перевірено обробку помилкових ситуацій. Наприклад, користувач не може створити заявку на власний автомобіль, неавторизований користувач не може додати оголошення або надіслати повідомлення, а користувач без ролі адміністратора не може отримати доступ до адміністративних маршрутів. Під час тестування також перевірялася робота валідації даних. Якщо користувач не заповнює обов'язкові поля форми автомобіля, сервер повертає повідомлення про помилку. Якщо пароль під час входу введено неправильно, авторизація не виконується.

Висновки до третього розділу

У третьому розділі було описано порядок встановлення та налаштування веб-застосунку, структуру його користувацького інтерфейсу та результати тестування основних функціональних можливостей. Визначено необхідні програмні засоби для запуску системи, порядок створення бази даних, налаштування змінних середовища та запуску серверної частини.

Розглянуто основні сторінки веб-застосунку: головну сторінку з розширеною фільтрацією, сторінки реєстрації, авторизації та відновлення пароля, сторінку перегляду автомобіля, форми додавання й редагування автомобіля, особистий кабінет, чат і адміністративну панель. Описано призначення кожного елемента інтерфейсу та порядок взаємодії користувача із системою.

Проведене тестування підтвердило коректність роботи основних модулів програмного комплексу. Було перевірено реєстрацію, авторизацію, відновлення пароля, пошук, розширену фільтрацію та сортування автомобілів,

		Поліщук І. А.			ДУ «Житомирська політехніка». 26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				93
Змн.	Арк.	№ докум.	Підпис	Дата		

створення, редагування та видалення оголошень, перевірку VIN-коду, керування фотографіями, роботу обраного, повідомлень, заявок, відгуків, історії ціни, статистики переглядів і адміністративних функцій. Отримані результати свідчать про те, що реалізований веб-застосунок відповідає визначеним функціональним вимогам і може бути використаний як прототип онлайн-платформи для продажу вживаних автомобілів.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				94
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

Під час виконання кваліфікаційної роботи бакалавра було спроектовано та реалізовано вебзастосунок продажу вживаних автомобілів. У межах роботи було проведено аналіз предметної області, визначено актуальність створення подібної системи та обґрунтовано необхідність автоматизації процесів розміщення оголошень, пошуку транспортних засобів і взаємодії між продавцями та покупцями.

У першому розділі було розглянуто особливості функціонування вебплатформ для продажу автомобілів, проаналізовано існуючі аналоги. У результаті аналізу було визначено їх основні переваги та недоліки. Встановлено, що більшість наявних платформ забезпечують базові можливості пошуку й перегляду оголошень, однак не завжди повною мірою реалізують механізми взаємодії між користувачами, опрацювання заявок на купівлю, історії зміни ціни та зручного адміністрування даних. На основі цього було сформовано основні вимоги до розроблюваного вебзастосунку. Також у першому розділі було обґрунтовано вибір архітектури та інструментальних засобів реалізації системи. Для побудови програмного продукту було обрано клієнт-серверну архітектуру монолітного типу, яка є доцільною для реалізації вебзастосунку такого масштабу. Клієнтська частина реалізується з використанням HTML, CSS та JavaScript, серверна частина - на основі Node.js, а для зберігання даних використовується реляційна система керування базами даних MySQL.

У другому розділі було виконано проектування вебзастосунку продажу вживаних автомобілів. Було визначено основні категорії користувачів системи: гість, зареєстрований користувач, власник оголошення та адміністратор. Для кожної категорії було описано доступні функціональні можливості. Гість може переглядати оголошення та використовувати пошук, зареєстрований користувач отримує можливість створювати оголошення, додавати автомобілі до обраного, надсилати заявки на придбання та

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				95
Змн.	Арк.	№ докум.	Підпис	Дата		

обмінюватися повідомленнями, а адміністратор здійснює контроль користувачів і розміщених оголошень.

У процесі проєктування було сформовано функціональні та нефункціональні вимоги до системи, побудовано варіанти використання, описано об'єктно-орієнтовану структуру вебзастосунку та визначено основні сутності предметної області. Окрему увагу було приділено проєктуванню бази даних. Було розроблено структуру таблиць для зберігання інформації про користувачів, автомобілі, фотографії, обране, повідомлення, заявки на придбання, відгуки, історію зміни ціни та токени відновлення пароля. Використання зовнішніх ключів, індексів і логічних зв'язків між таблицями забезпечує цілісність даних та ефективну роботу системи.

Також у другому розділі було описано алгоритми роботи основних функціональних процесів вебзастосунку. Зокрема, розглянуто алгоритми авторизації користувача, додавання автомобіля, надсилання заявки на придбання, обміну повідомленнями, підтвердження або відхилення заявки, відновлення пароля, редагування автомобіля та збереження історії зміни ціни. Реалізовані алгоритми забезпечують узгоджену взаємодію між клієнтською частиною, сервером і базою даних.

У межах реалізації програмного комплексу було створено серверну частину з окремими модулями для роботи з користувачами, автомобілями, заявками, повідомленнями, відгуками, обраним і адміністративними функціями. Було реалізовано REST API, маршрутизацію запитів, middleware для перевірки авторизації, підключення до бази даних MySQL та механізми обробки завантажених зображень. Клієнтська частина забезпечує взаємодію користувача із системою через вебінтерфейс і виконує запити до серверної частини.

У третьому розділі було описано порядок встановлення та налаштування вебзастосунку, структуру його користувацького інтерфейсу та результати тестування. Було наведено послідовність підготовки програмного середовища, створення бази даних, налаштування параметрів підключення, встановлення

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				96
Змн.	Арк.	№ докум.	Підпис	Дата		

залежностей і запуску серверної частини. Це дозволяє розгорнути систему на локальному комп'ютері або сервері та забезпечити її подальше використання. Інтерфейс вебзастосунку було розроблено з урахуванням основних сценаріїв роботи користувачів. Описано головну сторінку з можливістю перегляду та фільтрації оголошень, сторінки реєстрації й авторизації, сторінку перегляду автомобіля, форми додавання та редагування оголошень, особистий кабінет користувача, чат між користувачами та адміністративну панель. Реалізований інтерфейс забезпечує зручну роботу з основними функціями системи та підтримує логічну взаємодію між користувачем і програмним продуктом.

Проведене тестування підтвердило коректність роботи основних модулів вебзастосунку. Було перевірено реєстрацію та авторизацію користувачів, відновлення пароля, перегляд оголошень, розширену фільтрацію, сортування, створення та редагування автомобілів, завантаження фотографій, додавання до обраного, надсилання заявок, обмін повідомленнями, залишення відгуків, роботу особистого кабінету та адміністративної панелі. Результати тестування показали, що система виконує поставлені функціональні вимоги.

Отже, у результаті виконання кваліфікаційної роботи було створено вебзастосунок продажу вживаних автомобілів, який забезпечує автоматизацію основних процесів купівлі та продажу транспортних засобів. Розроблена система дозволяє користувачам переглядати оголошення, здійснювати пошук і фільтрацію, створювати власні оголошення, керувати фотографіями, надсилати заявки на придбання, обмінюватися повідомленнями та залишати відгуки. Практичне значення роботи полягає у можливості використання розробленого вебзастосунку як прототипу онлайн-платформи для продажу вживаних автомобілів із можливістю подальшого розширення функціоналу.

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				97
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. AUTO.RIA. Автомобілі в Україні. URL: <https://auto.ria.com/> (дата звернення: 12.08.2026)..
2. RST.ua. Автобазар України. URL: <https://rst.ua/> (дата звернення: 12.08.2026).
3. ДСТУ ISO/IEC/IEEE 12207:2018. Інженерія систем і програмного забезпечення. Процеси життєвого циклу програмного забезпечення. Київ : ДП «УкрНДНЦ», 2018.
4. Pressman R. S., Maxim B. R. Software Engineering: A Practitioner's Approach. 9th ed. New York : McGraw-Hill Education, 2020. 705 p.
5. Sommerville I. Software Engineering. 10th ed. Boston : Pearson, 2016. 816 p.
6. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.
7. Booch G., Rumbaugh J., Jacobson I. The Unified Modeling Language User Guide. 2nd ed. Boston : Addison-Wesley Professional, 2005. 496 p.
8. Larman C. Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development. 3rd ed. New Jersey : Prentice Hall, 2004. 736 p.
9. Object Management Group. Unified Modeling Language. Version 2.5.1. URL: <https://www.omg.org/spec/UML/2.5.1> (дата звернення: 12.08.2026).
10. ISO/IEC/IEEE 29148:2018. Systems and software engineering — Life cycle processes — Requirements engineering. Geneva : ISO, 2018.
11. ISO/IEC/IEEE 42010:2022. Software, systems and enterprise — Architecture description. Geneva : ISO, 2022.
12. Fowler M. Patterns of Enterprise Application Architecture. Boston : Addison-Wesley Professional, 2002. 560 p.
13. Fowler M., Lewis J. Microservices. URL: <https://martinfowler.com/articles/microservices.html> (дата звернення: 12.08.2026).

14. WHATWG. HTML Living Standard. URL: <https://html.spec.whatwg.org/>
(дата звернення: 12.08.2026).
15. W3C. HTML 5.2 W3C Recommendation. URL:
<https://www.w3.org/TR/html52/> (дата звернення: 12.08.2026).
16. W3C. Cascading Style Sheets. URL: <https://www.w3.org/Style/CSS/> (дата
звернення: 12.08.2026).
17. MDN Web Docs. HTML: HyperText Markup Language. URL:
<https://developer.mozilla.org/en-US/docs/Web/HTML> (дата звернення:
12.08.2026).
18. MDN Web Docs. CSS: Cascading Style Sheets. URL:
<https://developer.mozilla.org/en-US/docs/Web/CSS> (дата звернення:
12.08.2026).
19. MDN Web Docs. JavaScript. URL: [https://developer.mozilla.org/en-
US/docs/Web/JavaScript](https://developer.mozilla.org/en-US/docs/Web/JavaScript) (дата звернення: 12.08.2026).
20. MDN Web Docs. Fetch API. URL: [https://developer.mozilla.org/en-
US/docs/Web/API/Fetch_API](https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API) (дата звернення: 12.08.2026).
21. Flanagan D. JavaScript: The Definitive Guide. 7th ed. Sebastopol : O'Reilly
Media, 2020. 706 p.
22. Haverbeke M. Eloquent JavaScript: A Modern Introduction to Programming.
3rd ed. San Francisco : No Starch Press, 2018. 472 p.
23. Node.js Documentation. URL: <https://nodejs.org/docs/latest/api/> (дата
звернення: 12.08.2026).
24. Express.js. Fast, unopinionated, minimalist web framework for Node.js.
URL: <https://expressjs.com/> (дата звернення: 12.08.2026).
25. npm Documentation. URL: <https://docs.npmjs.com/> (дата звернення:
12.08.2026).
26. RFC 9110. HTTP Semantics. URL: <https://www.rfc-editor.org/rfc/rfc9110>
(дата звернення: 12.08.2026).
27. MySQL 8.0 Reference Manual. URL:
<https://dev.mysql.com/doc/refman/8.0/en/> (дата звернення: 12.08.2026).

28. MySQL Workbench Manual. URL: <https://dev.mysql.com/doc/workbench/en/> (дата звернення: 12.08.2026).
29. Beaulieu A. Learning SQL: Generate, Manipulate, and Retrieve Data. 3rd ed. Sebastopol : O'Reilly Media, 2020. 384 p.
30. RFC 7519. JSON Web Token (JWT). URL: <https://www.rfc-editor.org/rfc/rfc7519> (дата звернення: 12.08.2026).
31. JWT.io. Introduction to JSON Web Tokens. URL: <https://jwt.io/introduction> (дата звернення: 12.08.2026).
32. npm. jsonwebtoken. URL: <https://www.npmjs.com/package/jsonwebtoken> (дата звернення: 12.08.2026).
33. npm. bcrypt. URL: <https://www.npmjs.com/package/bcrypt> (дата звернення: 12.08.2026).
34. OWASP Foundation. Password Storage Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Password_Storage_Cheat_Sheet.html (дата звернення: 12.08.2026).
35. npm. Multer. URL: <https://www.npmjs.com/package/multer> (дата звернення: 12.08.2026).
36. OWASP Foundation. OWASP Top Ten Web Application Security Risks. URL: <https://owasp.org/www-project-top-ten/> (дата звернення: 12.08.2026).
37. OWASP Foundation. Authentication Cheat Sheet. URL: https://cheatsheetseries.owasp.org/cheatsheets/Authentication_Cheat_Sheet.html (дата звернення: 12.08.2026).
38. JetBrains. WebStorm Documentation. URL: <https://www.jetbrains.com/help/webstorm/> (дата звернення: 12.08.2026).

ДОДАТКИ

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				101
Змн.	Арк.	№ докум.	Підпис	Дата		

Лістинг db.js

```

const mysql = require('mysql2/promise');
const path = require('path');
require('dotenv').config({ path: path.join(__dirname, '..', '.env') });

const pool = mysql.createPool({
  host: process.env.DB_HOST,
  user: process.env.DB_USER,
  password: process.env.DB_PASSWORD,
  database: process.env.DB_NAME,
  waitForConnections: true,
  connectionLimit: 10
});

(async () => {
  try {
    const conn = await pool.getConnection();
    console.log('MySQL connected');

    // Add missing columns to cars table
    const carColumns = [
      ['mileage', 'INT NULL'],
      ['year', 'INT NULL'],
      ['vin_code', 'VARCHAR(17) NULL AFTER model'],
      ['fuel_type', 'VARCHAR(50) NULL'],
      ['transmission', 'VARCHAR(50) NULL'],
      ['description', 'TEXT NULL']
    ];

    for (const [column, definition] of carColumns) {
      const [rows] = await conn.query(
        `SELECT COLUMN_NAME
         FROM INFORMATION_SCHEMA.COLUMNS
         WHERE TABLE_SCHEMA = DATABASE()
              AND TABLE_NAME = 'cars'
              AND COLUMN_NAME = ?`,
        [column]
      );

      if (!rows.length) {
        console.log(`Adding column ${column} to cars table`);
        await conn.query(`ALTER TABLE cars ADD COLUMN ${column}
${definition}`);
      }
    }

    const [titleRows] = await conn.query(
      `SELECT COLUMN_NAME
       FROM INFORMATION_SCHEMA.COLUMNS
       WHERE TABLE_SCHEMA = DATABASE()
            AND TABLE_NAME = 'cars'
            AND COLUMN_NAME = 'title'`
    );

    if (!titleRows.length) {
      console.log('Adding title column to cars table');
      await conn.query(`ALTER TABLE cars ADD COLUMN title VARCHAR(255)
NULL AFTER user_id`);
    }
  }
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				102
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const [vinIndexRows] = await conn.query(
  `SELECT INDEX_NAME
   FROM INFORMATION_SCHEMA.STATISTICS
   WHERE TABLE_SCHEMA = DATABASE()
     AND TABLE_NAME = 'cars'
     AND INDEX_NAME = 'idx_cars_vin_code'`
);

if (!vinIndexRows.length) {
  console.log('Adding VIN unique index to cars table');
  await conn.query('CREATE UNIQUE INDEX idx_cars_vin_code ON
cars(vin_code)');
}

await conn.query(`
CREATE TABLE IF NOT EXISTS car_price_history (
  id INT AUTO_INCREMENT PRIMARY KEY,
  car_id INT NOT NULL,
  old_price DECIMAL(10,2) NOT NULL,
  new_price DECIMAL(10,2) NOT NULL,
  changed_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (car_id) REFERENCES cars(id) ON DELETE CASCADE
)
`);

const [priceHistoryIndexRows] = await conn.query(
  `SELECT INDEX_NAME
   FROM INFORMATION_SCHEMA.STATISTICS
   WHERE TABLE_SCHEMA = DATABASE()
     AND TABLE_NAME = 'car_price_history'
     AND INDEX_NAME = 'idx_car_price_history_car'`
);

if (!priceHistoryIndexRows.length) {
  console.log('Adding car price history index');
  await conn.query('CREATE INDEX idx_car_price_history_car ON
car_price_history(car_id, changed_at)');
}

await conn.query(`
CREATE TABLE IF NOT EXISTS password_reset_tokens (
  id INT AUTO_INCREMENT PRIMARY KEY,
  user_id INT NOT NULL,
  token_hash VARCHAR(64) UNIQUE NOT NULL,
  expires_at DATETIME NOT NULL,
  created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
  FOREIGN KEY (user_id) REFERENCES users(id) ON DELETE CASCADE
)
`);

const resetTokenIndexes = [
  ['idx_password_reset_tokens_user', 'CREATE INDEX
idx_password_reset_tokens_user ON password_reset_tokens(user_id)'],
  ['idx_password_reset_tokens_expires', 'CREATE INDEX
idx_password_reset_tokens_expires ON password_reset_tokens(expires_at)']
];

for (const [indexName, createIndexQuery] of resetTokenIndexes) {
  const [resetIndexRows] = await conn.query(
    `SELECT INDEX_NAME
     FROM INFORMATION_SCHEMA.STATISTICS
     WHERE TABLE_SCHEMA = DATABASE()
       AND TABLE_NAME = 'password_reset_tokens'

```

```

        AND INDEX_NAME = ?`,
        [indexName]
    );

    if (!resetIndexRows.length) {
        console.log(`Adding password reset index ${indexName}`);
        await conn.query(createIndexQuery);
    }
}

const [reviewColumns] = await conn.query(
    `SELECT COLUMN_NAME
    FROM INFORMATION_SCHEMA.COLUMNS
    WHERE TABLE_SCHEMA = DATABASE()
    AND TABLE_NAME = 'reviews'
    ORDER BY ORDINAL_POSITION`
);

const columnNames = reviewColumns.map(c => c.COLUMN_NAME);
console.log('Reviews table columns:', columnNames);

if (!columnNames.includes('reviewer_id')) {
    console.log('Adding reviewer_id column to reviews table');
    await conn.query('ALTER TABLE reviews ADD COLUMN reviewer_id INT
NULL AFTER id');
}

if (!columnNames.includes('seller_id')) {
    console.log('Adding seller_id column to reviews table');
    await conn.query('ALTER TABLE reviews ADD COLUMN seller_id INT
NULL AFTER reviewer_id');
}

if (!columnNames.includes('car_id')) {
    console.log('Adding car_id column to reviews table');
    await conn.query('ALTER TABLE reviews ADD COLUMN car_id INT NULL
AFTER seller_id');
}

try {
    const [constraints] = await conn.query(
        `SELECT CONSTRAINT_NAME
        FROM INFORMATION_SCHEMA.TABLE_CONSTRAINTS
        WHERE TABLE_SCHEMA = DATABASE()
        AND TABLE_NAME = 'reviews'
        AND CONSTRAINT_TYPE = 'FOREIGN KEY'`
    );

    const constraintNames = constraints.map(c => c.CONSTRAINT_NAME);

    if (!constraintNames.some(c => c.includes('reviewer')) {
        await conn.query(
            'ALTER TABLE reviews ADD CONSTRAINT fk_reviews_reviewer
FOREIGN KEY (reviewer_id) REFERENCES users(id) ON DELETE CASCADE'
        );
    }

    if (!constraintNames.some(c => c.includes('seller')) {
        await conn.query(
            'ALTER TABLE reviews ADD CONSTRAINT fk_reviews_seller
FOREIGN KEY (seller_id) REFERENCES users(id) ON DELETE CASCADE'
        );
    }

    if (!constraintNames.some(c => c.includes('car')) {

```

```

        await conn.query(
            'ALTER TABLE reviews ADD CONSTRAINT fk_reviews_car
FOREIGN KEY (car_id) REFERENCES cars(id) ON DELETE CASCADE'
        );
    }
    } catch (e) {
        console.log('Foreign keys may already exist or there was an
issue:', e.message);
    }

    conn.release();
} catch (err) {
    console.error('Database error:', err);
}
})();

```

module.exports = pool;

Лістинг adminController.js

```

const db = require('../config/db');

exports.getUsers = async (req, res) => {
    try {
        const [users] = await db.query(
            `SELECT id, name, email, phone, role, rating, is_banned,
created_at FROM users ORDER BY created_at DESC`
        );
        res.json(users);
    } catch (err) {
        res.status(500).json(err);
    }
};

exports.deleteUser = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();

        const [cars] = await connection.query('SELECT id FROM cars WHERE
user_id = ?', [req.params.id]);
        const carIds = cars.map(car => car.id);

        if (carIds.length) {
            await connection.query('DELETE FROM car_images WHERE car_id IN
(?)', [carIds]);
            await connection.query('DELETE FROM favorites WHERE car_id IN
(?)', [carIds]);
            await connection.query('DELETE FROM reviews WHERE car_id IN (?)',
[carIds]);
            await connection.query('DELETE FROM views_history WHERE car_id IN
(?)', [carIds]);
            await connection.query('DELETE FROM messages WHERE car_id IN
(?)', [carIds]);
            await connection.query('DELETE FROM orders WHERE car_id IN (?)',
[carIds]);
            await connection.query('DELETE FROM cars WHERE id IN (?)',
[carIds]);
        }

        await connection.query('DELETE FROM favorites WHERE user_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM reviews WHERE reviewer_id = ? OR

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				105
Змн.	Арк.	№ докум.	Підпис	Дата		

```

seller_id = ?', [req.params.id, req.params.id]);
    await connection.query('DELETE FROM views_history WHERE user_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM messages WHERE sender_id = ? OR
receiver_id = ?', [req.params.id, req.params.id]);
    await connection.query('DELETE FROM orders WHERE buyer_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM users WHERE id = ?',
[req.params.id]);

    await connection.commit();
    res.json({ message: 'User deleted' });
  } catch (err) {
    await connection.rollback();
    res.status(500).json({ message: 'Failed to delete user' });
  } finally {
    connection.release();
  }
};

exports.getCars = async (req, res) => {
  try {
    const [cars] = await db.query(
      `SELECT
        cars.*,
        users.name AS seller_name,
        users.email AS seller_email,
        (
          SELECT image_url
          FROM car_images
          WHERE car_images.car_id = cars.id
          ORDER BY id ASC
          LIMIT 1
        ) AS first_image
      FROM cars
      JOIN users ON users.id = cars.user_id
      ORDER BY cars.created_at DESC`
    );

    res.json(cars);
  } catch (err) {
    res.status(500).json({ message: 'Failed to load cars' });
  }
};

exports.deleteCar = async (req, res) => {
  const connection = await db.getConnection();

  try {
    await connection.beginTransaction();
    await connection.query('DELETE FROM car_images WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM favorites WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM reviews WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM views_history WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM messages WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM orders WHERE car_id = ?',
[req.params.id]);
    await connection.query('DELETE FROM cars WHERE id = ?',

```

```

[req.params.id]);
    await connection.commit();

    res.json({ message: 'Car deleted' });
  } catch (err) {
    await connection.rollback();
    res.status(500).json({ message: 'Failed to delete car' });
  } finally {
    connection.release();
  }
};

```

Лістинг authController.js

```

const db = require('../config/db');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');
const crypto = require('crypto');

function hashResetToken(token) {
    return crypto.createHash('sha256').update(token).digest('hex');
}

exports.register = async (req, res) => {
    try {
        const { name, email, password, phone } = req.body;

        if (!name || !email || !password) {
            return res.status(400).json({ message: 'Name, email and password are required' });
        }

        const [existingUsers] = await db.query(
            'SELECT id FROM users WHERE email = ?',
            [email]
        );

        if (existingUsers.length) {
            return res.status(409).json({ message: 'User with this email already exists' });
        }

        const hashed = await bcrypt.hash(password, 10);

        await db.query(
            'INSERT INTO users (name, email, password, phone, role) VALUES (?, ?, ?, ?, ?)',
            [name, email, hashed, phone, 'user']
        );

        res.json({ message: 'Registered' });
    } catch (err) {
        console.error('Register error:', err);
        res.status(500).json({ message: 'Registration failed' });
    }
};

exports.login = async (req, res) => {
    try {
        const { email, password } = req.body;

        const [users] = await db.query(
            'SELECT * FROM users WHERE email = ?',

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				107
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        [email]
    );

    if (!users.length) {
        return res.status(404).json({ message: 'User not found' });
    }

    const user = users[0];

    const match = await bcrypt.compare(password, user.password);

    if (!match) {
        return res.status(400).json({ message: 'Wrong password' });
    }

    const token = jwt.sign(
        {
            id: user.id,
            email: user.email,
            role: user.role
        },
        process.env.JWT_SECRET,
        { expiresIn: '7d' }
    );

    res.json({
        token,
        user: {
            id: user.id,
            name: user.name,
            role: user.role
        }
    });

} catch (err) {
    res.status(500).json(err);
}

};

exports.requestPasswordReset = async (req, res) => {
    try {
        const { email } = req.body;

        if (!email) {
            return res.status(400).json({ message: 'Email is required' });
        }

        const [users] = await db.query('SELECT id, email FROM users WHERE email = ?', [email]);

        if (!users.length) {
            return res.status(404).json({ message: 'User with this email was not found' });
        }

        const user = users[0];
        const token = crypto.randomBytes(32).toString('hex');
        const tokenHash = hashResetToken(token);

        await db.query('DELETE FROM password_reset_tokens WHERE user_id = ?', [user.id]);
        await db.query(
            `INSERT INTO password_reset_tokens (user_id, token_hash,`

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				108
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    expires_at)
        VALUES (?, ?, DATE_ADD(NOW(), INTERVAL 1 HOUR))`,
        [user.id, tokenHash]
    );

    res.json({
        message: 'Password reset link created',
        resetLink: `reset-password.html?token=${token}`
    });
} catch (err) {
    console.error('Password reset request error:', err);
    res.status(500).json({ message: 'Failed to create password reset
link' });
}
};

exports.resetPassword = async (req, res) => {
    try {
        const { token, password } = req.body;

        if (!token || !password) {
            return res.status(400).json({ message: 'Token and password are
required' });
        }

        if (password.length < 6) {
            return res.status(400).json({ message: 'Password must contain at
least 6 characters' });
        }

        const tokenHash = hashResetToken(token);
        const [tokens] = await db.query(
            `SELECT password_reset_tokens.*, users.id AS user_id
            FROM password_reset_tokens
            JOIN users ON users.id = password_reset_tokens.user_id
            WHERE password_reset_tokens.token_hash = ?
            AND password_reset_tokens.expires_at > NOW()
            LIMIT 1`,
            [tokenHash]
        );

        if (!tokens.length) {
            return res.status(400).json({ message: 'Password reset link is
invalid or expired' });
        }

        const resetToken = tokens[0];
        const hashed = await bcrypt.hash(password, 10);

        await db.query('UPDATE users SET password = ? WHERE id = ?', [hashed,
resetToken.user_id]);
        await db.query('DELETE FROM password_reset_tokens WHERE user_id = ?',
[resetToken.user_id]);

        res.json({ message: 'Password updated' });
    } catch (err) {
        console.error('Password reset error:', err);
        res.status(500).json({ message: 'Failed to reset password' });
    }
};

exports.updateProfile = async (req, res) => {
    try {

```

```

    const { name, phone, password } = req.body;

    if (password) {
        const hashed = await bcrypt.hash(password, 10);
        await db.query('UPDATE users SET name = ?, phone = ?, password =
? WHERE id = ?', [name, phone, hashed, req.user.id]);
    } else {
        await db.query('UPDATE users SET name = ?, phone = ? WHERE id =
?', [name, phone, req.user.id]);
    }

    res.json({ message: 'Profile updated' });
} catch (err) {
    res.status(500).json(err);
}
};

exports.getProfile = async (req, res) => {
    try {
        const [users] = await db.query(
            'SELECT id, name, email, phone, role, rating, is_banned,
created_at FROM users WHERE id = ?',
            [req.user.id]
        );
        if (!users.length) return res.status(404).json({ message: 'User not
found' });
        res.json(users[0]);
    } catch (err) {
        res.status(500).json(err);
    }
};

```

Лістинг carController.js

```

const db = require('../config/db');
const fs = require('fs');
const path = require('path');

const uploadsDir = path.join(__dirname, '..', 'uploads');

function normalizeVinCode(value) {
    if (value === undefined || value === null || String(value).trim() === '')
    {
        return null;
    }

    const vinCode = String(value).trim().toUpperCase();

    if (!/^[A-HJ-NPR-Z0-9]{17}$/.test(vinCode)) {
        const error = new Error('VIN-код має містити 17 символів: цифри та
латинські літери без I, O, Q');
        error.statusCode = 400;
        throw error;
    }

    return vinCode;
}

function handleCarSaveError(res, err) {
    if (err.code === 'ER_DUP_ENTRY' && String(err.message ||
'').includes('idx_cars_vin_code')) {
        return res.status(409).json({ message: 'Авто з таким VIN-кодом вже
існує' });
    }
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				110
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (err.statusCode) {
      return res.status(err.statusCode).json({ message: err.message });
    }

    return res.status(500).json(err);
  }

exports.getAllCars = async (req, res) => {
  try {
    let query = `
      SELECT
        cars.*,
        users.name AS seller_name,
        users.rating AS seller_rating,
        (
          SELECT image_url
          FROM car_images
          WHERE car_images.car_id = cars.id
          ORDER BY id ASC
          LIMIT 1
        ) AS first_image
      FROM cars
      JOIN users ON users.id = cars.user_id
      WHERE cars.status = 'active'
    `;
    const params = [];

    if (req.query.brand) {
      query += ' AND cars.brand LIKE ?';
      params.push(`%${req.query.brand}%`);
    }

    if (req.query.model) {
      query += ' AND cars.model LIKE ?';
      params.push(`%${req.query.model}%`);
    }

    if (req.query.city) {
      query += ' AND cars.city LIKE ?';
      params.push(`%${req.query.city}%`);
    }

    if (req.query.minPrice) {
      query += ' AND cars.price >= ?';
      params.push(req.query.minPrice);
    }

    if (req.query.maxPrice) {
      query += ' AND cars.price <= ?';
      params.push(req.query.maxPrice);
    }

    if (req.query.yearFrom) {
      query += ' AND cars.year >= ?';
      params.push(req.query.yearFrom);
    }

    if (req.query.yearTo) {
      query += ' AND cars.year <= ?';
      params.push(req.query.yearTo);
    }
  }

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				111
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (req.query.minMileage) {
      query += ' AND cars.mileage >= ?';
      params.push(req.query.minMileage);
    }

    if (req.query.maxMileage) {
      query += ' AND cars.mileage <= ?';
      params.push(req.query.maxMileage);
    }

    if (req.query.fuel_type) {
      query += ' AND cars.fuel_type = ?';
      params.push(req.query.fuel_type);
    }

    if (req.query.transmission) {
      query += ' AND cars.transmission = ?';
      params.push(req.query.transmission);
    }

    const sortMap = {
      newest: 'cars.created_at DESC',
      price_asc: 'cars.price ASC',
      price_desc: 'cars.price DESC',
      year_desc: 'cars.year DESC',
      mileage_asc: 'cars.mileage ASC'
    };

    query += ` ORDER BY ${sortMap[req.query.sort]} || sortMap.newest`;

    const [cars] = await db.query(query, params);
    res.json(cars);
  } catch (err) {
    res.status(500).json(err);
  }
};

exports.getMyCars = async (req, res) => {
  try {
    const [cars] = await db.query(
      `SELECT
        cars.*,
        (
          SELECT COUNT(*)
          FROM views_history
          WHERE views_history.car_id = cars.id
        ) AS view_count,
        (
          SELECT COUNT(DISTINCT user_id)
          FROM views_history
          WHERE views_history.car_id = cars.id
            AND user_id IS NOT NULL
        ) AS unique_viewers,
        (
          SELECT MAX(viewed_at)
          FROM views_history
          WHERE views_history.car_id = cars.id
        ) AS last_viewed_at,
        (
          SELECT image_url
          FROM car_images
          WHERE car_images.car_id = cars.id
          ORDER BY id ASC
      `
    );
  }
};

```

```

        LIMIT 1
      ) AS first_image
    FROM cars
    WHERE cars.user_id = ?
    ORDER BY cars.created_at DESC`,
    [req.user.id]
  );

  res.json(cars);
} catch (err) {
  res.status(500).json({ message: 'Failed to load user cars' });
}
};

exports.getCarById = async (req, res) => {
  try {
    const [cars] = await db.query(
      `SELECT
        cars.*,
        users.name AS seller_name,
        users.email AS seller_email,
        users.phone AS seller_phone,
        users.rating AS seller_rating,
        users.created_at AS seller_created_at
      FROM cars
      JOIN users ON users.id = cars.user_id
      WHERE cars.id = ?`,
      [req.params.id]
    );

    if (!cars.length) {
      return res.status(404).json({ message: 'Not found' });
    }

    const [images] = await db.query(
      'SELECT * FROM car_images WHERE car_id = ?',
      [req.params.id]
    );

    const [priceHistory] = await db.query(
      `SELECT id, old_price, new_price, changed_at
      FROM car_price_history
      WHERE car_id = ?
      ORDER BY changed_at DESC, id DESC`,
      [req.params.id]
    );

    res.json({
      ...cars[0],
      images,
      price_history: priceHistory
    });
  } catch (err) {
    res.status(500).json(err);
  }
};

exports.createCar = async (req, res) => {
  try {
    const {
      brand,
      model,

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				113
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        vin_code,
        city,
        price,
        mileage,
        year,
        fuel_type,
        transmission,
        description
    } = req.body;
    const normalizedVinCode = normalizeVinCode(vin_code);

    const [result] = await db.query(
        `INSERT INTO cars
            (brand, model, vin_code, city, price, mileage, year,
fuel_type, transmission, description, user_id)
            VALUES (?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)`,
        [
            brand,
            model,
            normalizedVinCode,
            city,
            price,
            mileage || null,
            year || null,
            fuel_type,
            transmission,
            description || null,
            req.user.id
        ]
    );

    const carId = result.insertId;

    if (req.files && req.files.length > 0) {
        const values = req.files.map(file => [
            carId,
            file.filename
        ]);

        await db.query(
            `INSERT INTO car_images (car_id, image_url) VALUES ?`,
            [values]
        );
    }

    res.status(201).json({
        message: 'Car created',
        carId
    });

} catch (err) {
    handleCarSaveError(res, err);
}

};

exports.updateCar = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();

        const {
            brand,

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				114
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        model,
        vin_code,
        city,
        price,
        mileage,
        year,
        fuel_type,
        transmission,
        description
    } = req.body;
    const normalizedVinCode = normalizeVinCode(vin_code);

    const [cars] = await connection.query(
        'SELECT * FROM cars WHERE id = ? AND user_id = ?',
        [req.params.id, req.user.id]
    );

    if (!cars.length) {
        await connection.rollback();
        return res.status(403).json({ message: 'Not authorized to update
this car' });
    }

    const oldPrice = Number(cars[0].price);
    const newPrice = Number(price);

    if (oldPrice !== newPrice) {
        await connection.query(
            `INSERT INTO car_price_history (car_id, old_price, new_price)
            VALUES (?, ?, ?)\`,
            [req.params.id, oldPrice, newPrice]
        );
    }

    await connection.query(
        `UPDATE cars
        SET brand=?, model=?, vin_code=?, city=?, price=?, mileage=?,
        year=?, fuel_type=?, transmission=?, description=?
        WHERE id=? AND user_id=?`,
        [
            brand,
            model,
            normalizedVinCode,
            city,
            price,
            mileage || null,
            year || null,
            fuel_type || null,
            transmission || null,
            description || null,
            req.params.id,
            req.user.id
        ]
    );

    if (req.files && req.files.length > 0) {
        const imageValues = req.files.map(file => [
            req.params.id,
            file.filename
        ]);

        await connection.query(
            'INSERT INTO car_images (car_id, image_url) VALUES ?',

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				115
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        [imageValues]
    );
}

await connection.commit();
res.json({ message: 'Updated' });

} catch (err) {
    await connection.rollback();
    handleCarSaveError(res, err);
} finally {
    connection.release();
}
};

exports.deleteCarImage = async (req, res) => {
    try {
        const [images] = await db.query(
            `SELECT car_images.*, cars.user_id
            FROM car_images
            JOIN cars ON cars.id = car_images.car_id
            WHERE car_images.id = ?
            AND car_images.car_id = ?`,
            [req.params.imageId, req.params.id]
        );

        if (!images.length) {
            return res.status(404).json({ message: 'Image not found' });
        }

        if (Number(images[0].user_id) !== Number(req.user.id)) {
            return res.status(403).json({ message: 'Not authorized to delete this image' });
        }

        await db.query('DELETE FROM car_images WHERE id = ?',
            [req.params.imageId]);

        const imagePath = path.resolve(uploadsDir, images[0].image_url ||
            '');
        const uploadsRoot = path.resolve(uploadsDir) + path.sep;

        if (images[0].image_url && imagePath.startsWith(uploadsRoot)) {
            fs.unlink(imagePath, err => {
                if (err && err.code !== 'ENOENT') {
                    console.error('Failed to delete image file:',
err.message);
                }
            });
        }

        res.json({ message: 'Image deleted' });
    } catch (err) {
        res.status(500).json({ message: 'Failed to delete image' });
    }
};

exports.deleteCar = async (req, res) => {
    try {
        // Check if the car belongs to the user
        const [cars] = await db.query(
            'SELECT * FROM cars WHERE id = ? AND user_id = ?',
            [req.params.id, req.user.id]
        );
    }
};

```

```

    );

    if (!cars.length) {
        return res.status(403).json({ message: 'Not authorized to delete
this car' });
    }

    await db.query('DELETE FROM cars WHERE id=? AND user_id=?',
[req.params.id, req.user.id]);

    res.json({ message: 'Deleted' });

} catch (err) {
    res.status(500).json(err);
}
};

exports.adminDeleteCar = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();
        await connection.query('DELETE FROM car_images WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM favorites WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM reviews WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM views_history WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM messages WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM orders WHERE car_id = ?',
[req.params.id]);
        await connection.query('DELETE FROM cars WHERE id = ?',
[req.params.id]);
        await connection.commit();

        res.json({ message: 'Car deleted' });
    } catch (err) {
        await connection.rollback();
        res.status(500).json({ message: 'Failed to delete car' });
    } finally {
        connection.release();
    }
};

```

Лістинг favoriteController.js

```

const db = require('../config/db');

exports.addFavorite = async (req, res) => {
    try {
        const { car_id } = req.body;

        await db.query('INSERT IGNORE INTO favorites (user_id, car_id) VALUES
(?, ?)', [req.user.id, car_id]);

        res.json({ message: 'Added to favorites' });
    } catch (err) {
        res.status(500).json(err);
    }
};

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				117
Змн.	Арк.	№ докум.	Підпис	Дата		

```

exports.getFavorites = async (req, res) => {
  try {
    const [cars] = await db.query(
      `SELECT
        cars.*,
        (
          SELECT image_url
          FROM car_images
          WHERE car_images.car_id = cars.id
          ORDER BY id ASC
          LIMIT 1
        ) AS first_image
      FROM favorites
      JOIN cars ON cars.id = favorites.car_id
      WHERE favorites.user_id = ?`,
      [req.user.id]
    );

    res.json(cars);
  } catch (err) {
    res.status(500).json(err);
  }
};

exports.removeFavorite = async (req, res) => {
  try {
    await db.query(
      'DELETE FROM favorites WHERE user_id = ? AND car_id = ?',
      [req.user.id, req.params.carId]
    );

    res.json({ message: 'Removed from favorites' });
  } catch (err) {
    res.status(500).json({ message: 'Favorite remove failed' });
  }
};

```

Лістинг historyController.js

```

const db = require('../config/db');

exports.addHistory = async (req, res) => {
  try {
    const { car_id } = req.body;
    const userId = req.user ? req.user.id : null;

    if (!car_id) {
      return res.status(400).json({ message: 'Car ID is required' });
    }

    const [cars] = await db.query('SELECT id, user_id FROM cars WHERE id = ?', [car_id]);

    if (!cars.length) {
      return res.status(404).json({ message: 'Car not found' });
    }

    if (userId && Number(cars[0].user_id) === Number(userId)) {
      return res.json({ message: 'Owner view ignored' });
    }

    await db.query('INSERT INTO views_history (user_id, car_id) VALUES (?, ?)', [userId, car_id]);
  }
};

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				118
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        res.status(201).json({ message: 'History added' });
    } catch (err) {
        res.status(500).json(err);
    }
};

exports.getHistory = async (req, res) => {
    try {
        const [data] = await db.query(
            `SELECT cars.*,
                (SELECT image_url FROM car_images WHERE car_images.car_id
= cars.id ORDER BY id ASC LIMIT 1) AS first_image,
                views_history.viewed_at
            FROM views_history
            JOIN cars ON cars.id = views_history.car_id
            WHERE views_history.user_id = ?
            ORDER BY views_history.viewed_at DESC`,
            [req.user.id]
        );

        res.json(data);
    } catch (err) {
        res.status(500).json(err);
    }
};

```

Лістинг messageController.js

```

const db = require('../config/db');

exports.sendMessage = async (req, res) => {
    try {
        const { receiver_id, car_id, message } = req.body;
        const senderId = req.user.id;

        if (!receiver_id || !message) {
            return res.status(400).json({ message: 'Receiver ID and message
are required' });
        }

        const [receivers] = await db.query('SELECT id FROM users WHERE id =
?', [receiver_id]);
        if (!receivers.length) {
            return res.status(404).json({ message: 'Receiver not found' });
        }

        let car = null;

        if (car_id) {
            const [cars] = await db.query('SELECT id, user_id FROM cars WHERE
id = ?', [car_id]);
            if (!cars.length) {
                return res.status(404).json({ message: 'Car not found' });
            }

            car = cars[0];

            if (senderId !== car.user_id && Number(receiver_id) !==
car.user_id) {
                return res.status(403).json({ message: 'Car conversations
must include the car owner' });
            }
        }
    }
};

```

```

    const [result] = await db.query(
      'INSERT INTO messages (sender_id, receiver_id, car_id, message,
is_read) VALUES (?, ?, ?, ?, ?)',
      [senderId, receiver_id, car_id || null, message, false]
    );

    res.status(201).json({
      message: 'Message sent',
      messageId: result.insertId
    });
  } catch (err) {
    console.error('Message send error:', err);
    res.status(500).json({ message: 'Failed to send message' });
  }
};

exports.getSellerConversations = async (req, res) => {
  try {
    const userId = req.user.id;

    const [conversations] = await db.query(
      `SELECT latest.car_id,
latest.other_user_id,
cars.brand,
cars.model,
users.name AS other_user_name,
users.email AS other_user_email,
messages.message AS last_message,
messages.created_at AS last_message_at,
COALESCE(unread.unread_count, 0) AS unread_count
FROM (
  SELECT grouped.car_id,
grouped.other_user_id,
MAX(grouped.message_id) AS last_message_id
FROM (
  SELECT messages.id AS message_id,
messages.car_id,
CASE
  WHEN messages.sender_id = ? THEN
messages.receiver_id
  ELSE messages.sender_id
END AS other_user_id
FROM messages
JOIN cars ON cars.id = messages.car_id
WHERE cars.user_id = ?
AND messages.car_id IS NOT NULL
AND (messages.sender_id = ? OR messages.receiver_id =
?)

) AS grouped
GROUP BY grouped.car_id, grouped.other_user_id
) AS latest
JOIN messages ON messages.id = latest.last_message_id
JOIN cars ON cars.id = latest.car_id
JOIN users ON users.id = latest.other_user_id
LEFT JOIN (
  SELECT messages.car_id,
messages.sender_id AS other_user_id,
COUNT(*) AS unread_count
FROM messages
JOIN cars ON cars.id = messages.car_id
WHERE cars.user_id = ?
AND messages.receiver_id = ?
AND messages.is_read = false

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				120
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        GROUP BY messages.car_id, messages.sender_id
    ) AS unread
    ON unread.car_id = latest.car_id
    AND unread.other_user_id = latest.other_user_id
    ORDER BY messages.created_at DESC`,
    [userId, userId, userId, userId, userId, userId]
);

res.json(conversations);
} catch (err) {
    console.error('Get seller conversations error:', err);
    res.status(500).json({ message: 'Failed to fetch conversations' });
}
};

exports.getMessages = async (req, res) => {
    try {
        const userId = req.user.id;
        const type = req.query.type; // 'sent', 'received', or 'all'

        let query = `
            SELECT messages.*,
                   sender.name AS sender_name,
                   sender.email AS sender_email,
                   receiver.name AS receiver_name,
                   receiver.email AS receiver_email,
                   cars.brand,
                   cars.model
            FROM messages
            LEFT JOIN users AS sender ON sender.id = messages.sender_id
            LEFT JOIN users AS receiver ON receiver.id = messages.receiver_id
            LEFT JOIN cars ON cars.id = messages.car_id
        `;

        if (type === 'sent') {
            query += ` WHERE messages.sender_id = ?`;
        } else if (type === 'received') {
            query += ` WHERE messages.receiver_id = ?`;
        } else {
            query += ` WHERE (messages.sender_id = ? OR messages.receiver_id
= ?)`;
        }

        query += ` ORDER BY messages.created_at DESC`;

        const params = (type === 'sent' || type === 'received') ? [userId] :
[userId, userId];
        const [messages] = await db.query(query, params);

        res.json(messages);
    } catch (err) {
        console.error('Get messages error:', err);
        res.status(500).json({ message: 'Failed to fetch messages' });
    }
};

exports.markAsRead = async (req, res) => {
    try {
        const messageId = req.params.id;
        const userId = req.user.id;

        const [messages] = await db.query(
            `SELECT id FROM messages WHERE id = ? AND receiver_id = ?`,

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				121
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        [messageId, userId]
    );

    if (!messages.length) {
        return res.status(403).json({ message: 'Not authorized to mark
this message' });
    }

    await db.query('UPDATE messages SET is_read = true WHERE id = ?',
[messageId]);

    res.json({ message: 'Message marked as read' });
} catch (err) {
    console.error('Mark as read error:', err);
    res.status(500).json({ message: 'Failed to mark message as read' });
}
};

exports.deleteMessage = async (req, res) => {
    try {
        const messageId = req.params.id;
        const userId = req.user.id;

        // Check if message belongs to the user
        const [messages] = await db.query(
            'SELECT id FROM messages WHERE id = ? AND (sender_id = ? OR
receiver_id = ?)',
            [messageId, userId, userId]
        );

        if (!messages.length) {
            return res.status(403).json({ message: 'Not authorized to delete
this message' });
        }

        await db.query('DELETE FROM messages WHERE id = ?', [messageId]);

        res.json({ message: 'Message deleted' });
    } catch (err) {
        console.error('Delete message error:', err);
        res.status(500).json({ message: 'Failed to delete message' });
    }
};

exports.getConversation = async (req, res) => {
    try {
        const userId = req.user.id;
        const otherUserId = req.params.userId;
        const carId = req.query.carId;

        let query = `
            SELECT messages.*,
                   sender.name AS sender_name,
                   receiver.name AS receiver_name,
                   cars.brand,
                   cars.model
            FROM messages
            LEFT JOIN users AS sender ON sender.id = messages.sender_id
            LEFT JOIN users AS receiver ON receiver.id = messages.receiver_id
            LEFT JOIN cars ON cars.id = messages.car_id
            WHERE (
                (messages.sender_id = ? AND messages.receiver_id = ?) OR
                (messages.sender_id = ? AND messages.receiver_id = ?)

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				122
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        )
    `;

    const params = [userId, otherUserId, otherUserId, userId];

    if (carId) {
        query += ` AND messages.car_id = ?`;
        params.push(carId);
    }

    query += ` ORDER BY messages.created_at ASC`;

    const [messages] = await db.query(query, params);

    res.json(messages);
} catch (err) {
    console.error('Get conversation error:', err);
    res.status(500).json({ message: 'Failed to fetch conversation' });
}
};

```

Лістинг orderController.js

```

const db = require('../config/db');

exports.createOrder = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();

        const { car_id } = req.body;

        const [cars] = await connection.query('SELECT id, user_id, status
FROM cars WHERE id = ?', [car_id]);
        if (!cars.length) {
            await connection.rollback();
            return res.status(404).json({ message: 'Car not found' });
        }

        if (cars[0].user_id === req.user.id) {
            await connection.rollback();
            return res.status(400).json({ message: 'You cannot order your own
car' });
        }

        if (cars[0].status === 'sold') {
            await connection.rollback();
            return res.status(400).json({ message: 'Car is already sold' });
        }

        const [existingOrders] = await connection.query(
            'SELECT id FROM orders WHERE buyer_id = ? AND car_id = ? AND
status = ?',
            [req.user.id, car_id, 'pending']
        );

        if (existingOrders.length) {
            await connection.rollback();
            return res.status(400).json({ message: 'You already have a
pending order for this car' });
        }

        const [result] = await connection.query(

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				123
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        'INSERT INTO orders (buyer_id, car_id, status) VALUES (?, ?, ?)',
        [req.user.id, car_id, 'pending']
    );

    await connection.commit();

    res.json({
        message: 'Order created and waiting for seller approval',
        orderId: result.insertId
    });
} catch (err) {
    await connection.rollback();
    console.error('Order error:', err);
    res.status(500).json(err);
} finally {
    connection.release();
}
};

exports.getOrders = async (req, res) => {
    try {
        const [orders] = await db.query(
            `SELECT orders.*,
            orders.status AS order_status,
            cars.brand,
            cars.model,
            cars.price,
            cars.city,
            cars.status AS car_status,
            (SELECT image_url FROM car_images WHERE car_images.car_id
= cars.id ORDER BY id ASC LIMIT 1) AS first_image,
            users.name AS seller_name,
            users.email AS seller_email
            FROM orders
            JOIN cars ON cars.id = orders.car_id
            JOIN users ON users.id = cars.user_id
            WHERE orders.buyer_id = ?
            ORDER BY orders.created_at DESC`,
            [req.user.id]
        );

        res.json(orders);
    } catch (err) {
        res.status(500).json(err);
    }
};

exports.getSellerOrders = async (req, res) => {
    try {
        const userId = req.user.id;

        const [orders] = await db.query(
            `SELECT orders.*,
            orders.status AS order_status,
            cars.id as car_id,
            cars.brand,
            cars.model,
            cars.price,
            cars.city,
            cars.status AS car_status,
            (SELECT image_url FROM car_images WHERE car_images.car_id
= cars.id ORDER BY id ASC LIMIT 1) AS first_image,
            buyers.name AS buyer_name,

```

```

        buyers.email AS buyer_email,
        buyers.phone AS buyer_phone
    FROM orders
    JOIN cars ON cars.id = orders.car_id
    JOIN users AS buyers ON buyers.id = orders.buyer_id
    WHERE cars.user_id = ?
    ORDER BY orders.created_at DESC`,
    [userId]
);

res.json(orders);
} catch (err) {
    console.error('Get seller orders error:', err);
    res.status(500).json(err);
}
};

exports.approveOrder = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();

        const orderId = req.params.id;
        const userId = req.user.id;

        const [orders] = await connection.query(
            `SELECT orders.*, cars.user_id
            FROM orders
            JOIN cars ON cars.id = orders.car_id
            WHERE orders.id = ?`,
            [orderId]
        );

        if (!orders.length) {
            await connection.rollback();
            return res.status(404).json({ message: 'Order not found' });
        }

        if (orders[0].user_id !== userId) {
            await connection.rollback();
            return res.status(403).json({ message: 'Not authorized to approve
this order' });
        }

        if (orders[0].status !== 'pending') {
            await connection.rollback();
            return res.status(400).json({ message: 'Only pending orders can
be approved' });
        }

        await connection.query(
            'UPDATE orders SET status = ? WHERE id = ?',
            ['approved', orderId]
        );

        await connection.query(
            'UPDATE cars SET status = ? WHERE id = ?',
            ['sold', orders[0].car_id]
        );

        await connection.query(
            'UPDATE orders SET status = ? WHERE car_id = ? AND id <> ? AND

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				125
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    status = '?',
        ['rejected', orders[0].car_id, orderId, 'pending']
    );

    await connection.commit();
    res.json({ message: 'Order approved and car marked as sold' });
} catch (err) {
    await connection.rollback();
    console.error('Approve order error:', err);
    res.status(500).json({ message: 'Failed to approve order' });
} finally {
    connection.release();
}
};

exports.rejectOrder = async (req, res) => {
    const connection = await db.getConnection();

    try {
        await connection.beginTransaction();

        const orderId = req.params.id;
        const userId = req.user.id;

        const [orders] = await connection.query(
            `SELECT orders.*, cars.user_id
            FROM orders
            JOIN cars ON cars.id = orders.car_id
            WHERE orders.id = ?`,
            [orderId]
        );

        if (!orders.length) {
            await connection.rollback();
            return res.status(404).json({ message: 'Order not found' });
        }

        if (orders[0].user_id !== userId) {
            await connection.rollback();
            return res.status(403).json({ message: 'Not authorized to reject
this order' });
        }

        if (orders[0].status !== 'pending') {
            await connection.rollback();
            return res.status(400).json({ message: 'Only pending orders can
be rejected' });
        }

        await connection.query(
            'UPDATE orders SET status = ? WHERE id = ?',
            ['rejected', orderId]
        );

        await connection.commit();
        res.json({ message: 'Order rejected' });
    } catch (err) {
        await connection.rollback();
        console.error('Reject order error:', err);
        res.status(500).json({ message: 'Failed to reject order' });
    } finally {
        connection.release();
    }
};

```

```

    }
};

```

Лістинг reviewController.js

```

const db = require('../config/db');

exports.addReview = async (req, res) => {
  try {
    const { car_id, rating, comment } = req.body;
    const numericCarId = Number(car_id);
    const numericRating = Number(rating);

    if (!numericCarId || !numericRating || numericRating < 1 ||
    numericRating > 5) {
      return res.status(400).json({ message: 'Valid car_id and rating
from 1 to 5 are required' });
    }

    const [cars] = await db.query('SELECT id, user_id FROM cars WHERE id
= ?', [numericCarId]);

    if (!cars.length) {
      return res.status(404).json({ message: 'Car not found' });
    }

    const sellerId = cars[0].user_id;

    await db.query(
      'INSERT INTO reviews (reviewer_id, seller_id, car_id, rating,
comment) VALUES (?, ?, ?, ?, ?)',
      [req.user.id, sellerId, numericCarId, numericRating, comment ||
null]
    );

    const [allReviews] = await db.query(
      'SELECT AVG(rating) as avg_rating FROM reviews WHERE seller_id =
?',
      [sellerId]
    );

    if (allReviews.length > 0) {
      const avgRating = allReviews[0].avg_rating || 0;
      await db.query('UPDATE users SET rating = ? WHERE id = ?',
[avgRating, sellerId]);
    }

    res.json({ message: 'Review added' });
  } catch (err) {
    console.error('Review error:', err);
    res.status(500).json({ message: 'Review add failed' });
  }
};

exports.getReviews = async (req, res) => {
  try {
    const carId = req.params.carId;
    const [reviews] = await db.query(
      `SELECT reviews.*,
reviewer_user.name AS reviewer_name,
seller_user.name AS seller_name
FROM reviews

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				127
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        JOIN users AS reviewer_user ON reviewer_user.id =
reviews.reviewer_id
        JOIN users AS seller_user ON seller_user.id = reviews.seller_id
        WHERE reviews.car_id = ?
        ORDER BY reviews.created_at DESC`,
        [carId]
    );
    res.json(reviews && reviews.length > 0 ? reviews : []);
} catch (err) {
    console.error('Get reviews error:', err);
    res.status(500).json({ message: 'Error fetching reviews', error:
err.message });
}
};

exports.getSellerReviews = async (req, res) => {
    try {
        const sellerId = req.params.sellerId;

        const [reviews] = await db.query(
            `SELECT reviews.*,
                reviewer_user.name AS reviewer_name,
                cars.brand,
                cars.model
            FROM reviews
            JOIN users AS reviewer_user ON reviewer_user.id =
reviews.reviewer_id
            JOIN cars ON cars.id = reviews.car_id
            WHERE reviews.seller_id = ?
            ORDER BY reviews.created_at DESC`,
            [sellerId]
        );

        const avgRating = reviews.length > 0
            ? (reviews.reduce((sum, r) => sum + r.rating, 0) /
reviews.length).toFixed(1)
            : 0;

        res.json({
            reviews,
            averageRating: avgRating,
            totalReviews: reviews.length
        });
    } catch (err) {
        console.error('Get seller reviews error:', err);
        res.status(500).json({ message: 'Error fetching seller reviews' });
    }
};

```

Лістинг adminMiddleware.js

```

module.exports = (req, res, next) => {
    if (!req.user || req.user.role !== 'admin') {
        return res.status(403).json({ message: 'Access denied' });
    }
    next();
};

```

Лістинг authMiddleware.js

```

const jwt = require('jsonwebtoken');

module.exports = (req, res, next) => {
    const header = req.headers['authorization'];

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				128
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    if (!header) {
        return res.status(401).json({ message: 'No token' });
    }

    const token = header.split(' ')[1];

    try {
        const decoded = jwt.verify(token, process.env.JWT_SECRET);
        req.user = decoded;
        next();
    } catch (err) {
        return res.status(401).json({ message: 'Invalid token' });
    }
};

```

Лістинг errorMiddleware.js

```

module.exports = (err, req, res, next) => {
    console.error(err);
    res.status(500).json({ message: 'Server error' });
};

```

Лістинг uploadMiddleware.js

```

const multer = require('multer');
const path = require('path');
const fs = require('fs');

const uploadDir = path.join(__dirname, '..', 'uploads');
if (!fs.existsSync(uploadDir)) {
    fs.mkdirSync(uploadDir, { recursive: true });
}

const storage = multer.diskStorage({
    destination(req, file, cb) {
        cb(null, uploadDir);
    },

    filename(req, file, cb) {
        cb(null, Date.now() + '-' + file.originalname);
    }
});

module.exports = multer({
    storage
}).array('images', 10);

```

Лістинг validateCar.js

```

module.exports = (req, res, next) => {
    const { brand, model, city, price, mileage, year, fuel_type, transmission } = req.body;

    if (!brand || !model || !city || !price) {
        return res.status(400).json({ message: 'Missing required fields: brand, model, city, price' });
    }

    if (mileage && Number(mileage) < 0) {
        return res.status(400).json({ message: 'Mileage cannot be negative' });
    }

    if (year && (Number(year) < 1900 || Number(year) > new Date().getFullYear() + 1)) {
        return res.status(400).json({ message: 'Invalid production year' });
    }
};

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				129
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    if (!fuel_type || !transmission) {
        return res.status(400).json({ message: 'Missing required fields:
fuel_type, transmission' });
    }

    next();
};

```

Лістинг adminRoutes.js

```

const express = require('express');
const router = express.Router();
const adminController = require('../controllers/adminController');
const adminMiddleware = require('../middlewares/adminMiddleware');
const authMiddleware = require('../middlewares/authMiddleware');

router.get('/users', authMiddleware, adminMiddleware,
adminController.getUsers);
router.delete('/users/:id', authMiddleware, adminMiddleware,
adminController.deleteUser);
router.get('/cars', authMiddleware, adminMiddleware,
adminController.getCars);
router.delete('/cars/:id', authMiddleware, adminMiddleware,
adminController.deleteCar);

module.exports = router;

```

Лістинг authRoutes.js

```

const express = require('express');
const router = express.Router();
const authController = require('../controllers/authController');
const authMiddleware = require('../middlewares/authMiddleware');

router.post('/register', authController.register);
router.post('/login', authController.login);
router.post('/password/forgot', authController.requestPasswordReset);
router.post('/password/reset', authController.resetPassword);

router.put('/profile', authMiddleware, authController.updateProfile);

router.get('/profile', authMiddleware, authController.getProfile);

module.exports = router;

```

Лістинг carRoutes.js

```

const express = require('express');
const router = express.Router();

const carController = require('../controllers/carController');
const authMiddleware = require('../middlewares/authMiddleware');
const adminMiddleware = require('../middlewares/adminMiddleware');
const uploadMiddleware = require('../middlewares/uploadMiddleware');
const validateCar = require('../middlewares/validateCar');

router.get('/', carController.getAllCars);
router.get('/user/my', authMiddleware, carController.getMyCars);
router.get('/:id', carController.getCarById);

router.post(
    '/',
    authMiddleware,
    uploadMiddleware,

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				130
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        validateCar,
        carController.createCar
    );

    router.put('/:id', authMiddleware, uploadMiddleware,
carController.updateCar);

    router.delete('/admin/:id', authMiddleware, adminMiddleware,
carController.adminDeleteCar);
    router.delete('/:id/images/:imageId', authMiddleware,
carController.deleteCarImage);
    router.delete('/:id', authMiddleware, carController.deleteCar);

    module.exports = router;

```

Лістинг favoriteRoutes.js

```

const express = require('express');
const router = express.Router();
const favoriteController = require('../controllers/favoriteController');
const authMiddleware = require('../middlewares/authMiddleware');

router.post('/', authMiddleware, favoriteController.addFavorite);
router.get('/', authMiddleware, favoriteController.getFavorites);
router.delete('/:carId', authMiddleware, favoriteController.removeFavorite);

module.exports = router;

```

Лістинг historyRoutes.js

```

const express = require('express');
const router = express.Router();
const jwt = require('jsonwebtoken');
const historyController = require('../controllers/historyController');
const authMiddleware = require('../middlewares/authMiddleware');

function optionalAuthMiddleware(req, res, next) {
    const header = req.headers['authorization'];

    if (!header) {
        return next();
    }

    const token = header.split(' ')[1];

    try {
        req.user = jwt.verify(token, process.env.JWT_SECRET);
        next();
    } catch (err) {
        return res.status(401).json({ message: 'Invalid token' });
    }
}

router.post('/', optionalAuthMiddleware, historyController.addHistory);
router.get('/', authMiddleware, historyController.getHistory);

module.exports = router;

```

Лістинг messageRoutes.js

```

const express = require('express');
const router = express.Router();
const messageController = require('../controllers/messageController');
const authMiddleware = require('../middlewares/authMiddleware');

router.post('/', authMiddleware, messageController.sendMessage);

```

```

router.get('/', authMiddleware, messageController.getMessages);
router.get('/seller-conversations', authMiddleware,
messageController.getSellerConversations);
router.get('/conversation/:userId', authMiddleware,
messageController.getConversation);
router.put('/:id/read', authMiddleware, messageController.markAsRead);
router.delete('/:id', authMiddleware, messageController.deleteMessage);

```

```
module.exports = router;
```

Лістинг orderRoutes.js

```

const express = require('express');
const router = express.Router();
const orderController = require('../controllers/orderController');
const authMiddleware = require('../middlewares/authMiddleware');

router.post('/', authMiddleware, orderController.createOrder);
router.get('/', authMiddleware, orderController.getOrders);
router.get('/seller/my', authMiddleware, orderController.getSellerOrders);
router.put('/:id/approve', authMiddleware, orderController.approveOrder);
router.put('/:id/reject', authMiddleware, orderController.rejectOrder);

```

```
module.exports = router;
```

Лістинг reviewRoutes.js

```

const express = require('express');
const router = express.Router();
const reviewController = require('../controllers/reviewController');
const authMiddleware = require('../middlewares/authMiddleware');

router.post('/', authMiddleware, reviewController.addReview);
router.get('/:carId', reviewController.getReviews);
router.get('/seller/:sellerId', reviewController.getSellerReviews);

```

```
module.exports = router;
```

Лістинг app.js

```

const express = require('express');
const cors = require('cors');
const path = require('path');
require('dotenv').config();

const app = express();

app.use(cors());
app.use(express.json());
app.use(express.urlencoded({ extended: true }));

// ensure uploads folder exists
const fs = require('fs');
const uploadsPath = path.join(__dirname, 'uploads');
if (!fs.existsSync(uploadsPath)) {
  fs.mkdirSync(uploadsPath, { recursive: true });
}
app.use('/uploads', express.static(uploadsPath));
// Serve client static files under /client
app.use('/client', express.static(path.join(__dirname, '..', 'client')));

app.use('/api/auth', require('./routes/authRoutes'));
app.use('/api/cars', require('./routes/carRoutes'));
app.use('/api/favorites', require('./routes/favoriteRoutes'));
app.use('/api/history', require('./routes/historyRoutes'));
app.use('/api/orders', require('./routes/orderRoutes'));

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				132
Змн.	Арк.	№ докум.	Підпис	Дата		

```
app.use('/api/reviews', require('./routes/reviewRoutes'));
app.use('/api/messages', require('./routes/messageRoutes'));
app.use('/api/admin', require('./routes/adminRoutes'));
```

```
const PORT = process.env.PORT || 3000;
```

```
app.listen(PORT, () => {
  console.log(`Server running on port ${PORT}`);
});
```

Лістинг admin.js

```
function setAdminTab(activeTab) {
  document.getElementById('usersSection').classList.toggle('hidden',
activeTab !== 'users');
  document.getElementById('carsSection').classList.toggle('hidden',
activeTab !== 'cars');
  document.getElementById('usersTab').classList.toggle('active', activeTab
=== 'users');
  document.getElementById('carsTab').classList.toggle('active', activeTab
=== 'cars');
}

async function showUsers(event) {
  if (event) event.preventDefault();
  setAdminTab('users');
  document.getElementById('adminSubtitle').innerText = 'Керування
користувачами платформи.';
  await loadUsers();
}

async function showCars(event) {
  if (event) event.preventDefault();
  setAdminTab('cars');
  document.getElementById('adminSubtitle').innerText = 'Керування всіма
оголошеннями авто.';
  await loadAdminCars();
}

async function loadUsers() {
  try {
    const users = await request('/admin/users');
    const container = document.getElementById('usersSection');

    if (!users || users.length === 0) {
      container.innerHTML = '<p class="empty">Користувачів немає.</p>';
      return;
    }

    container.innerHTML = users.map(user => `
      <article class="card">
        <div class="card-body">
          <h3>${user.name || 'Без імені'}</h3>
          <div class="info-list">
            <div class="info-
row"><span>ID</span><strong>${user.id}</strong></div>
            <div class="info-
row"><span>Email</span><strong>${user.email}</strong></div>
            <div class="info-
row"><span>Роль</span><strong>${user.role}</strong></div>
            <div class="info-
row"><span>Телефон</span><strong>${user.phone || 'Не вказано'}</strong></div>
            <div class="info-
row"><span>Рейтинг</span><strong>${user.rating ? user.rating.toFixed(1) :
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				133
Змн.	Арк.	№ докум.	Підпис	Дата		

```

'Немає'}</strong></div>
        <div class="info-
row"><span>Заблокований</span><strong>${user.is_banned ? 'Так' :
'Hi'}</strong></div>
        <div class="info-
row"><span>Створено</span><strong>${user.created_at ? new
Date(user.created_at).toLocaleString('uk-UA') : 'Не вказано'}</strong></div>
        </div>
        <div class="actions">
            <button type="button"
onclick="deleteUser(${user.id})" class="btn-danger">Видалити</button>
        </div>
    </div>
</article>
`)).join('');
} catch (err) {
    document.getElementById('usersSection').innerHTML = `<p
class="empty">${err.message || 'Помилка при завантаженні
користувачів.'}</p>`;
}
}

async function loadAdminCars() {
    try {
        const cars = await request('/admin/cars');
        const container = document.getElementById('carsSection');

        if (!cars || cars.length === 0) {
            container.innerHTML = '<p class="empty">Оголошень немає.</p>';
            return;
        }

        container.innerHTML = cars.map(car => {
            const card = renderCarCard(car);
            return card.replace(
                '</div>\n            </article>',
                `<div class="info-list">
                    <div class="info-row"><span>ID
авто</span><strong>${car.id}</strong></div>
                    <div class="info-
row"><span>Продавець</span><strong>${car.seller_name || 'Не
вказано'}</strong></div>
                    <div class="info-
row"><span>Email</span><strong>${car.seller_email || 'Не
вказано'}</strong></div>
                    </div>
                    <div class="actions">
                        <button type="button"
onclick="deleteAdminCar(${car.id})" class="btn-danger">Видалити авто</button>
                    </div>
                </div>
            </article>`
            );
        }).join('');
    } catch (err) {
        document.getElementById('carsSection').innerHTML = `<p
class="empty">${err.message || 'Помилка при завантаженні авто.'}</p>`;
    }
}

async function deleteUser(id) {
    if (!confirm('Видалити цього користувача разом з усіма його
оголошеннями?')) {

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				134
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        return;
    }

    try {
        await request(`/admin/users/${id}`, 'DELETE');
        alert('Користувача видалено');
        loadUsers();
    } catch (err) {
        alert(err.message || 'Помилка при видаленні користувача');
    }
}

async function deleteAdminCar(id) {
    if (!confirm('Видалити це оголошення авто?')) {
        return;
    }

    try {
        await request(`/admin/cars/${id}`, 'DELETE');
        alert('Авто видалено');
        loadAdminCars();
    } catch (err) {
        alert(err.message || 'Помилка при видаленні авто');
    }
}

```

Лістинг api.js

```

const API_URL = 'http://127.0.0.1:3000/api';

function getToken() {
    return localStorage.getItem('token');
}

async function request(endpoint, method = 'GET', data = null, isForm = false)
{
    const headers = {};

    const token = getToken();

    if (token) {
        headers.Authorization = `Bearer ${token}`;
    }

    if (!isForm) {
        headers['Content-Type'] = 'application/json';
    }

    const response = await fetch(API_URL + endpoint, {
        method,
        headers,
        body: data
            ? (isForm ? data : JSON.stringify(data))
            : null
    });

    const text = await response.text();

    if (!response.ok) {
        console.error('API ERROR:', text);
        let message = text || 'Request failed';

        try {
            const errorData = JSON.parse(text);

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				135
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        message = errorData.message || message;
    } catch (e) {
    }

    throw new Error(message);
}

try {
    return JSON.parse(text);
} catch (e) {
    console.error('NOT JSON RESPONSE:', text);
    throw e;
}
}

```

Лістинг auth.js

```

async function register(e) {
    e.preventDefault();

    await request('/auth/register', 'POST', {
        name: document.getElementById('name').value,
        email: document.getElementById('email').value,
        password: document.getElementById('password').value,
        phone: document.getElementById('phone').value
    });

    alert('Реєстрація успішна');
    location.href = 'login.html';
}

async function login(e) {
    e.preventDefault();

    const res = await request('/auth/login', 'POST', {
        email: document.getElementById('email').value,
        password: document.getElementById('password').value
    });

    if (res.token) {
        localStorage.setItem('token', res.token);

        // Decode token to check role
        const payload = JSON.parse(atob(res.token.split('.')[1]));

        // Redirect to admin panel if admin, otherwise to index
        if (payload.role === 'admin') {
            window.location.href = 'admin.html';
        } else {
            window.location.href = 'index.html';
        }
    } else {
        alert(res.message);
    }
}

function showAuthMessage(id, message, isVisible = true) {
    const element = document.getElementById(id);

    if (!element) return;

    element.innerHTML = message;
    element.style.display = isVisible ? 'block' : 'none';
}

```

```

async function requestPasswordReset(e) {
    e.preventDefault();

    showAuthMessage('errorMessage', '', false);
    showAuthMessage('successMessage', '', false);

    try {
        const result = await request('/auth/password/forgot', 'POST', {
            email: document.getElementById('resetEmail').value
        });

        const resetUrl = new URL(result.resetLink,
window.location.href).href;

        showAuthMessage(
            'successMessage',
            `Посилання створено. Відкрийте його, щоб задати новий
пароль:<br><a href="${resetUrl}">${resetUrl}</a>`
        );
    } catch (err) {
        showAuthMessage('errorMessage', err.message || 'Не вдалося створити
посилання для відновлення паролю.');
```

```

    }
}

function getResetToken() {
    return new URLSearchParams(window.location.search).get('token');
}

function checkResetToken() {
    if (!document.getElementById('newPassword')) return;

    if (!getResetToken()) {
        showAuthMessage('errorMessage', 'Посилання для відновлення паролю
некоректне або не містить токен.');
```

```

        document.querySelector('button[type="submit"]').disabled = true;
    }
}

async function resetPassword(e) {
    e.preventDefault();

    showAuthMessage('errorMessage', '', false);
    showAuthMessage('successMessage', '', false);

    const password = document.getElementById('newPassword').value;
    const confirmPassword = document.getElementById('confirmPassword').value;

    if (password !== confirmPassword) {
        showAuthMessage('errorMessage', 'Паролі не співпадають.');
```

```

        return;
    }

    try {
        await request('/auth/password/reset', 'POST', {
            token: getResetToken(),
            password
        });

        showAuthMessage(
            'successMessage',
            'Пароль змінено. Тепер можна увійти з новим паролем.'
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				137
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    );
    e.target.reset();
  } catch (err) {
    showAuthMessage('errorMessage', err.message || 'Не вдалося змінити
    пароль.');
```

Лістинг cars.js

```

function imageUrl(fileName) {
  return fileName ? `http://127.0.0.1:3000/uploads/${fileName}` : '';
}

function formatPrice(value) {
  return Number(value || 0).toLocaleString('uk-UA', {
    style: 'currency',
    currency: 'USD',
    maximumFractionDigits: 0
  });
}

function requireLogin(message = 'Щоб виконати цю дію, потрібно увійти в
акаунт.') {
  if (getToken()) return true;
  alert(message);
  return false;
}

function renderCarCard(car) {
  const media = car.first_image
    ? `![${car.brand} ${car.model}](${imageUrl(car.first_image)})★ Рейтинг:
    ${car.seller_rating.toFixed(1)}</p>`
    : '';

  return `
  <article class="card car-card">
    ${media}
    <div class="card-body">
      <h3 class="car-title">${car.brand} ${car.model}</h3>
      <p class="car-meta">${car.city}</p>
      <p class="car-meta">${specs}</p>
      ${sellerRating}
      <p class="price">${formatPrice(car.price)}</p>
      <a href="car.html?id=${car.id}" class="btn btn-
      secondary">Детальніше</a>
    </div>
  </article>`
}
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				138
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    `;
}

async function loadCars(endpoint = '/cars') {
    const cars = await request(endpoint);
    const container = document.getElementById('cars');

    if (!container) return;

    if (!cars || cars.length === 0) {
        container.innerHTML = '<p class="empty">Оголошень поки немає.</p>';
        return;
    }

    container.innerHTML = cars.map(renderCarCard).join('');
}

function getFilterValue(id) {
    const element = document.getElementById(id);
    return element ? element.value.trim() : '';
}

async function filterCars(event) {
    if (event) {
        event.preventDefault();
    }

    const params = new URLSearchParams();
    const filters = {
        brand: getFilterValue('brandFilter'),
        model: getFilterValue('modelFilter'),
        city: getFilterValue('cityFilter'),
        minPrice: getFilterValue('minPrice'),
        maxPrice: getFilterValue('maxPrice'),
        yearFrom: getFilterValue('yearFrom'),
        yearTo: getFilterValue('yearTo'),
        minMileage: getFilterValue('minMileage'),
        maxMileage: getFilterValue('maxMileage'),
        fuel_type: getFilterValue('fuelTypeFilter'),
        transmission: getFilterValue('transmissionFilter'),
        sort: getFilterValue('sortFilter')
    };

    Object.entries(filters).forEach(([key, value]) => {
        if (value) {
            params.append(key, value);
        }
    });

    await loadCars(`/cars?${params.toString()}`);
}

async function resetFilters() {
    const toolbar = document.querySelector('.toolbar');

    if (toolbar) {
        toolbar.reset();
    }

    await loadCars();
}

async function createCar(e) {

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				139
Змн.	Арк.	№ докум.	Підпис	Дата		

```

e.preventDefault();

if (!requireLogin('Щоб додати авто, потрібно увійти в акаунт.')) {
    location.href = 'login.html';
    return;
}

const formData = new FormData();
formData.append('brand', document.getElementById('brand').value);
formData.append('model', document.getElementById('model').value);
formData.append('vin_code', document.getElementById('vin_code').value);
formData.append('city', document.getElementById('city').value);
formData.append('price', document.getElementById('price').value);
formData.append('mileage', document.getElementById('mileage').value);
formData.append('year', document.getElementById('year').value);
formData.append('fuel_type', document.getElementById('fuel_type').value);
formData.append('transmission',
document.getElementById('transmission').value);
formData.append('description',
document.getElementById('description').value);

for (const file of document.getElementById('images').files) {
    formData.append('images', file);
}

const response = await fetch('http://127.0.0.1:3000/api/cars', {
    method: 'POST',
    headers: {
        Authorization: 'Bearer ' + getToken()
    },
    body: formData
});

const data = await response.json();

if (response.ok) {
    alert('Авто додано');
    location.href = 'index.html';
} else {
    alert(data.message || 'Помилка додавання авто');
}
}

```

Лістинг favorites.js

```

async function addToFavorites(carId) {
    try {
        await request('/favorites', 'POST', { car_id: carId });
        alert('Додано в обране');
    } catch (err) {
        alert(err.message || 'Помилка при додаванні в обране');
    }
}

async function removeFromFavorites(carId) {
    await request(`/favorites/${carId}`, 'DELETE');
}

```

Лістинг layout.js

```

function updateHeader() {
    const token = getToken();
    const loginLink = document.getElementById('loginLink');
    const profileLink = document.getElementById('profileLink');
    const logoutLink = document.getElementById('logoutLink');
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				140
Змн.	Арк.	№ докум.	Підпис	Дата		

```

const addCarLink = document.getElementById('addCarLink');

if (loginLink) loginLink.classList.toggle('hidden', Boolean(token));
if (profileLink) profileLink.classList.toggle('hidden', !token);
if (logoutLink) logoutLink.classList.toggle('hidden', !token);

if (addCarLink) {
  addCarLink.addEventListener('click', event => {
    if (!token) {
      event.preventDefault();
      alert('Щоб додати авто, потрібно увійти в акаунт.');
```

Лістинг add-car.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Додати авто | AutoHub</title>
  <link rel="stylesheet" href="css/style.css">
</head>
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				141
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<body>
<header class="site-header">
  <nav class="nav">
    <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
    <div class="nav-links">
      <a href="index.html">Головна</a>
      <a href="add-car.html" id="addCarLink">Додати авто</a>
      <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
      <a href="login.html" id="loginLink">Увійти</a>
      <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
    </div>
  </nav>
</header>

<main class="page">
  <section class="page-title">
    <h1>Додати авто</h1>
    <p>Заповніть характеристики авто та додайте кілька фотографій. Перше
фото буде обкладинкою оголошення.</p>
  </section>

  <form class="form-panel" onsubmit="createCar(event)">
    <div class="form-grid">
      <label>Марка
        <input id="brand" required>
      </label>
      <label>Модель
        <input id="model" required>
      </label>
      <label>VIN-код
        <input id="vin_code" maxlength="17" pattern="[A-HJ-NPR-Za-hj-
npr-z0-9]{17}" placeholder="Наприклад WBAJA71070CD12345">
      </label>
      <label>Місто
        <input id="city" required>
      </label>
      <label>Ціна, USD
        <input id="price" type="number" min="0" required>
      </label>
      <label>Пробіг, км
        <input id="mileage" type="number" min="0" required>
      </label>
      <label>Рік виготовлення
        <input id="year" type="number" min="1900" max="2100"
required>
      </label>
      <label>Тип палива
        <select id="fuel_type" required>
          <option value="">Оберіть тип палива</option>
          <option value="Бензин">Бензин</option>
          <option value="Дизель">Дизель</option>
          <option value="Газ/Бензин">Газ/Бензин</option>
          <option value="Гібрид">Гібрид</option>
          <option value="Електро">Електро</option>
        </select>
      </label>
      <label>Тип трансмісії
        <select id="transmission" required>
          <option value="">Оберіть трансмісію</option>
          <option value="Механічна">Механічна</option>

```

		Поліщук І. А.			ДУ «Житомирська політехніка». 26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		142

```

        <option value="Автоматична">Автоматична</option>
        <option value="Роботизована">Роботизована</option>
        <option value="Варіатор">Варіатор</option>
    </select>
</label>
<label class="full">Опис авто
    <textarea id="description" required placeholder="Опишіть
стан, комплектацію та важливі деталі"></textarea>
</label>
<label class="full">Фотографії
    <input id="images" type="file" accept="image/*" multiple>
</label>
</div>
<div class="actions">
    <button type="submit">Опублікувати</button>
    <a href="index.html" class="btn btn-secondary">Назад</a>
</div>
</form>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Додавайте чесні оголошення з якісними фото</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/cars.js"></script>
</body>
</html>

```

Лістинг admin.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Адмін панель | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">
            <a href="index.html">Головна</a>
            <a href="#" id="usersTab"
onclick="showUsers(event)">Користувачі</a>
            <a href="#" id="carsTab" onclick="showCars(event)">Авто</a>
            <a href="#" id="logoutLink" onclick="logout()">Вийти</a>
        </div>
    </nav>
</header>

<main class="page">
    <section class="page-title">
        <h1>Адмін панель</h1>
        <p id="adminSubtitle">Керування користувачами платформи.</p>
    </section>

```

```

    <div id="usersSection" class="grid"></div>
    <div id="carsSection" class="grid hidden"></div>
</main>

<footer class="site-footer">
  <div class="footer-inner">
    <strong>AutoHub</strong>
    <span>Адміністративний доступ</span>
  </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/cars.js"></script>
<script src="js/admin.js"></script>
<script>
  function checkAdmin() {
    const token = getToken();
    if (!token) {
      location.href = 'login.html';
      return false;
    }

    const payload = JSON.parse(atob(token.split('.')[1]));
    if (payload.role !== 'admin') {
      alert('Доступ заборонено. Ви не адміністратор.');
```

Лістинг car.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Авто | AutoHub</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
  <nav class="nav">
    <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
    <div class="nav-links">
      <a href="index.html">Головна</a>
      <a href="add-car.html" id="addCarLink">Додати авто</a>
      <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
      <a href="login.html" id="loginLink">Увійти</a>
      <a href="#" id="logoutLink" class="hidden">
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				144
Змн.	Арк.	№ докум.	Підпис	Дата		

```

onclick="logout()">Вийти</a>
    </div>
</nav>
</header>

<main class="page">
    <div id="car"></div>

    <section class="mt-24">
        <div class="action-buttons">
            <button onclick="addToFavoritesButton()" id="favoriteBtn">♥
Додати в обране</button>
            <button onclick="buyCarButton()" id="buyBtn"
style="display:none;">🛒 Купити авто</button>
        </div>
    </section>

    <section class="chat-section" id="chatSection" style="display:none;">
        <h2>Чат з власником</h2>
        <div class="chat-messages" id="chatMessages"></div>
        <form class="chat-form" onsubmit="sendMessage(event)">
            <input type="text" id="messageInput" placeholder="Напишіть
повідомлення..." required>
            <button type="submit">Відправити</button>
        </form>
    </section>

    <section class="mt-24">
        <div class="page-title">
            <h2>Відгуки</h2>
        </div>
        <div id="reviews" class="grid"></div>

        <form class="form-panel mt-24" onsubmit="submitReview(event)">
            <div class="form-grid">
                <label>Оцінка
                    <input type="number" id="rating" min="1" max="5"
placeholder="1-5" required>
                </label>
                <label class="full">Коментар
                    <textarea id="comment" placeholder="Ваш
коментар"></textarea>
                </label>
            </div>
            <div class="actions">
                <button type="submit">Надіслати відгук</button>
            </div>
        </form>
    </section>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Перевіряйте деталі перед покупкою</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/cars.js"></script>
<script src="js/favorites.js"></script>
</script>

```

```

let currentCarId;
let currentSellerId;
let currentImageIndex = 0;
let carImages = [];
let lightboxZoom = 1;

function emptyValue(value) {
    return value === null || value === undefined || value === '' ? 'Не
вказано' : value;
}

function formatDate(date) {
    return date ? new Date(date).toLocaleString('uk-UA') : 'Не вказано';
}

function escapeHtml(value) {
    return String(value ?? '').replace(/[\<>"']/g, char => ({
        '&': '&amp;',
        '<': '&lt;',
        '>': '&gt;',
        '"': '&quot;',
        "'": '&#39;';
    }[char]));
}

function getStatusBadge(status) {
    if (status === 'sold') {
        return '<span class="status-badge status-sold">● ПРОДАНО</span>';
    }
    return '<span class="status-badge status-active">□ АКТИВНО</span>';
}

async function addToFavoritesButton() {
    if (!requireLogin('Щоб додати авто в обране, потрібно увійти в акаунт.'))
    {
        return;
    }

    await addToFavorites(currentCarId);
}

async function buyCarButton() {
    if (!requireLogin('Щоб купити авто, потрібно увійти в акаунт.')) {
        return;
    }

    const confirmBuy = confirm('Ви впевнені, що хочете створити замовлення на
це авто?\n\nВласник авто повинен буде підтвердити продаж.');
```

```

    if (!confirmBuy) return;

    try {
        const result = await request('/orders', 'POST', { car_id:
currentCarId });
        alert('✓ Замовлення створено!\n\nВласник авто отримає сповіщення і
підтвердить продаж. Слідкуйте за статусом в профілі.');
```

```

    } catch (err) {
        alert(err.message || 'Не вдалося створити замовлення');
    }
}

function updateCarousel() {
    const mainImage = document.getElementById('carouselMain');
    if (carImages.length > 0) {

```

```

        mainImage.innerHTML = ``;
    }

    // Update thumbnails
    document.querySelectorAll('.carousel-thumbnail').forEach((thumb, index)
=> {
        thumb.classList.toggle('active', index === currentImageIndex);
    });
}

function previousImage() {
    if (carImages.length === 0) return;
    currentImageIndex = (currentImageIndex - 1 + carImages.length) %
carImages.length;
    updateCarousel();
}

function nextImage() {
    if (carImages.length === 0) return;
    currentImageIndex = (currentImageIndex + 1) % carImages.length;
    updateCarousel();
}

function selectImage(index) {
    currentImageIndex = index;
    updateCarousel();
}

function updateLightbox() {
    const image = document.getElementById('lightboxImage');
    const counter = document.getElementById('lightboxCounter');

    if (!image || carImages.length === 0) return;

    image.src = imageUrl(carImages[currentImageIndex].image_url);
    image.alt = `Фото авто ${currentImageIndex + 1}`;
    image.style.transform = `scale(${lightboxZoom})`;

    if (counter) {
        counter.textContent = `${currentImageIndex + 1} / ${carImages.length}
· ${Math.round(lightboxZoom * 100)}%`;
    }
}

function resetLightboxZoom() {
    lightboxZoom = 1;
    updateLightbox();
}

function setLightboxZoom(value) {
    lightboxZoom = Math.min(3, Math.max(0.5, value));
    updateLightbox();
}

function zoomLightboxImage(delta, event) {
    if (event) {
        event.preventDefault();
        event.stopPropagation();
    }

    setLightboxZoom(lightboxZoom + delta);
}

```

```

}

function handleLightboxWheel(event) {
    const lightbox = document.getElementById('imageLightbox');
    const isOpen = lightbox && lightbox.classList.contains('active');

    if (!isOpen) return;

    zoomLightboxImage(event.deltaY < 0 ? 0.1 : -0.1, event);
}

function openImageLightbox(index) {
    if (carImages.length === 0) return;

    currentIndex = index;
    lightboxZoom = 1;
    updateCarousel();

    const lightbox = document.getElementById('imageLightbox');
    if (!lightbox) return;

    lightbox.classList.add('active');
    document.body.classList.add('no-scroll');
    updateLightbox();
}

function closeImageLightbox() {
    const lightbox = document.getElementById('imageLightbox');
    if (!lightbox) return;

    lightbox.classList.remove('active');
    document.body.classList.remove('no-scroll');
}

function previousLightboxImage(event) {
    if (event) event.stopPropagation();
    lightboxZoom = 1;
    previousImage();
    updateLightbox();
}

function nextLightboxImage(event) {
    if (event) event.stopPropagation();
    lightboxZoom = 1;
    nextImage();
    updateLightbox();
}

function handleLightboxKeydown(event) {
    const lightbox = document.getElementById('imageLightbox');
    const isOpen = lightbox && lightbox.classList.contains('active');

    if (!isOpen) return;

    if (event.key === 'Escape') {
        closeImageLightbox();
    }

    if (event.key === 'ArrowLeft') {
        previousLightboxImage(event);
    }

    if (event.key === 'ArrowRight') {

```

```

        nextLightboxImage(event);
    }

    if (event.key === '+' || event.key === '=') {
        zoomLightboxImage(0.1, event);
    }

    if (event.key === '-') {
        zoomLightboxImage(-0.1, event);
    }

    if (event.key === '0') {
        resetLightboxZoom();
    }
}

async function loadMessages() {
    try {
        const messages = await
request(`/messages/conversation/${currentSellerId}?carId=${currentCarId}`);
        const container = document.getElementById('chatMessages');

        if (!messages || messages.length === 0) {
            container.innerHTML = '<p style="text-align:center;color:#999;">Повідомлень поки немає</p>';
            return;
        }

        container.innerHTML = messages.map(msg => `
            <div class="chat-message ${msg.sender_id === getCurrentUserId() ?
'sent' : 'received'}">
                <div class="chat-sender">${escapeHtml(msg.sender_name ||
'Ви')}</div>
                <div>${escapeHtml(msg.message)}</div>
                <small>${formatDate(msg.created_at)}</small>
            </div>
        `).join('');

        container.scrollTop = container.scrollHeight;
    } catch (err) {
        console.log('Could not load messages:', err.message);
    }
}

async function sendMessage(e) {
    e.preventDefault();

    const message = document.getElementById('messageInput').value;
    if (!message.trim()) return;

    try {
        await request('/messages', 'POST', {
            receiver_id: currentSellerId,
            car_id: currentCarId,
            message: message
        });

        document.getElementById('messageInput').value = '';
        await loadMessages();
    } catch (err) {
        alert(err.message || 'Не вдалося відправити повідомлення');
    }
}

```

```

function getCurrentUserId() {
  try {
    const token = getToken();
    if (!token) return null;
    const payload = JSON.parse(atob(token.split('.')[1]));
    return payload.id;
  } catch (e) {
    return null;
  }
}

async function loadReviews() {
  const container = document.getElementById('reviews');

  try {
    const reviews = await request(`/reviews/${currentCarId}`);

    if (!reviews || reviews.length === 0) {
      container.innerHTML = '<p class="empty">Відгуків поки немає.</p>';
      return;
    }

    container.innerHTML = reviews.map(review => `
      <article class="card">
        <div class="card-body">
          <strong>Оцінка: ${review.rating}/5</strong>
          <p>${review.comment || 'Немає коментару'}</p>
          <small class="muted">${review.reviewer_name ||
review.reviewer_id || 'Анонімно'} · ${formatDate(review.created_at)}</small>
        </div>
      </article>
    `).join('');
  } catch (err) {
    container.innerHTML = `<p class="empty">${err.message || 'Не вдалося завантажити відгуки.'}</p>`;
  }
}

async function submitReview(e) {
  e.preventDefault();

  if (!requireLogin('Щоб залишити відгук, потрібно увійти в акаунт.')) {
    return;
  }

  try {
    await request('/reviews', 'POST', {
      car_id: Number(currentCarId),
      rating: Number(document.getElementById('rating').value),
      comment: document.getElementById('comment').value
    });

    alert('Відгук додано');
    e.target.reset();
    loadReviews();
  } catch (err) {
    alert(err.message || 'Помилка при додаванні відгуку');
  }
}

async function loadCarDetails() {

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				150
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    currentCarId = new URLSearchParams(window.location.search).get('id');
    const car = await request(`/cars/${currentCarId}`);
    const container = document.getElementById('car');

    carImages = car.images && car.images.length ? car.images : [];
    currentIndex = 0;

    const currentUserId = getCurrentUserId();
    const isOwner = Number(currentUserId) === Number(car.user_id);
    const isSold = car.status === 'sold';

    currentSellerId = car.user_id;

    if (!isOwner) {
        request('/history', 'POST', { car_id: currentCarId }).catch(err => {
            console.log('Could not record view:', err.message);
        });
    }

    document.getElementById('buyBtn').style.display = (!isOwner && !isSold) ?
    'block' : 'none';
    document.getElementById('chatSection').style.display = (!isOwner &&
    getToken()) ? 'block' : 'none';

    if (!isOwner && getToken()) {
        loadMessages();
        setInterval(loadMessages, 3000);
    }

    const galleryHtml = carImages.length > 0 ? `
        <div class="carousel-container">
            <div class="carousel-main" id="carouselMain">
                
            </div>
            ${carImages.length > 1 ? `
                <div class="carousel-controls">
                    <button class="carousel-btn"
onclick="previousImage()"><</button>
                    <button class="carousel-btn"
onclick="nextImage()">>>/button>
                </div>
                ` : ''}
            <div class="carousel-thumbnails">
                ${carImages.map((img, idx) => `
                    <div class="carousel-thumbnail ${idx === 0 ? 'active' :
''}" onclick="selectImage(${idx})">
                        
                    </div>
                `).join('')}
            </div>
            <div class="image-lightbox" id="imageLightbox"
onclick="closeImageLightbox()">
                <button type="button" class="lightbox-close"
onclick="closeImageLightbox()">×</button>
                ${carImages.length > 1 ? `
                    <button type="button" class="lightbox-nav lightbox-prev"
onclick="previousLightboxImage(event)"><</button>
                    <button type="button" class="lightbox-nav lightbox-next"
onclick="nextLightboxImage(event)">>>/button>
                ` : ''}
            </div>
        </div>
    ` : '';

```

```

        ` : ''}
        <div class="lightbox-zoom-controls"
onclick="event.stopPropagation()">
            <button type="button" onclick="zoomLightboxImage(-0.1,
event)">-</button>
            <button type="button"
onclick="resetLightboxZoom()">100%</button>
            <button type="button" onclick="zoomLightboxImage(0.1,
event)">+</button>
        </div>
        
        <div class="lightbox-counter" id="lightboxCounter">1 /
${carImages.length}</div>
    </div>
    ` : '<div class="carousel-container"><div class="carousel-main"><div
class="carousel-placeholder">Немає фото</div></div></div>';

    const priceHistoryHtml = car.price_history && car.price_history.length ?
    `
        <div class="info-list">
            ${car.price_history.map(item => `
                <div class="info-row">
                    <span>${formatDate(item.changed_at)}</span>
                    <strong>${formatPrice(item.old_price)} &rarr;
${formatPrice(item.new_price)}</strong>
                </div>
            `).join('')}
        </div>
    ` : '<p class="empty">Ціна не змінювалася.</p>';

    container.innerHTML = `
        <section class="page-title">
            <h1>${car.brand} ${car.model}</h1>
            <p>${car.city} · ${formatPrice(car.price)}</p>
            ${getStatusBadge(car.status)}
        </section>
        <div class="detail-layout">
            <article class="card">
                <div class="card-body">
                    ${galleryHtml}
                    <h2 class="mt-24">Інформація про авто</h2>
                    <div class="info-list">
                        <div class="info-row"><span>ID
авто</span><strong>${car.id}</strong></div>
                        <div class="info-
row"><span>Марка</span><strong>${emptyValue(car.brand)}</strong></div>
                        <div class="info-
row"><span>Модель</span><strong>${emptyValue(car.model)}</strong></div>
                        <div class="info-row"><span>VIN-
код</span><strong>${emptyValue(car.vin_code)}</strong></div>
                        <div class="info-
row"><span>Місто</span><strong>${emptyValue(car.city)}</strong></div>
                        <div class="info-
row"><span>Ціна</span><strong>${formatPrice(car.price)}</strong></div>
                        <div class="info-
row"><span>Пробіг</span><strong>${car.mileage} ?
`${Number(car.mileage).toLocaleString('uk-UA')} км` : 'Не
вказано'</strong></div>
                        <div class="info-row"><span>Рік
виготовлення</span><strong>${emptyValue(car.year)}</strong></div>
                        <div class="info-row"><span>Тип

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				152
Змн.	Арк.	№ докум.	Підпис	Дата		

```

палива</span><strong>${emptyValue(car.fuel_type)}</strong></div>
        <div class="info-row"><span>Тип
трансмисії</span><strong>${emptyValue(car.transmission)}</strong></div>
        <div class="info-
row"><span>Статус</span><strong>${getStatusBadge(car.status)}</strong></div>
        <div class="info-
row"><span>Створено</span><strong>${formatDate(car.created_at)}</strong></div>
    >
        </div>
        <h2 class="mt-24">Історія ціни</h2>
        ${priceHistoryHtml}
        <h2 class="mt-24">Опис</h2>
        <p class="description-
text">${emptyValue(car.description)}</p>
        </div>
    </article>
    <aside class="seller-panel">
        <h2>Продавець</h2>
        <div class="info-list">
            <div class="info-
row"><span>Ім'я</span><strong>${car.seller_name || 'Не
вказано'}</strong></div>
            <div class="info-
row"><span>Email</span><strong>${car.seller_email || 'Не
вказано'}</strong></div>
            <div class="info-
row"><span>Телефон</span><strong>${car.seller_phone || 'Не
вказано'}</strong></div>
            <div class="info-row"><span>Рейтинг</span><strong>★
${car.seller_rating ? car.seller_rating.toFixed(1) : 'Немає
відгуків'}</strong></div>
            <div class="info-row"><span>На сайті
з</span><strong>${formatDate(car.seller_created_at)}</strong></div>
            </div>
        </aside>
    </div>
    `;

    loadReviews();
}

```

```

window.addEventListener('DOMContentLoaded', loadCarDetails);
window.addEventListener('keydown', handleLightboxKeydown);
window.addEventListener('wheel', handleLightboxWheel, { passive: false });
</script>
</body>
</html>

```

Лістинг edit-car.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Редарувати авто | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">

```

```

        <a href="index.html">Головна</a>
        <a href="add-car.html" id="addCarLink">Додати авто</a>
        <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
        <a href="login.html" id="loginLink">Увійти</a>
        <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
    </div>
</nav>
</header>

<main class="page">
    <section class="page-title">
        <h1>Редагувати авто</h1>
        <p>Оновить характеристики та опис оголошення.</p>
    </section>

    <form class="form-panel" onsubmit="updateCar(event)">
        <div class="form-grid">
            <label>Марка
                <input id="brand" required>
            </label>
            <label>Модель
                <input id="model" required>
            </label>
            <label>VIN-код
                <input id="vin_code" maxlength="17" pattern="[A-HJ-NPR-Za-hj-
npr-z0-9]{17}" placeholder="Наприклад WBAJA71070CD12345">
            </label>
            <label>Місто
                <input id="city" required>
            </label>
            <label>Ціна, USD
                <input id="price" type="number" min="0" required>
            </label>
            <label>Пробіг, км
                <input id="mileage" type="number" min="0" required>
            </label>
            <label>Рік виготовлення
                <input id="year" type="number" min="1900" max="2100"
required>
            </label>
            <label>Тип палива
                <select id="fuel_type" required>
                    <option value="">Оберіть тип палива</option>
                    <option value="Бензин">Бензин</option>
                    <option value="Дизель">Дизель</option>
                    <option value="Газ/Бензин">Газ/Бензин</option>
                    <option value="Гібрид">Гібрид</option>
                    <option value="Електро">Електро</option>
                </select>
            </label>
            <label>Тип трансмісії
                <select id="transmission" required>
                    <option value="">Оберіть трансмісію</option>
                    <option value="Механічна">Механічна</option>
                    <option value="Автоматична">Автоматична</option>
                    <option value="Роботизована">Роботизована</option>
                    <option value="Варіатор">Варіатор</option>
                </select>
            </label>
            <label class="full">Опис авто
                <textarea id="description" required></textarea>

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				154
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        </label>
        <label class="full">Додати нові фото
            <input id="images" type="file" accept="image/*" multiple>
        </label>
    </div>
    <section class="edit-images-section">
        <h2>Поточні фото</h2>
        <div id="currentImages" class="edit-images-grid"></div>
    </section>
    <div class="actions">
        <button type="submit">Оновити</button>
        <a href="index.html" class="btn btn-secondary">Назад</a>
    </div>
</form>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Підтримуйте оголошення актуальним</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script>
let carId;
let currentImages = [];

function imageUrl(fileName) {
    return fileName ? `http://127.0.0.1:3000/uploads/${fileName}` : '';
}

function renderCurrentImages() {
    const container = document.getElementById('currentImages');
    if (!container) return;

    if (!currentImages.length) {
        container.innerHTML = '<p class="empty">Фото ще не додано.</p>';
        return;
    }

    container.innerHTML = currentImages.map(image => `
        <article class="edit-image-item">
            
            <button type="button" class="btn-danger"
onclick="deleteCarImage(${image.id})">Видалити</button>
        </article>
    `).join('');
}

async function deleteCarImage(imageId) {
    if (!confirm('Видалити це фото?')) {
        return;
    }

    try {
        await request(`/cars/${carId}/images/${imageId}`, 'DELETE');
        currentImages = currentImages.filter(image => Number(image.id) !==
Number(imageId));
        renderCurrentImages();
    } catch (err) {
        alert(err.message || 'Не вдалося видалити фото');
    }
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				155
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }
}

async function loadCar() {
  if (!getToken()) {
    alert('Щоб редагувати авто, потрібно увійти в акаунт.');
```

location.href = 'login.html';

```
    return;
  }

  const carId = new URLSearchParams(window.location.search).get('id');
  const car = await request(`/cars/${carId}`);

  document.getElementById('brand').value = car.brand || '';
  document.getElementById('model').value = car.model || '';
  document.getElementById('vin_code').value = car.vin_code || '';
  document.getElementById('city').value = car.city || '';
  document.getElementById('price').value = car.price || '';
  document.getElementById('mileage').value = car.mileage || '';
  document.getElementById('year').value = car.year || '';
  document.getElementById('fuel_type').value = car.fuel_type || '';
  document.getElementById('transmission').value = car.transmission || '';
  document.getElementById('description').value = car.description || '';
  currentImages = car.images || [];
  renderCurrentImages();
}

async function updateCar(e) {
  e.preventDefault();

  try {
    const formData = new FormData();
    formData.append('brand', document.getElementById('brand').value);
    formData.append('model', document.getElementById('model').value);
    formData.append('vin_code',
document.getElementById('vin_code').value);
    formData.append('city', document.getElementById('city').value);
    formData.append('price', document.getElementById('price').value);
    formData.append('mileage', document.getElementById('mileage').value);
    formData.append('year', document.getElementById('year').value);
    formData.append('fuel_type',
document.getElementById('fuel_type').value);
    formData.append('transmission',
document.getElementById('transmission').value);
    formData.append('description',
document.getElementById('description').value);

    for (const file of document.getElementById('images').files) {
      formData.append('images', file);
    }

    const response = await
fetch(`http://127.0.0.1:3000/api/cars/${carId}`, {
      method: 'PUT',
      headers: {
        Authorization: 'Bearer ' + getToken()
      },
      body: formData
    });

    const data = await response.json();

    if (!response.ok) {

```

```

        throw new Error(data.message || 'Помилка при оновленні авто');
    }

    alert('Оновлено');
    location.href = `car.html?id=${carId}`;
} catch (err) {
    alert(err.message || 'Помилка при оновленні авто');
}
}

window.addEventListener('DOMContentLoaded', loadCar);
</script>
</body>
</html>

```

Лістинг forgot-password.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Відновлення паролю | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">
            <a href="index.html">Головна</a>
            <a href="add-car.html" id="addCarLink">Додати авто</a>
            <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
            <a href="login.html" id="loginLink">Увійти</a>
            <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
        </div>
    </nav>
</header>

<main class="page">
    <section class="page-title center-text">
        <h1>Відновлення паролю</h1>
        <p>Введіть email акаунта, і сайт створить одноразове посилання для
зміни паролю.</p>
    </section>

    <form class="form-panel" onsubmit="requestPasswordReset(event)">
        <div id="errorMessage" class="error-message"></div>
        <div id="successMessage" class="success-message"></div>
        <div class="form-grid">
            <label class="full">Email
                <input id="resetEmail" type="email" required>
            </label>
        </div>
        <div class="actions">
            <button type="submit">Створити посилання</button>
            <a href="login.html" class="btn btn-secondary">Назад до входу</a>
        </div>
    </form>
</main>

```

```

<footer class="site-footer">
  <div class="footer-inner">
    <strong>AutoHub</strong>
    <span>Безпечне відновлення доступу до акаунта</span>
  </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/auth.js"></script>
</body>
</html>

```

Лістинг index.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>AutoHub</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
  <nav class="nav">
    <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
    <div class="nav-links">
      <a href="index.html">Головна</a>
      <a href="add-car.html" id="addCarLink">Додати авто</a>
      <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
      <a href="login.html" id="loginLink">Увійти</a>
      <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
    </div>
  </nav>
</header>

<main class="page">
  <section class="page-title">
    <h1>Знайдіть авто для покупки</h1>
    <p>Переглядайте оголошення, фільтруйте за містом і ціною та
відкривайте повну інформацію про авто.</p>
  </section>

  <form class="toolbar" onsubmit="filterCars(event)">
    <input id="brandFilter" placeholder="Марка">
    <input id="modelFilter" placeholder="Модель">
    <input id="cityFilter" placeholder="Місто">
    <input id="minPrice" type="number" min="0" placeholder="Ціна від">
    <input id="maxPrice" type="number" min="0" placeholder="Ціна до">
    <input id="yearFrom" type="number" min="1900" max="2100"
placeholder="Рік від">
    <input id="yearTo" type="number" min="1900" max="2100"
placeholder="Рік до">
    <input id="minMileage" type="number" min="0" placeholder="Пробіг
від">
    <input id="maxMileage" type="number" min="0" placeholder="Пробіг до">
    <select id="fuelTypeFilter">
      <option value="">Будь-яке паливо</option>
      <option value="Бензин">Бензин</option>
      <option value="Дизель">Дизель</option>

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				
Змн.	Арк.	№ докум.	Підпис	Дата		158

```

        <option value="Газ/Бензин">Газ/Бензин</option>
        <option value="Гібрид">Гібрид</option>
        <option value="Електро">Електро</option>
    </select>
    <select id="transmissionFilter">
        <option value="">Будь-яка коробка</option>
        <option value="Механічна">Механічна</option>
        <option value="Автоматична">Автоматична</option>
        <option value="Роботизована">Роботизована</option>
        <option value="Варіатор">Варіатор</option>
    </select>
    <select id="sortFilter">
        <option value="newest">Спочатку нові</option>
        <option value="price_asc">Спочатку дешевші</option>
        <option value="price_desc">Спочатку дорожчі</option>
        <option value="year_desc">Новіший рік</option>
        <option value="mileage_asc">Менший пробіг</option>
    </select>
    <div class="toolbar-actions">
        <button type="submit">Фільтрувати</button>
        <button type="button" class="btn-secondary"
onclick="resetFilters()">Скинути</button>
    </div>
</form>

<div id="cars" class="grid"></div>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Сучасний маркетплейс авто</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/cars.js"></script>
<script>
    window.addEventListener('DOMContentLoaded', () => loadCars());
</script>
</body>
</html>

```

Лістинг login.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Bxід | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">
            <a href="index.html">Головна</a>
            <a href="add-car.html" id="addCarLink">Додати авто</a>
            <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				159
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        <a href="login.html" id="loginLink">Увійти</a>
        <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
    </div>
</nav>
</header>

<main class="page">
    <section class="page-title center-text">
        <h1>Вхід</h1>
        <p>Увійдіть, щоб додавати авто, зберігати обране та залишати
відгуки.</p>
    </section>

    <form class="form-panel" onsubmit="login(event)">
        <div id="errorMessage" class="error-message"></div>
        <div class="form-grid">
            <label class="full">Email
                <input id="email" type="email" required>
            </label>
            <label class="full">Пароль
                <input id="password" type="password" required>
            </label>
        </div>
        <div class="actions">
            <button type="submit">Увійти</button>
            <a href="register.html" class="btn btn-
secondary">Зареєструватися</a>
            <a href="forgot-password.html" class="btn btn-secondary">Забули
пароль?</a>
        </div>
    </form>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Акаунт потрібен для дій з оголошеннями</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/auth.js"></script>
<script>
    const originalLogin = window.login;
    window.login = async function(e) {
        const errorDiv = document.getElementById('errorMessage');
        errorDiv.style.display = 'none';

        try {
            await originalLogin(e);
        } catch (err) {
            errorDiv.innerText = err.message || 'Помилка входу. Перевірте
email та пароль.';
            errorDiv.style.display = 'block';
        }
    };
</script>
</body>
</html>

```

Лістинг profile.html

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				160
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<!DOCTYPE html>
<html lang="uk">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Профіль | AutoHub</title>
  <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
  <nav class="nav">
    <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
    <div class="nav-links">
      <a href="index.html">Головна</a>
      <a href="add-car.html" id="addCarLink">Додати авто</a>
      <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
      <a href="login.html" id="loginLink">Увійти</a>
      <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
    </div>
  </nav>
</header>

<main class="page">
  <section class="page-title">
    <h1>Мій профіль</h1>
    <p>Дані акаунта, ваші оголошення та обрані авто.</p>
  </section>

  <div class="detail-layout">
    <section class="profile-data">
      <h2>Дані користувача</h2>
      <div class="info-list" id="profileInfo"></div>
    </section>

    <form class="form-panel" onsubmit="saveProfile(event)">
      <h2>Редагувати</h2>
      <div class="form-grid">
        <label class="full">Ім'я
          <input id="nameInput" required>
        </label>
        <label class="full">Телефон
          <input id="phoneInput" type="tel">
        </label>
        <label class="full">Новий пароль
          <input id="passwordInput" type="password"
placeholder="Залиште порожнім, якщо не змінюєте">
        </label>
      </div>
      <div class="actions">
        <button type="submit">Зберегти профіль</button>
      </div>
    </form>
  </div>

  <section class="mt-24">
    <div class="page-title">
      <h2>Додані мною авто</h2>
    </div>
    <div id="myCars" class="grid"></div>
  </section>

```

```

<section class="mt-24">
  <div class="page-title">
    <h2>Мої улюблені авто</h2>
  </div>
  <div id="myFavorites" class="grid"></div>
</section>

<section class="mt-24">
  <div class="page-title">
    <h2>Мої замовлення</h2>
  </div>
  <div id="myOrders" class="grid"></div>
</section>

<section class="mt-24">
  <div class="page-title">
    <h2>Замовлення на мої авто - очікують підтвердження</h2>
  </div>
  <div id="mySellerOrders" class="grid"></div>
</section>

<section class="mt-24">
  <div class="page-title">
    <h2>Чати з покупцями</h2>
  </div>
  <div id="sellerChats" class="grid"></div>
</section>
</main>

<footer class="site-footer">
  <div class="footer-inner">
    <strong>AutoHub</strong>
    <span>Ваші дані доступні тільки після входу</span>
  </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/cars.js"></script>
<script src="js/favorites.js"></script>
<script>
function formatPrice(value) {
  return Number(value || 0).toLocaleString('uk-UA', {
    style: 'currency',
    currency: 'USD',
    maximumFractionDigits: 0
  });
}

function formatDate(date) {
  return date ? new Date(date).toLocaleString('uk-UA') : 'Не вказано';
}

function getStatusBadge(status) {
  if (status === 'sold') {
    return '<span style="display:inline-block;background:#f8d7da;color:#721c24;padding:4px 8px;border-radius:4px;font-weight:bold;font-size:12px;">● ПРОДАНО</span>';
  }
  return '<span style="display:inline-block;background:#d4edda;color:#155724;padding:4px 8px;border-radius:4px;font-weight:bold;font-size:12px;">□ АКТИВНО</span>';
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				162
Змн.	Арк.	№ докум.	Підпис	Дата		

```

function renderProfileInfo(user) {
    document.getElementById('profileInfo').innerHTML = `
        <div class="info-
row"><span>ID</span><strong>${user.id}</strong></div>
        <div class="info-row"><span>Ім'я</span><strong>${user.name} || 'Не
вказано'</strong></div>
        <div class="info-row"><span>Email</span><strong>${user.email} || 'Не
вказано'</strong></div>
        <div class="info-row"><span>Телефон</span><strong>${user.phone} || 'Не
вказано'</strong></div>
        <div class="info-row"><span>Рейтинг</span><strong>${user.rating ?
user.rating.toFixed(1) : 'Немає'} ★</strong></div>
        <div class="info-row"><span>Роль</span><strong>${user.role} ||
'user'</strong></div>
        <div class="info-
row"><span>Створено</span><strong>${formatDate(user.created_at)}</strong></di
v>
    `;
}

function withExtraActions(card, actionsHtml) {
    return card.replace(
        '</div>\n          </article>',
        `<div class="actions">${actionsHtml}</div></div></article>`
    );
}

function renderMyCarCard(car) {
    const cardWithStatus = renderCarCard(car);
    const statusHtml = getStatusBadge(car.status);
    const statsHtml = `
        <div class="info-list">
            <div class="info-
row"><span>Перегляди</span><strong>${Number(car.view_count) ||
0}.toLocaleString('uk-UA')}</strong></div>
            <div class="info-row"><span>Унікальні
користувачі</span><strong>${Number(car.unique_viewers) ||
0}.toLocaleString('uk-UA')}</strong></div>
            <div class="info-row"><span>Останній
перегляд</span><strong>${formatDate(car.last_viewed_at)}</strong></div>
        </div>
    `;
    const withStatus = cardWithStatus.replace(
        '<p class="price">',
        `${statusHtml}<p class="price">`
    ).replace(
        '<a href="car.html',
        `${statsHtml}<a href="car.html`
    );
    return withExtraActions(
        withStatus,
        `<a class="btn btn-secondary" href="edit-
car.html?id=${car.id}">Редагувати</a>
        <button type="button" class="btn-danger"
onclick="deleteMycar(${car.id})">Видалити</button>`
    );
}

async function deleteMycar(carId) {
    if (!confirm('Ви впевнені, що хочете видалити це оголошення?')) {
        return;
    }
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				163
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    }

    try {
      await request(`/cars/${carId}`, 'DELETE');
      alert('Оголошення видалено');
      loadProfile();
    } catch (err) {
      alert(err.message || 'Не вдалося видалити оголошення');
    }
  }

function renderFavoriteCard(car) {
  return withExtraActions(
    renderCarCard(car),
    `<button type="button" class="btn-danger"
onclick="deleteFavorite(${car.id})">Видалити з обраного</button>`
  );
}

async function deleteFavorite(carId) {
  try {
    await removeFromFavorites(carId);
    alert('Авто видалено з обраного');
    loadProfile();
  } catch (err) {
    alert(err.message || 'Не вдалося видалити авто з обраного');
  }
}

function escapeHtml(value) {
  return String(value ?? '').replace(/([<>"']/g, char => ({
    '&': '&amp;',
    '<': '&lt;',
    '>': '&gt;',
    '"': '&quot;',
    "'": '&#39;',
  }[char]));
}

function getOrderStatusText(order) {
  const status = order.order_status || order.status;

  if (status === 'pending') return '❏ Очікує підтвердження';
  if (status === 'approved') return '✅ Підтверджено';
  if (status === 'rejected') return '❌ Відхилено';

  return 'Статус невідомий';
}

function getChatContainerId(carId, otherUserId) {
  return `sellerChatMessages-${carId}-${otherUserId}`;
}

async function renderSellerConversationMessages(conversation) {
  const container =
document.getElementById(getChatContainerId(conversation.car_id,
conversation.other_user_id));
  if (!container) return;

  try {
    const messages = await
request(`/messages/conversation/${conversation.other_user_id}?carId=${convers
ation.car_id}`);

```

```

        if (!messages || messages.length === 0) {
            container.innerHTML = '<p class="empty">Повідомлень поки
немає.</p>';
            return;
        }

        container.innerHTML = messages.map(message => `
            <div class="chat-message ${message.sender_id ===
conversation.other_user_id ? 'received' : 'sent'}">
                <div class="chat-sender">${escapeHtml(message.sender_name ||
'' )}</div>
                <div>${escapeHtml(message.message)}</div>
                <small>${formatDate(message.created_at)}</small>
            </div>
        `).join('');

        container.scrollTop = container.scrollHeight;
    } catch (err) {
        container.innerHTML = `<p class="empty">${escapeHtml(err.message ||
'Не вдалося завантажити чат.' )}</p>`;
    }
}

async function loadSellerChats() {
    const container = document.getElementById('sellerChats');
    if (!container) return;

    try {
        const conversations = await request('/messages/seller-
conversations');

        if (!conversations || conversations.length === 0) {
            container.innerHTML = '<p class="empty">Чатів з покупцями поки
немає.</p>';
            return;
        }

        container.innerHTML = conversations.map(conversation => `
            <article class="card">
                <div class="card-body chat-thread">
                    <div class="chat-thread-header">
                        <div>
                            <h3>${escapeHtml(conversation.brand)}
${escapeHtml(conversation.model)}</h3>
                            <p class="car-
meta">${escapeHtml(conversation.other_user_name ||
conversation.other_user_email || 'Користувач')}</p>
                        </div>
                        <strong>${conversation.unread_count ?
`${conversation.unread_count} нових` : ''}</strong>
                    </div>
                    <div class="chat-messages"
id="${getChatContainerId(conversation.car_id,
conversation.other_user_id)}"></div>
                    <form class="chat-form"
onsubmit="sendSellerMessage(event, ${conversation.other_user_id},
${conversation.car_id})">
                        <input type="text" name="message"
placeholder="Напишіть повідомлення..." required>
                        <button type="submit">Відправити</button>
                    </form>
                </div>
            </article>
        `).join('');
    } catch (err) {
        container.innerHTML = `<p class="empty">Не вдалося завантажити чати.</p>`;
    }
}

```

```

        </article>
    `).join('');

    await
    Promise.all(conversations.map(renderSellerConversationMessages));
    } catch (err) {
        container.innerHTML = `<p class="empty">${escapeHtml(err.message ||
'Не вдалося завантажити чати.')}</p>`;
    }
}

async function sendSellerMessage(event, receiverId, carId) {
    event.preventDefault();

    const input = event.target.elements.message;
    const message = input.value.trim();
    if (!message) return;

    try {
        await request('/messages', 'POST', {
            receiver_id: receiverId,
            car_id: carId,
            message
        });

        input.value = '';
        await loadSellerChats();
    } catch (err) {
        alert(err.message || 'Не вдалося відправити повідомлення');
    }
}

async function loadProfile() {
    if (!getToken()) {
        location.href = 'login.html';
        return;
    }

    try {
        const user = await request('/auth/profile');
        renderProfileInfo(user);
        document.getElementById('nameInput').value = user.name || '';
        document.getElementById('phoneInput').value = user.phone || '';

        const myCars = await request('/cars/user/my');
        document.getElementById('myCars').innerHTML = myCars && myCars.length
        ? myCars.map(renderMyCarCard).join('')
        : '<p class="empty">Ви ще не додали жодного авто.</p>';

        const favorites = await request('/favorites');
        document.getElementById('myFavorites').innerHTML = favorites &&
favorites.length
        ? favorites.map(renderFavoriteCard).join('')
        : '<p class="empty">Улюблених авто поки немає.</p>';

        try {
            const orders = await request('/orders');
            const ordersContainer = document.getElementById('myOrders');

            if (!orders || orders.length === 0) {
                ordersContainer.innerHTML = '<p class="empty">Ви не робили
замовлень.</p>';
            } else {

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				166
Змн.	Арк.	№ докум.	Підпис	Дата		

```

ordersContainer.innerHTML = orders.map(order => {
  const statusText = getOrderStatusText(order);

  return `
    <article class="card">
      <div class="card-body">
        <h3>${order.brand} ${order.model}</h3>
        <div class="info-list">
          <div class="info-
row"><span>Ціна</span><strong>${formatPrice(order.price)}</strong></div>
          <div class="info-
row"><span>Місто</span><strong>${order.city}</strong></div>
          <div class="info-row"><span>Статус
замовлення</span><strong>${statusText}</strong></div>
          <div class="info-
row"><span>Продавець</span><strong>${order.seller_name || 'Не
вказано'}</strong></div>
          <div class="info-row"><span>Email
продавця</span><strong>${order.seller_email || 'Не вказано'}</strong></div>
          </div>
          <div class="actions">
            <a class="btn btn-secondary"
href="car.html?id=${order.car_id}">Переглянути авто</a>
          </div>
        </div>
      </article>
    `;
  })
  .join('');
}
} catch (err) {
  console.log('Could not load orders:', err.message);
  document.getElementById('myOrders').innerHTML = '<p
class="empty">Не вдалося завантажити замовлення</p>';
}

try {
  const sellerOrders = await request('/orders/seller/my');
  const sellerOrdersContainer =
document.getElementById('mySellerOrders');

  if (!sellerOrders || sellerOrders.length === 0) {
    sellerOrdersContainer.innerHTML = '<p class="empty">Немає
замовлень на підтвердження.</p>';
  } else {
    sellerOrdersContainer.innerHTML = sellerOrders.map(order => {
      const statusText = getOrderStatusText(order);

      const actionButtons = (order.order_status ||
order.status) === 'pending' ? `
        <button class="btn btn-success"
onclick="approveOrder(${order.id}, '${order.brand} ${order.model}')">✓
Підтвердити продаж</button>
        <button class="btn-danger"
onclick="rejectOrder(${order.id}, '${order.brand} ${order.model}')">✗
Відхилити</button>
      ` : '';

      return `
        <article class="card">
          <div class="card-body">
            <h3>${order.brand} ${order.model}</h3>
            <div class="info-list">
              <div class="info-

```

```

row"><span>Ціна</span><strong>${formatPrice(order.price)}</strong></div>
                                <div class="info-
row"><span>Статус</span><strong>${statusText}</strong></div>
                                <div class="info-
row"><span>Покупець</span><strong>${order.buyer_name || 'Не
вказано'}</strong></div>
                                <div class="info-row"><span>Email
покупця</span><strong>${order.buyer_email || 'Не вказано'}</strong></div>
                                <div class="info-row"><span>Телефон
покупця</span><strong>${order.buyer_phone || 'Не вказано'}</strong></div>
                                </div>
                                <div class="actions">
                                    ${actionButtons}
                                </div>
                            </div>
                        </article>
                    `;
                }).join('');
            }
        } catch (err) {
            console.log('Could not load seller orders:', err.message);
            document.getElementById('mySellerOrders').innerHTML = '<p
class="empty">Не вдалося завантажити замовлення</p>';
        }

        await loadSellerChats();
    } catch (err) {
        alert(err.message || 'Не вдалося завантажити профіль');
    }
}

async function approveOrder(orderId, carName) {
    const confirmApprove = confirm('Підтвердити продаж авто "${carName}"?');
    if (!confirmApprove) return;

    try {
        await request(`/orders/${orderId}/approve`, 'PUT');
        alert('✓ Продаж підтверджено! Авто помічено як продано.');
```

```

        loadProfile();
    } catch (err) {
        alert(err.message || 'Не вдалося підтвердити замовлення');
    }
}

async function rejectOrder(orderId, carName) {
    const confirmReject = confirm('Відхилити замовлення на "${carName}"?');
    if (!confirmReject) return;

    try {
        await request(`/orders/${orderId}/reject`, 'PUT');
        alert('✗ Замовлення відхилено.');
```

```

        loadProfile();
    } catch (err) {
        alert(err.message || 'Не вдалося відхилити замовлення');
    }
}

async function saveProfile(e) {
    e.preventDefault();

    try {
        await request('/auth/profile', 'PUT', {
            name: document.getElementById('nameInput').value,
```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				168
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        phone: document.getElementById('phoneInput').value,
        password: document.getElementById('passwordInput').value
    });

    alert('Профіль оновлено');
    document.getElementById('passwordInput').value = '';
    loadProfile();
} catch (err) {
    alert(err.message || 'Помилка при збереженні профілю');
}
}

window.addEventListener('DOMContentLoaded', loadProfile);
</script>
</body>
</html>

```

Лістинг register.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Реєстрація | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">
            <a href="index.html">Головна</a>
            <a href="add-car.html" id="addCarLink">Додати авто</a>
            <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
            <a href="login.html" id="loginLink">Увійти</a>
            <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
        </div>
    </nav>
</header>

<main class="page">
    <section class="page-title center-text">
        <h1>Реєстрація</h1>
        <p>Створіть акаунт для публікації оголошень.</p>
    </section>

    <form class="form-panel" onsubmit="register(event)">
        <div id="errorMessage" class="error-message"></div>
        <div class="form-grid">
            <label class="full">Ім'я
                <input id="name" required>
            </label>
            <label class="full">Email
                <input id="email" type="email" required>
            </label>
            <label class="full">Пароль
                <input id="password" type="password" required>
            </label>
            <label class="full">Телефон
                <input id="phone" type="tel">
        </div>
    </form>

```

```

        </label>
    </div>
    <div class="actions">
        <button type="submit">Зареєструватися</button>
        <a href="login.html" class="btn btn-secondary">Увійти</a>
    </div>
</form>
</main>

<footer class="site-footer">
    <div class="footer-inner">
        <strong>AutoHub</strong>
        <span>Контактні дані допомагають покупцям зв'язатися з вами</span>
    </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/auth.js"></script>
<script>
    const originalRegister = window.register;
    window.register = async function(e) {
        const errorDiv = document.getElementById('errorMessage');
        errorDiv.style.display = 'none';

        try {
            await originalRegister(e);
        } catch (err) {
            errorDiv.innerText = err.message || 'Помилка реєстрації.
Спробуйте ще раз.';
            errorDiv.style.display = 'block';
        }
    };
</script>
</body>
</html>

```

Лістинг reset-password.html

```

<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Новий пароль | AutoHub</title>
    <link rel="stylesheet" href="css/style.css">
</head>
<body>
<header class="site-header">
    <nav class="nav">
        <a class="brand" href="index.html"><span class="brand-
mark">A</span>AutoHub</a>
        <div class="nav-links">
            <a href="index.html">Головна</a>
            <a href="add-car.html" id="addCarLink">Додати авто</a>
            <a href="profile.html" id="profileLink"
class="hidden">Профіль</a>
            <a href="login.html" id="loginLink">Увійти</a>
            <a href="#" id="logoutLink" class="hidden"
onclick="logout()">Вийти</a>
        </div>
    </nav>
</header>

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				170
Змн.	Арк.	№ докум.	Підпис	Дата		

```

<main class="page">
  <section class="page-title center-text">
    <h1>Новий пароль</h1>
    <p>Введіть новий пароль для свого акаунта.</p>
  </section>

  <form class="form-panel" onsubmit="resetPassword(event)">
    <div id="errorMessage" class="error-message"></div>
    <div id="successMessage" class="success-message"></div>
    <div class="form-grid">
      <label class="full">Новий пароль
      <input id="newPassword" type="password" minlength="6"
required>
      </label>
      <label class="full">Повторіть пароль
      <input id="confirmPassword" type="password" minlength="6"
required>
      </label>
    </div>
    <div class="actions">
      <button type="submit">Змінити пароль</button>
      <a href="login.html" class="btn btn-secondary">Назад до входу</a>
    </div>
  </form>
</main>

<footer class="site-footer">
  <div class="footer-inner">
    <strong>AutoHub</strong>
    <span>Після зміни паролю увійдіть з новими даними</span>
  </div>
</footer>

<script src="js/api.js"></script>
<script src="js/layout.js"></script>
<script src="js/auth.js"></script>
<script>
  window.addEventListener('DOMContentLoaded', checkResetToken);
</script>
</body>
</html>

```

Лістинг style.css

```

:root {
  --bg: #f4f7fb;
  --surface: #ffffff;
  --surface-soft: #eef3f8;
  --text: #172033;
  --muted: #637083;
  --line: #dce4ee;
  --primary: #2563eb;
  --primary-dark: #1d4ed8;
  --accent: #0f766e;
  --danger: #dc2626;
  --shadow: 0 14px 35px rgba(23, 32, 51, 0.09);
  --radius: 8px;
}

* {
  box-sizing: border-box;
}

body {

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				171
Змн.	Арк.	№ докум.	Підпис	Дата		

```

margin: 0;
min-height: 100vh;
display: flex;
flex-direction: column;
background: var(--bg);
color: var(--text);
font-family: Inter, "Segoe UI", Arial, sans-serif;
line-height: 1.5;
}

body.no-scroll {
  overflow: hidden;
}

a {
  color: inherit;
  text-decoration: none;
}

.site-header {
  position: sticky;
  top: 0;
  z-index: 10;
  background: rgba(255, 255, 255, 0.94);
  border-bottom: 1px solid var(--line);
  backdrop-filter: blur(12px);
}

.nav {
  width: min(1180px, calc(100% - 32px));
  min-height: 72px;
  margin: 0 auto;
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 20px;
}

.brand {
  display: flex;
  align-items: center;
  gap: 10px;
  font-size: 22px;
  font-weight: 800;
  letter-spacing: 0;
}

.brand-mark {
  width: 36px;
  height: 36px;
  display: grid;
  place-items: center;
  border-radius: var(--radius);
  background: var(--text);
  color: #fff;
}

.nav-links {
  display: flex;
  align-items: center;
  flex-wrap: wrap;
  justify-content: flex-end;
  gap: 8px;
}

```

```

}

.nav-links a,
.btn,
button {
  min-height: 40px;
  display: inline-flex;
  align-items: center;
  justify-content: center;
  gap: 8px;
  border: 1px solid transparent;
  border-radius: var(--radius);
  padding: 10px 14px;
  background: var(--primary);
  color: #fff;
  font: inherit;
  font-weight: 700;
  cursor: pointer;
  transition: background 0.2s ease, border-color 0.2s ease, transform 0.2s
ease;
}

.nav-links a {
  background: transparent;
  color: var(--muted);
}

.nav-links a:hover,
.nav-links a.active {
  background: var(--surface-soft);
  color: var(--text);
}

.btn:hover,
button:hover {
  background: var(--primary-dark);
  transform: translateY(-1px);
}

.btn-secondary {
  background: var(--surface);
  color: var(--text);
  border-color: var(--line);
}

.btn-secondary:hover {
  background: var(--surface-soft);
}

.btn-danger {
  background: var(--danger);
}

.btn-success {
  background: #28a745;
  color: #fff;
}

.btn-success:hover {
  background: #218838;
}

.status-badge {

```

```

display: inline-block;
padding: 6px 12px;
border-radius: 20px;
font-weight: 700;
font-size: 14px;
margin-top: 8px;
}

.status-active {
background: #d4edda;
color: #155724;
}

.status-sold {
background: #f8d7da;
color: #721c24;
}

.hidden {
display: none !important;
}

.page {
width: min(1180px, calc(100% - 32px));
margin: 0 auto;
padding: 34px 0 44px;
flex: 1;
}

.page-title {
margin: 0 0 22px;
}

.page-title h1,
.page-title h2 {
margin: 0 0 6px;
font-size: clamp(28px, 4vw, 42px);
}

.page-title p {
margin: 0;
color: var(--muted);
}

.toolbar,
.card,
.form-panel,
.seller-panel,
.profile-data {
background: var(--surface);
border: 1px solid var(--line);
border-radius: var(--radius);
box-shadow: var(--shadow);
}

.toolbar {
display: grid;
grid-template-columns: repeat(auto-fit, minmax(150px, 1fr));
gap: 12px;
padding: 16px;
margin-bottom: 24px;
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				174
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.toolbar-actions {
  display: grid;
  grid-template-columns: repeat(2, minmax(0, 1fr));
  gap: 10px;
  grid-column: span 2;
}

.toolbar-actions button {
  min-width: 140px;
  white-space: nowrap;
}

.grid {
  display: grid;
  grid-template-columns: repeat(auto-fill, minmax(260px, 1fr));
  gap: 18px;
}

.card {
  overflow: hidden;
}

.card-body {
  padding: 16px;
}

.car-card {
  display: flex;
  flex-direction: column;
  min-height: 100%;
}

.car-image {
  width: 100%;
  aspect-ratio: 16 / 10;
  object-fit: cover;
  background: var(--surface-soft);
  display: block;
}

.car-placeholder {
  width: 100%;
  aspect-ratio: 16 / 10;
  display: grid;
  place-items: center;
  background: var(--surface-soft);
  color: var(--muted);
  font-weight: 700;
}

.car-title {
  margin: 0 0 6px;
  font-size: 20px;
}

.car-meta {
  margin: 0 0 14px;
  color: var(--muted);
}

.price {
  font-size: 24px;
  font-weight: 800;
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				175
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        color: var(--accent);
    }

    .form-panel {
        max-width: 680px;
        margin: 0 auto;
        padding: 24px;
    }

    .form-grid {
        display: grid;
        grid-template-columns: repeat(2, 1fr);
        gap: 14px;
    }

    .form-grid .full {
        grid-column: 1 / -1;
    }

    .edit-images-section {
        margin-top: 24px;
    }

    .edit-images-section h2 {
        margin: 0 0 12px;
        font-size: 22px;
    }

    .edit-images-grid {
        display: grid;
        grid-template-columns: repeat(auto-fill, minmax(150px, 1fr));
        gap: 12px;
    }

    .edit-image-item {
        display: grid;
        gap: 8px;
    }

    .edit-image-item img {
        width: 100%;
        aspect-ratio: 16 / 10;
        object-fit: cover;
        border: 1px solid var(--line);
        border-radius: var(--radius);
        background: var(--surface-soft);
    }

    .edit-image-item button {
        width: 100%;
    }

    label {
        display: grid;
        gap: 6px;
        color: var(--muted);
        font-weight: 700;
    }

    input,
    textarea,
    select {
        width: 100%;
    }

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				176
Змн.	Арк.	№ докум.	Підпис	Дата		

```

    min-height: 44px;
    border: 1px solid var(--line);
    border-radius: var(--radius);
    padding: 10px 12px;
    color: var(--text);
    background: #fff;
    font: inherit;
}

textarea {
    min-height: 110px;
    resize: vertical;
}

input:focus,
textarea:focus {
    outline: 3px solid rgba(37, 99, 235, 0.16);
    border-color: var(--primary);
}

.actions {
    display: flex;
    gap: 10px;
    flex-wrap: wrap;
    margin-top: 16px;
}

.detail-layout {
    display: grid;
    grid-template-columns: minmax(0, 1.7fr) minmax(280px, 0.8fr);
    gap: 22px;
}

.gallery {
    display: grid;
    grid-template-columns: repeat(auto-fit, minmax(190px, 1fr));
    gap: 12px;
}

.gallery img {
    width: 100%;
    aspect-ratio: 16 / 10;
    object-fit: cover;
    border-radius: var(--radius);
    border: 1px solid var(--line);
}

.carousel-container {
    position: relative;
    background: #f5f5f5;
    border-radius: var(--radius);
    overflow: hidden;
    margin-bottom: 24px;
}

.carousel-main {
    width: 100%;
    aspect-ratio: 16 / 9;
    background: #e0e0e0;
    display: flex;
    align-items: center;
    justify-content: center;
    overflow: hidden;

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				177
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.carousel-main img {
  width: 100%;
  height: 100%;
  object-fit: cover;
  cursor: zoom-in;
}

.carousel-placeholder {
  font-size: 18px;
  color: #999;
  text-align: center;
}

.carousel-controls {
  position: absolute;
  top: 50%;
  transform: translateY(-50%);
  width: 100%;
  display: flex;
  justify-content: space-between;
  padding: 0 16px;
  z-index: 10;
}

.carousel-btn {
  background: rgba(0, 0, 0, 0.5);
  color: #fff;
  border: none;
  width: 40px;
  height: 40px;
  border-radius: 50%;
  cursor: pointer;
  font-size: 20px;
  display: flex;
  align-items: center;
  justify-content: center;
  transition: background 0.3s;
}

.carousel-btn:hover {
  background: rgba(0, 0, 0, 0.8);
}

.carousel-thumbnails {
  display: flex;
  gap: 8px;
  padding: 12px;
  overflow-x: auto;
  background: #fff;
}

.carousel-thumbnail {
  width: 60px;
  height: 60px;
  border-radius: 4px;
  cursor: pointer;
  border: 2px solid transparent;
  overflow: hidden;
  flex-shrink: 0;
  transition: border-color 0.3s;
}

```

```

.carousel-thumbnail img {
  width: 100%;
  height: 100%;
  object-fit: cover;
}

.carousel-thumbnail.active {
  border-color: var(--primary);
}

.image-lightbox {
  position: fixed;
  inset: 0;
  z-index: 100;
  display: none;
  align-items: center;
  justify-content: center;
  padding: 72px 76px;
  background: rgba(15, 23, 42, 0.92);
}

.image-lightbox.active {
  display: flex;
}

.lightbox-image {
  max-width: 100%;
  max-height: 100%;
  object-fit: contain;
  border-radius: var(--radius);
  box-shadow: 0 24px 70px rgba(0, 0, 0, 0.45);
  transition: transform 0.16s ease;
  transform-origin: center center;
}

.lightbox-close,
.lightbox-nav {
  position: absolute;
  border-color: rgba(255, 255, 255, 0.24);
  background: rgba(255, 255, 255, 0.12);
  color: #fff;
}

.lightbox-close {
  top: 20px;
  right: 20px;
  width: 48px;
  height: 48px;
  padding: 0;
  font-size: 32px;
  line-height: 1;
}

.lightbox-nav {
  top: 50%;
  width: 52px;
  height: 52px;
  padding: 0;
  transform: translateY(-50%);
  font-size: 24px;
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				179
Змн.	Арк.	№ докум.	Підпис	Дата		

```

.lightbox-prev {
  left: 20px;
}

.lightbox-next {
  right: 20px;
}

.lightbox-close:hover,
.lightbox-nav:hover {
  background: rgba(255, 255, 255, 0.22);
}

.lightbox-counter {
  position: absolute;
  left: 50%;
  bottom: 24px;
  transform: translateX(-50%);
  padding: 8px 12px;
  border-radius: var(--radius);
  background: rgba(255, 255, 255, 0.12);
  color: #fff;
  font-weight: 700;
}

.lightbox-zoom-controls {
  position: absolute;
  top: 20px;
  left: 50%;
  display: grid;
  grid-template-columns: 44px 68px 44px;
  gap: 8px;
  transform: translateX(-50%);
}

.lightbox-zoom-controls button {
  min-height: 44px;
  padding: 0;
  border-color: rgba(255, 255, 255, 0.24);
  background: rgba(255, 255, 255, 0.12);
  color: #fff;
  white-space: nowrap;
}

.lightbox-zoom-controls button:hover {
  background: rgba(255, 255, 255, 0.22);
}

.action-buttons {
  display: flex;
  gap: 12px;
  flex-wrap: wrap;
  margin-top: 24px;
}

.action-buttons button,
.action-buttons a {
  flex: 1;
  min-width: 150px;
}

.chat-section {
  background: #f9f9f9;
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				180
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        border-radius: var(--radius);
        padding: 16px;
        margin-top: 24px;
    }

    .chat-messages {
        background: #fff;
        border: 1px solid #e0e0e0;
        border-radius: 4px;
        height: 300px;
        overflow-y: auto;
        padding: 12px;
        margin-bottom: 12px;
    }

    .chat-message {
        margin-bottom: 12px;
        padding: 8px;
        border-radius: 4px;
    }

    .chat-message.sent {
        background: #e3f2fd;
        text-align: right;
        margin-left: 40px;
    }

    .chat-message.received {
        background: #f5f5f5;
        margin-right: 40px;
    }

    .chat-sender {
        font-size: 12px;
        color: #666;
        margin-bottom: 4px;
    }

    .chat-form {
        display: flex;
        gap: 8px;
    }

    .chat-form input {
        flex: 1;
    }

    .chat-thread {
        display: grid;
        gap: 12px;
    }

    .chat-thread-header {
        display: flex;
        align-items: flex-start;
        justify-content: space-between;
        gap: 12px;
        flex-wrap: wrap;
    }

    .seller-panel,
    .profile-data {
        padding: 18px;
    }

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				181
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.info-list {
  display: grid;
  gap: 10px;
  margin: 0;
}

.info-row {
  display: flex;
  justify-content: space-between;
  align-items: flex-start;
  gap: 14px;
  padding-bottom: 10px;
  border-bottom: 1px solid var(--line);
  min-width: 0;
}

.info-row:last-child {
  border-bottom: 0;
  padding-bottom: 0;
}

.info-row span:first-child {
  color: var(--muted);
  flex: 0 0 auto;
}

.info-row strong {
  min-width: 0;
  max-width: 100%;
  text-align: right;
  overflow-wrap: anywhere;
  word-break: break-word;
}

.description-text {
  margin: 0;
  color: var(--text);
  white-space: pre-wrap;
}

.site-footer {
  border-top: 1px solid var(--line);
  background: #172033;
  color: #dbeafe;
}

.footer-inner {
  width: min(1180px, calc(100% - 32px));
  min-height: 78px;
  margin: 0 auto;
  display: flex;
  align-items: center;
  justify-content: space-between;
  gap: 18px;
  flex-wrap: wrap;
}

.muted,
.empty,
.error-message {
  color: var(--muted);
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				182
Змн.	Арк.	№ докум.	Підпис	Дата		

```

}

.error-message {
  display: none;
  padding: 10px 12px;
  margin-bottom: 14px;
  border-radius: var(--radius);
  background: #fee2e2;
  color: #991b1b;
}

.success-message {
  display: none;
  padding: 10px 12px;
  margin-bottom: 14px;
  border-radius: var(--radius);
  background: #dcfce7;
  color: #166534;
  overflow-wrap: anywhere;
}

.success-message a {
  color: #14532d;
  font-weight: 800;
  text-decoration: underline;
}

.center-text {
  text-align: center;
}

.mt-8 {
  margin-top: 8px;
}

.mt-24 {
  margin-top: 24px;
}

@media (max-width: 760px) {
  .nav,
  .footer-inner {
    align-items: flex-start;
    flex-direction: column;
    padding: 14px 0;
  }

  .nav-links {
    justify-content: flex-start;
  }

  .toolbar,
  .form-grid,
  .detail-layout {
    grid-template-columns: 1fr;
  }

  .toolbar-actions {
    grid-column: 1;
  }

  .info-row {
    flex-direction: column;
  }

```

```

        gap: 4px;
    }

    .info-row strong {
        text-align: left;
    }

    .image-lightbox {
        padding: 72px 16px 64px;
    }

    .lightbox-close {
        top: 14px;
        right: 14px;
    }

    .lightbox-zoom-controls {
        top: auto;
        bottom: 62px;
    }

    .lightbox-nav {
        width: 44px;
        height: 44px;
    }

    .lightbox-prev {
        left: 12px;
    }

    .lightbox-next {
        right: 12px;
    }
}

```

		Поліщук І. А.			ДУ «Житомирська політехніка».26.121.18.000 – ПП	Арк.
		Вакалюк Т. Г.				184
Змн.	Арк.	№ докум.	Підпис	Дата		