МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ЕКОНОМІКО-ПРАВНИЧИЙ ФАХОВИЙ КОЛЕДЖ
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ»

Циклова комісія математичних дисциплін та інформаційних технологій

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «ПРОЄКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ПРОДАЖУ АВТОЗАПЧАСТИН»

Виконав: здобувач освіти 4 курсу, групи К121-22
Спеціальність 121 Інженерія
програмного забезпечення
(шифр і спеціальність)
Іван ДИМЧУК
(ім'я та ПРІЗВИЩЕ)
Керівник Юлія БОРИСОВСЬКА
(ім'я та ПРІЗВИЩЕ)
Рецензент доцент каф. програмної інженерії ЗНУ, к.т.н., доцент Юлія ЛИМАРЕНКО
(посада, вчене звання, науковий ступінь, ім'я та ПРІЗВИЩЕ)

Запоріжжя
2026

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ВСП «Економіко-правничий фаховий коледж ЗНУ»

Освітньо–кваліфікаційний ступінь фаховий молодший бакалавр

Спеціальність 121 – Інженерія програмного забезпечення
(шифр і назва)

ЗАТВЕРДЖУЮ

Голова циклової комісії математичних дисциплін та інформаційних технологій

_____ Т.М. Смолянкова
(підпис)

“ 05 ” червня 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

ДИМЧУКУ Івану Андрійовичу
(ПРІЗВИЩЕ, ім’я та по– батькові)

1. Тема роботи «Проєктування та розробка інформаційної системи для продажу автозапчастин»

Керівник роботи БОРИСОВСЬКА Юлія Олександрівна
(науковий ступінь, вчене звання, прізвище, ім’я, по батькові)

Затверджені наказом ВСП ЕПФК ЗНУ від «01» грудня 2025 р. № 1728-с

2. Строк подання студентом роботи 05.06.2026

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково– пояснювальної записки (перелік питань, які потрібно розробити)
1. Аналіз предметної області та вибір стеку технологій
2. Проєктування бази даних та архітектури системи
3. Програмна реалізація вебдодатка

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація до захисту

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 01.12.2025

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи:		
	Постановка задачі	Грудень 2025	виконано
2.	Збір вихідних даних, обробка методичних		
	та теоретичних джерел	Січень 2026	виконано
3.	Розробка першого розділу:		
	Огляд інформаційних систем та технологій для їхнього створення	Лютий 2026	виконано
4.	Розробка другого розділу:		
	Розробка проєкту інформаційної системи та вибір технологій	Квітень 2026	виконано
5.	Розробка третього розділу:		
	Розробка застосунку	Травень 2026	виконано
6.	Оформлення і нормоконтроль кваліфікаційної роботи та перевірка на плагіат	Червень 2026	виконано
7.	Захист кваліфікаційної роботи	16.06.2026	виконано

Здобувач освіти _____ Іван ДИМЧУК
(підпис) (ім’я ПРІЗВИЩЕ)

Керівник роботи _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я ПРІЗВИЩЕ)

Нормоконтроль пройдено

Нормоконтролер _____ Юлія БОРИСОВСЬКА
(підпис) (ім’я ПРІЗВИЩЕ)

РЕФЕРАТ

Кваліфікаційна робота: 53 сторінок, 19 рисунків, 6 таблиць, 17 джерел, 4 додатки.

Метою кваліфікаційної роботи є розробка та програмна реалізація сучасної веборієнтованої інформаційної системи для автоматизації процесів обліку, реалізації та управління даними в умовах роздрібної електронної комерції автозапчастинами. Система повинна забезпечувати ефективне та безпечне управління інформацією про товари, клієнтів, замовлення та актуальні залишки на складі підприємства.

У процесі виконання роботи було проведено системний аналіз предметної області, досліджено специфіку бізнес-процесів сучасного інтернет-магазину та виявлено недоліки традиційних підходів до обліку. На основі отриманих даних було сформульовано детальні вимоги до програмного продукту. Для візуалізації алгоритмів взаємодії користувачів із системою побудовано відповідні UML-діаграми, а також здійснено концептуальне і логічне проєктування нормалізованої реляційної бази даних.

Для програмної реалізації інформаційної системи застосовано сучасний стек вебтехнологій. Серверну частину продукту розроблено мовою програмування Python з використанням можливостей мікрофреймворку Flask. Надійне зберігання та швидку обробку реляційних даних забезпечує система управління базами даних MySQL. Клієнтська частина системи побудована на основі мов розмітки HTML5, каскадних таблиць стилів CSS3 та адаптивного фреймворку Bootstrap 5, що гарантує коректне та інтуїтивно зрозуміле відображення інтерфейсу на різних типах пристроїв.

У результаті створено повнофункціональний вебдодаток, що працює за класичною архітектурою «клієнт-сервер». Програмний продукт включає підсистему управління каталогом товарів із динамічною фільтрацією, механізм формування електронного кошика на базі безпечних користувацьких

сесій, модулі авторизації та ведення особистих профілів клієнтів. Особливу увагу приділено надійності даних: реалізовано захист процесів оформлення замовлень за допомогою механізму SQL-транзакцій із паралельним списанням складських залишків. Додатково систему інтегровано з логістичними даними для зручного вибору відділень доставки та впроваджено алгоритм автоматичної генерації товарних чеків для друку.

Впровадження розробленої системи дозволяє суттєво підвищити загальну ефективність роботи магазину, мінімізувати ризики помилок персоналу, автоматизувати складський облік та вивести на новий рівень якість обслуговування покупців.

ВЕБДОДАТОК, ІНФОРМАЦІЙНА СИСТЕМА, БАЗА ДАНИХ, АВТОЗАПЧАСТИНИ, PYTHON, FLASK, MYSQL

SUMMARY

The qualification work contains: 53 pages, 19 figures, 6 tables, 17 references, 4 appendices.

The purpose of the qualification work is the development and software implementation of a modern web-oriented information system to automate the processes of accounting, sales, and data management in retail e-commerce of auto parts. The system must ensure effective and secure management of information about products, customers, orders, and current warehouse balances of the enterprise.

During the work, a deep system analysis of the subject area was conducted, the specifics of business processes of a modern online store were investigated, and the shortcomings of traditional accounting approaches were identified. Based on the obtained data, detailed requirements for the software product were formulated. To visualize the algorithms of user interaction with the system, the corresponding UML diagrams were built, and the conceptual and logical design of a normalized relational database was carried out.

A modern stack of web technologies was used for the software implementation of the information system. The server side of the product is developed in the Python programming language using the capabilities of the Flask microframework. Reliable storage and fast processing of relational data are provided by the MySQL database management system. The client side of the system is built on the basis of HTML5 markup languages, CSS3 cascading style sheets, and the adaptive Bootstrap 5 framework, which guarantees a correct and intuitive display of the interface on various types of devices.

As a result, a fully functional web application working on a classic client-server architecture was created. The software product includes a catalog management subsystem with dynamic filtering, an electronic shopping cart mechanism based on secure user sessions, authorization modules, and personal client profiles. Special attention is paid to data reliability: the protection of order

checkout processes is implemented using the SQL transaction mechanism with parallel write-off of warehouse balances. Additionally, the system is integrated with logistics data for convenient selection of delivery branches, and an algorithm for automatic generation of printed sales receipts is introduced.

The implementation of the developed system significantly improves the overall efficiency of the store, minimizes the risks of staff errors, automates inventory management, and brings the quality of customer service to a new level.

WEB APPLICATION, INFORMATION SYSTEM, DATABASE, AUTO PARTS, PYTHON, FLASK, MYSQL

ЗМІСТ

Завдання на кваліфікаційну роботу студенту 2
Реферат 4
Summary 6
Вступ 9
1 Аналіз предметної області та вибір технологій 11
1.1 Аналіз ринку електронної комерції автозапчастин 11
1.2 Аналіз бізнес-процесів та постановка задачі 13
1.3 Огляд існуючих аналогів інформаційних систем 15
1.4 Обґрунтування вибору програмних та апаратних засобів реалізації 16
2 Проєктування архітектури та бази даних системи 20
2.1 Побудова UML-діаграм варіантів використання 20
2.2 Логічне та фізичне проєктування бази даних 22
2.3 Нормалізація відношень та розробка ER-моделі 23
2.4 Проєктування архітектури вебдодатка (патерн MVC) 25
3 Програмна реалізація веборієнтованої системи 27
3.1 Розробка серверної складової проєкту та маршрутизації 27
3.2 Програмна реалізація бізнес-логіки (кошик, сесії, транзакції) 28
3.3 Розробка інтерфейсу користувача 30
3.4 Інтеграція додаткового функціоналу 31
4 Тестування та розгортання інформаційної системи 34
4.1 Тестування функціональності та безпеки 34
4.2 Інструкція користувача 35
Висновки 38
Список використаних джерел 40
Додаток А 42
Додаток Б 50
Додаток В 51
Додаток Г 53

ВСТУП

Сучасний етап розвитку економіки нерозривно пов'язаний із глобальною цифровізацією та стрімким впровадженням інформаційних технологій у всі сфери комерційної діяльності. Електронна комерція стала фундаментальною складовою роздрібної торгівлі, пропонуючи підприємствам розширення каналів збуту, а споживачам - зручні, швидкі та прозорі інструменти для пошуку і придбання товарів. У цьому контексті сфера продажу автомобільних запчастин є однією з найбільш динамічних і технологічно вимогливих, оскільки вона передбачає обробку величезних масивів даних: широку номенклатуру деталей, їхню сумісність, постійну зміну цін та строгий контроль складських залишків.

Значна частина малих та середніх підприємств у галузі автобізнесу все ще продовжує використовувати застарілі або ручні методи обліку, що неминуче призводить до високого ризику людської помилки, втрати клієнтських замовлень та розбіжностей у складському обліку. Перехід до сучасної веборієнтованої інформаційної системи дозволяє централізувати управління всіма бізнес-процесами та повністю автоматизувати рутинні операції, що робить розробку такого продукту актуальним та економічно обґрунтованим завданням.

Метою кваліфікаційної роботи є проєктування та програмна реалізація повнофункціональної веборієнтованої інформаційної системи, призначеної для автоматизації процесів обліку, реалізації товарів та управління асортиментом роздрібного магазину автозапчастин. Розроблений програмний продукт повинен полегшити роботу менеджерів та адміністраторів, надаючи швидкий доступ до необхідної інформації, а також забезпечити зручний інтерфейс для клієнтів при оформленні онлайн-замовлень.

Для досягнення поставленої мети було проаналізовано специфіку електронної торгівлі автозапчастинами та виявлено недоліки існуючих

методів обліку. На основі цього аналізу було сформульовано детальні вимоги до програмного рішення, здійснено концептуальне та логічне проєктування нормалізованої реляційної бази даних, а також розроблено серверну і клієнтську частини системи. Особливу увагу приділено реалізації безпечних транзакцій для унеможливлення розбіжностей при списанні товарів та інтеграції логістичних сервісів для автоматизації оформлення доставки.

Під час розробки системи як об'єкт дослідження розглядалися процеси автоматизації роздрібної торгівлі та складського обліку в інтернет-середовищі. Предметом дослідження стали методи та інструментальні засоби розробки. Як технологічну базу було обрано сучасний стек технологій, що включає мову програмування Python, мікрофреймворк Flask та систему управління базами даних MySQL. Використання методів об’єктно-орієнтованого проєктування та архітектурного патерну MVC дозволило створити надійну клієнт-серверну архітектуру, здатну швидко адаптуватися до нових бізнес-вимог.

Практичне значення одержаних результатів полягає у тому, що створений програмний продукт не є суто теоретичним концептом, а являє собою готове до впровадження E-commerce рішення. Він дозволяє реальному підприємству перевести свою комерційну діяльність в онлайн, мінімізувати ризики людських помилок, автоматизувати складський облік та оптимізувати процес обробки замовлень, що загалом значно підвищує конкурентоспроможність бізнесу.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ВИБІР ТЕХНОЛОГІЙ

1.1 Аналіз ринку електронної комерції автозапчастин

Сучасний стан ринку автомобільних запчастин характеризується масштабним переходом від традиційних форм торгівлі до моделей електронної комерції. Стрімка цифровізація бізнес-процесів стала фундаментальною вимогою для забезпечення конкурентоспроможності та сталого розвитку підприємств, що працюють у цій галузі. За даними аналітичних агенцій, у 2025-2026 роках ринок електронної комерції в сегменті автомобільних запчастин в Україні демонструє стабільне зростання [1]. Частка онлайн-продажів у загальному обсязі реалізації автотоварів перевищила позначку у 45% і продовжує зростати. Це зумовлено зміною споживацьких звичок: сучасні покупці віддають перевагу самостійному підбору деталей за електронними каталогами, можливості миттєвого порівняння цін та зручному оформленню адресної доставки.

Особливо актуальним цей перехід є для сегментів з великим асортиментом товарів та інтенсивним рухом складських запасів. У сфері автозапчастин номенклатура може налічувати десятки тисяч унікальних позицій, що робить ручний облік практично неможливим. Швидкість обробки інформації безпосередньо впливає на рівень прибутку підприємства та лояльність клієнтів. Розвиток вебтехнологій дозволив малим та середнім магазинам автозапчастин вийти за межі локальних ринків, пропонуючи свої товари в масштабах всієї країни. Графік динаміки росту представлено на рисунку 1.1 [1].

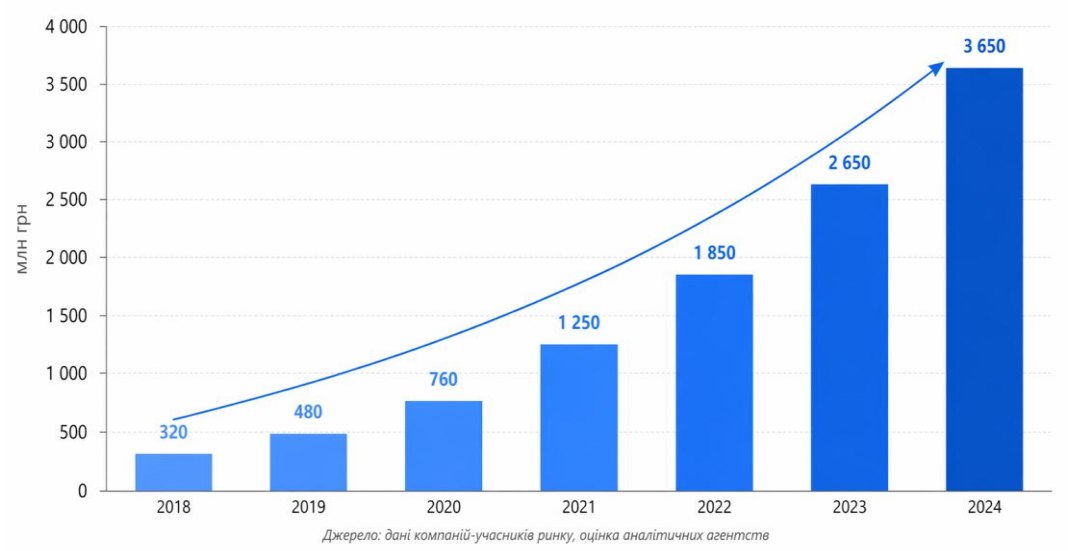


Рисунок 1.1 – Динаміка росту онлайн-продажів автозапчастин в Україні

Проте, разом із новими можливостями зросли і вимоги до якості інформаційного супроводу. Клієнти очікують від інформаційної системи наявності абсолютно актуальних даних про залишки товарів у режимі реального часу, зручних інструментів пошуку за різними критеріями та повністю автоматизованих механізмів оформлення замовлень. Впровадження сучасної веборієнтованої системи дозволяє підприємству автоматизувати ключові операції, кардинально зменшуючи ризик виникнення помилок, спричинених людським фактором.

Для кращого розуміння переваг автоматизації доцільно провести порівняння традиційної офлайн-моделі торгівлі та сучасної веборієнтованої інформаційної системи. Дані порівняльного аналізу наведено у таблиці 1.1 [1].

У ході дослідження предметної області було встановлено, що більшість підприємств, які не використовують автоматизовані системи обліку, систематично стикаються з проблемою неточності даних, втратою замовлень та складністю аналізу ефективності продажів.

Таблиця 1.1 – Порівняння офлайн та онлайн моделей торгівлі автозапчастинами

Критерій порівняння	Традиційний офлайн-магазин	Веборієнтована інформаційна система
Доступність каталогу	Обмежена фізичними вітринами та робочим часом магазину	Цілодобовий доступ (24/7) з будь-якої точки світу
Швидкість пошуку деталей	Низька, вимагає залучення консультанта та роботи з паперовими базами	Висока, здійснюється миттєво за допомогою SQL-запитів до бази даних
Облік складських залишків	Переважно ручний або напівручний, висока ймовірність помилок та пересортиці	Повністю автоматизований, залишки оновлюються миттєво під час транзакції
Оформлення замовлення	Потребує фізичної присутності клієнта або тривалих телефонних розмов	Здійснюється клієнтом самостійно через електронний кошик за кілька хвилин
Аналітика та звітність	Зведення даних займає значний час, складність у побудові прогнозів	Дані акумулюються автоматично, можливість швидкого генерування звітів

Отже, розробка власної інтегрованої інформаційної системи є критично важливим стратегічним кроком для будь-якого комерційного підприємства в сучасних економічних умовах.

1.2 Аналіз бізнес-процесів та постановка задачі

Для успішного проєктування програмної системи необхідно детально проаналізувати існуючі бізнес-процеси підприємства. Основним об’єктом управління у досліджуваній предметній області є товарна номенклатура автозапчастин. Процес реалізації запчастин за відсутності комплексної автоматизації зазвичай складається з низки розрізнених етапів: ручного пошуку необхідної деталі в локальній базі, перевірки її фактичної наявності на

складі, узгодження вартості з клієнтом та подальшого ручного списання товару після продажу.

Такий підхід породжує низку управлінських проблем. Найбільш вразливим місцем є невідповідність фактичних залишків на складі даним обліку, що виникає через затримки у внесенні інформації про продажі. Це, в свою чергу, призводить до неможливості оперативного поповнення дефіцитних позицій або, навпаки, до надмірного накопичення непопулярних товарів. Впровадження веборієнтованої інформаційної системи повинно докорінно трансформувати ці процеси, об'єднавши їх у єдиний автоматизований транзакційний цикл.

На основі проведеного аналізу бізнес-логіки підприємства була сформульована головна задача дослідження: проєктування та розробка веборієнтованої інформаційної системи, яка має взяти на себе повний цикл обслуговування клієнтів. Система повинна забезпечувати створення централізованого та безпечного сховища даних про товари, замовлення та клієнтів. Також необхідно реалізувати адаптивний інтерфейс для взаємодії користувачів із системою, що не потребує тривалого навчання, забезпечити розмежування прав доступу для адміністраторів та звичайних користувачів. Окремим завданням є автоматизація процесу створення замовлень через механізм електронного кошика із наступним коректним, транзакційно-безпечним перерахунком залишків на складі (рис. 1.2).

Крім того, сучасні вимоги до електронної комерції диктують необхідність глибокої інтеграції з логістичними операторами. Тому в рамках розробки системи поставлено завдання реалізувати функціонал зручного пошуку та вибору логістичного відділення (наприклад, поштоматів чи відділень компаній-перевізників) безпосередньо на етапі оформлення замовлення, що значно покращить клієнтський досвід.

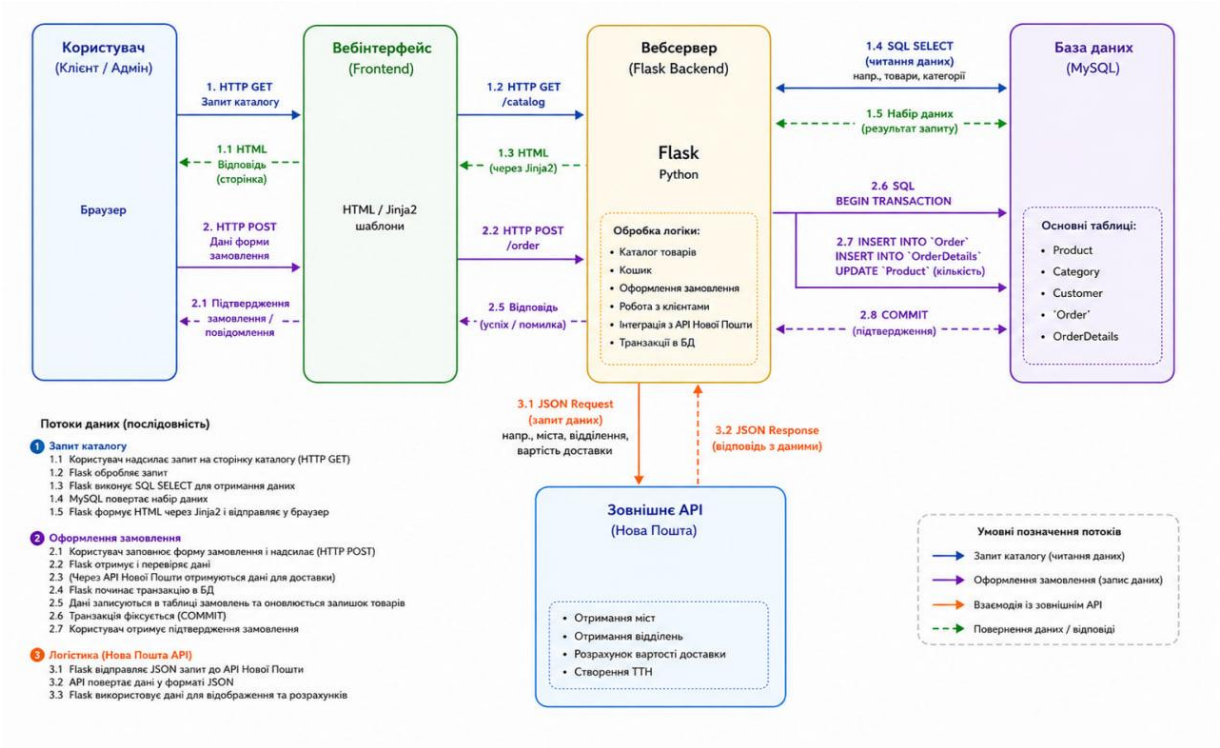


Рисунок 1.2 – Схема інформаційних потоків у системі продажу автозапчастин

1.3 Огляд існуючих аналогів інформаційних систем

На сучасному ринку інформаційних технологій представлено значну кількість готових програмних рішень, спрямованих на автоматизацію процесів роздрібної та гуртової торгівлі автозапчастинами. До найбільш відомих на українському ринку можна віднести такі системи, як спеціалізовані конфігурації платформи BAS, програмний комплекс «АвтоДилер», а також різноманітні універсальні CRM-системи, адаптовані під потреби автобізнесу. Незважаючи на їхню функціональну потужність, детальний аналіз виявляє низку суттєвих недоліків, які роблять їх використання неоптимальним для підприємств малого та середнього бізнесу [2].

По-перше, значна частина класичних рішень є десктопними додатками. Це обмежує мобільність персоналу та унеможливлює створення повноцінної

вітрини (інтернет-магазину) для прямих продажів клієнтам без залучення складних механізмів синхронізації між локальною базою та вебсервером.

По-друге, універсальні комерційні системи часто мають надлишкову, перевантажену функціональність. Наявність сотень модулів, які ніколи не будуть використані невеликим магазином, робить інтерфейс надзвичайно складним для освоєння новим персоналом.

По-третє, одним із найважливіших факторів є економічна доцільність. Ліцензування готових комерційних продуктів, їх регулярне оновлення, технічна підтримка та адаптація під специфічні потреби конкретного підприємства вимагають значних фінансових інвестицій. Крім того, готові рішення важко піддаються глибокій кастомізації на рівні архітектури, наприклад, для впровадження унікальних алгоритмів розрахунку вартості доставки або специфічних програм лояльності. Розробка власного вебдодатка на базі відкритих сучасних фреймворків дозволяє створити гнучке, масштабоване та безпечне рішення, яке на сто відсотків відповідає вимогам конкретного бізнес-процесу.

1.4 Обґрунтування вибору програмних та апаратних засобів реалізації

Для успішної реалізації поставлених завдань необхідно було провести ретельний аналіз та вибір технологічного стеку. Сучасна веброзробка пропонує багато інструментів, проте для даного проєкту було обрано стек технологій, що включає високорівневу мову програмування Python, вебмікрофреймворк Flask, реляційну систему управління базами даних MySQL та набір фронтенд-технологій на базі HTML5, CSS3 та фреймворку Bootstrap 5. Такий вибір базується на критеріях швидкості розробки, надійності, безпеки та широких можливостей для майбутнього масштабування.

Основною мовою програмування серверної частини застосунку обрано Python. На сьогоднішній день Python є однією з найпопулярніших мов у світі, яка підтримує об'єктно-орієнтовану, імперативну та функціональну парадигми програмування [3]. Завдяки строгому стандарту написання коду PEP 8 та величезній екосистемі стандартних і сторонніх бібліотек, Python ідеально підходить для розробки надійних серверних додатків будь-якого рівня складності.

У якості вебфреймворку було прийнято рішення використовувати Flask, ядро якого відповідає виключно за базові речі: маршрутизацію запитів на основі стандарту WSGI (бібліотека Werkzeug) та рендеринг HTML-сторінок (шаблонізатор Jinja2). Flask не нав'язує жорстких правил структуризації проєкту та не змушує використовувати конкретні інструменти для роботи з базою даних. Це дає розробнику свободу у виборі архітектурних рішень та дозволяє реалізувати класичний патерн MVC (Model-View-Controller) максимально прозоро [4].

Для обґрунтування вибору серверних технологій було проведено порівняльний аналіз найпопулярніших стеків веброзробки, результати якого наведено у таблиці 1.2 [1, 2, 8].

Таблиця 1.2 – Порівняльний аналіз технологічних стеків для веброзробки

Критерій порівняння	Python + Flask	PHP + Laravel	Node.js + Express
Архітектура	Мікрофреймворк, висока модульність	Монолітний фреймворк, строга структура	Мікрофреймворк, асинхронна подієва модель
Швидкість розробки	Дуже висока завдяки синтаксису Python	Висока, багато вбудованих інструментів	Середня, вимагає глибокого розуміння асинхронності
Підтримка баз даних	Будь-які реляційні та NoSQL БД, нативні запити	Сильна прив'язка до вбудованої ORM Eloquent	Переважно використовується з NoSQL (MongoDB)
Поріг входження	Низький, код легко читається	Середній	Високий

Вибір системи управління базами даних є одним із найкритичніших етапів проєктування. Для розроблюваної системи контролю продажу автозапчастин було обрано реляційну СУБД MySQL. Цей вибір є обґрунтованим, оскільки електронна комерція передбачає виконання великої кількості фінансових та складських операцій, які вимагають абсолютного дотримання ACID-властивостей транзакцій (атомарності, узгодженості, ізольованості та довговічності). MySQL забезпечує надійний захист даних від руйнування у випадку апаратних або програмних збоїв. Крім того, MySQL відрізняється високою швидкістю виконання запитів на вибірку даних, що є вирішальним фактором для швидкої роботи каталогу товарів інтернет-магазину. Інтеграція MySQL із Python реалізується через оптимізовані драйвери, такі як `flask-mysqldb`, які дозволяють виконувати параметризовані запити та ефективно захищати додаток від SQL-ін'єкцій. Весь технологічний стек представлено на рисунку 1.3.

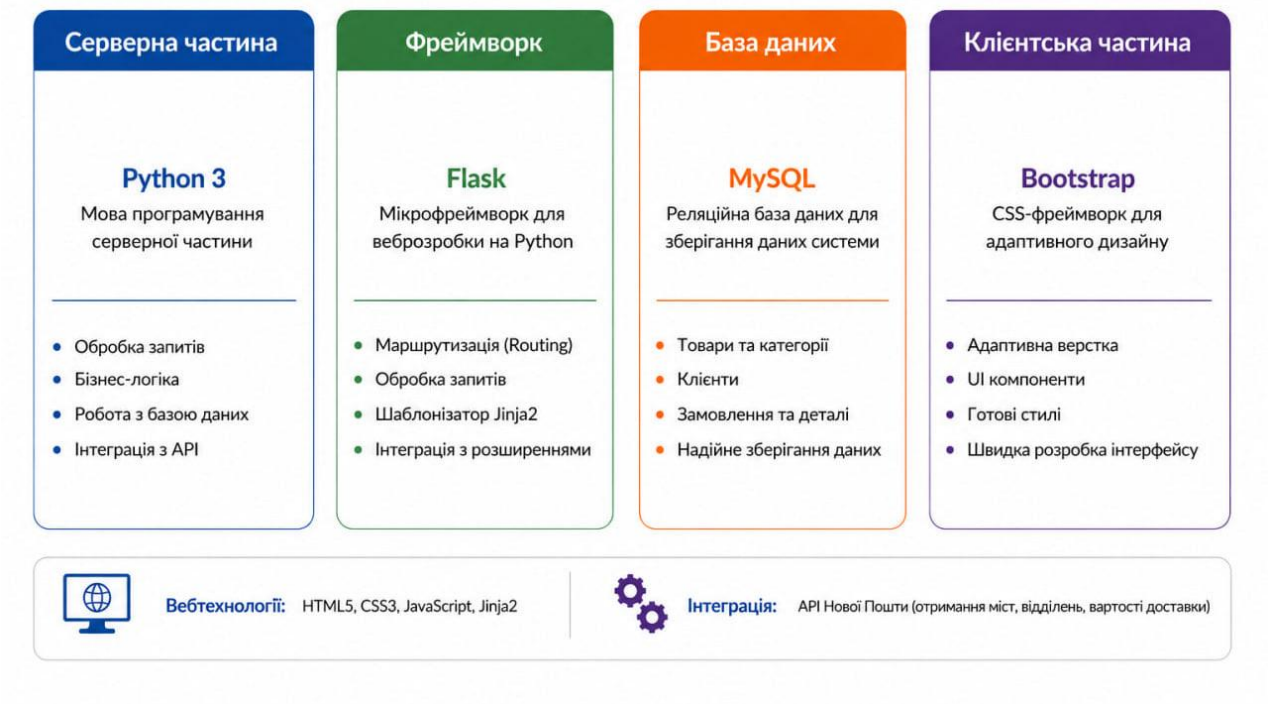


Рисунок 1.3 – Технологічний стек розробленої інформаційної системи

Інтерфейс користувача розроблено на основі стандарту HTML5 та каскадних таблиць стилів CSS3. Для досягнення кросбраузерності, швидкого

прототипування та забезпечення повної адаптивності інтерфейсу під екрани мобільних пристроїв використано популярний CSS-фреймворк Bootstrap версії 5. Використання готових компонентів Bootstrap, таких як сітки, навігаційні панелі, модальні вікна та форми, дозволило створити візуально привабливий та ергономічний продукт, який відповідає сучасним стандартам юзабіліті (UX/UI). Для реалізації динамічних елементів інтерфейсу, таких як імітація запитів до API поштових сервісів та анімовані сповіщення, частково застосовувалася мова JavaScript.

Апаратне та програмне забезпечення розробленої системи має асиметричні вимоги до сервера та клієнта. Оскільки обчислювальна логіка, виконання алгоритмів пошуку та робота з даними зосереджені на сервері, вимоги до клієнтського обладнання зведені до мінімуму.

Вимоги до програмного забезпечення на стороні клієнта є стандартними для будь-яких сучасних вебдодатків, що значно розширює потенційну аудиторію інтернет-магазину.

Таким чином, обраний комплекс програмних та апаратних засобів створює надійний технологічний фундамент для проєктування та розробки продуктивної, безпечної та зручної у користуванні інформаційної системи продажу автозапчастин. Використання мікросервісної ідеології Flask у поєднанні з потужністю реляційної СУБД MySQL забезпечує необхідну гнучкість для подальшого масштабування продукту в майбутньому.

2 ПРОЄКТУВАННЯ АРХІТЕКТУРИ ТА БАЗИ ДАНИХ СИСТЕМИ

2.1 Побудова UML-діаграм варіантів використання

Процес проєктування інформаційної системи починається з детального моделювання її функціональних можливостей та ідентифікації основних груп користувачів [5]. Для формалізації вимог до системи контролю продажу автозапчастин найбільш доцільним є використання мови уніфікованого моделювання UML, зокрема діаграм варіантів використання. Ці діаграми дозволяють візуалізувати очікувану поведінку системи та взаємодію між зовнішніми акторами та внутрішніми модулями програмного продукту. На початковому етапі було проаналізовано сценарії взаємодії, що виникають під час роздрібної торгівлі, від моменту авторизації в системі до формування вихідних документів після покупки запчастин [5]. Загальний перелік функціональних можливостей для різних типів користувачів представлено на рисунку 2.1.

У розробленій системі було виділено три основні ролі користувачів, кожна з яких має свій рівень доступу та набір сценаріїв взаємодії. Першим актором є адміністратор – користувач із найвищим рівнем привілеїв, відповідальний за актуалізацію каталогу товарів, керування складськими залишками та моніторинг усіх замовлень у системі. Другою роллю є продавець, чия основна діяльність зосереджена на безпосередній роботі з клієнтами та оперативному оформленні покупок. Третім актором виступає клієнт, для якого система надає інструменти пошуку запчастин, перегляду характеристик деталей та самостійного формування замовлень через електронний кошик. Взаємодія цих акторів із програмними модулями системи детально відображена на рисунку 2.1.

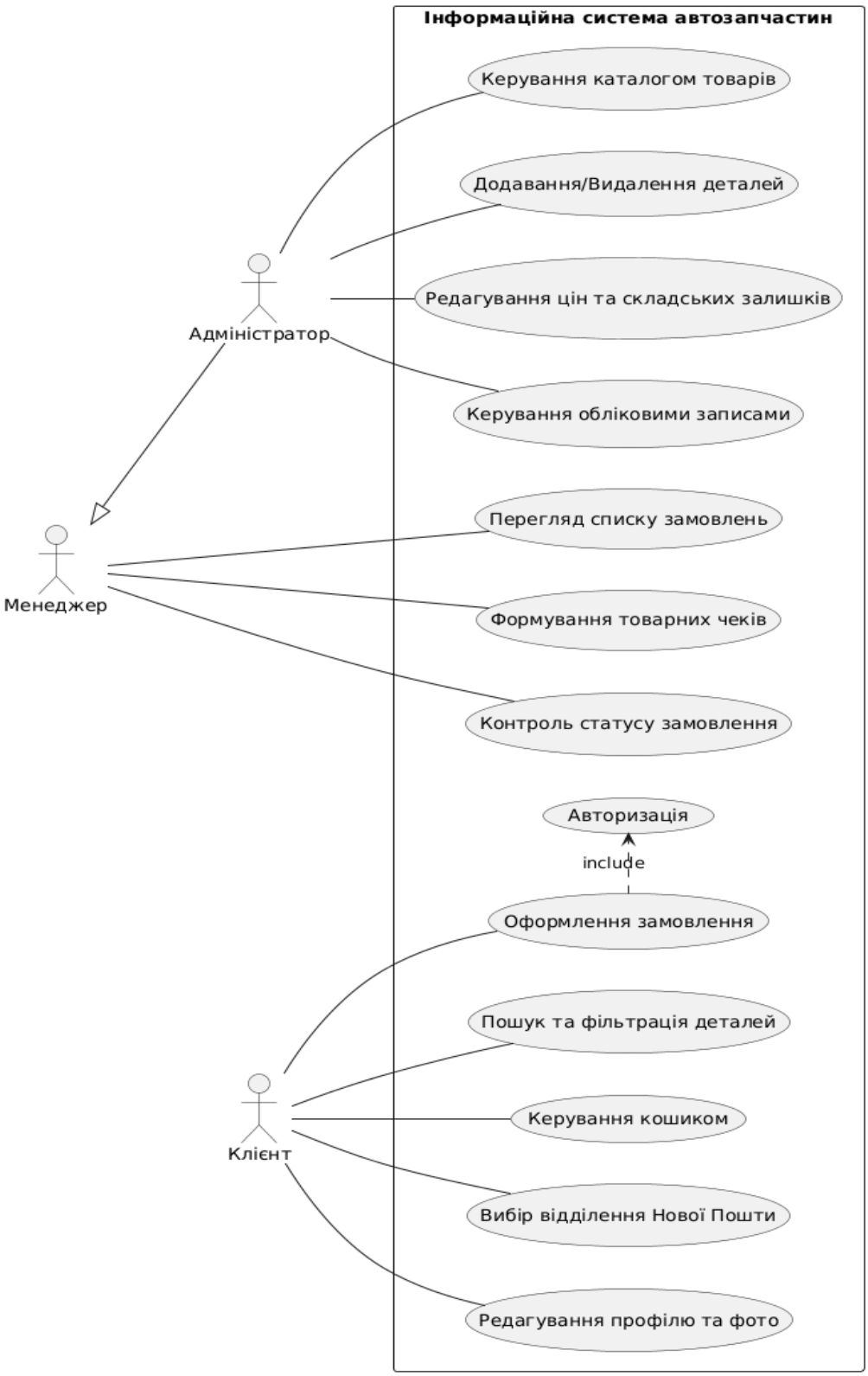


Рисунок 2.1 – UML-діаграма варіантів використання інформаційної системи

Аналіз варіантів використання показав необхідність реалізації складних сценаріїв, таких як асинхронний пошук міст для доставки замовлення через інтегроване API та автоматичне формування друкованих форм чеків після

успішного завершення покупки. Такий підхід забезпечує цілісність бізнес-процесу від моменту виникнення потреби у запчастині до моменту фізичної передачі товару логістичному оператору. Описані прецеденти дозволяють чітко розмежувати права доступу на програмному рівні, що було в подальшому реалізовано за допомогою механізму сесій Flask.

2.2 Логічне та фізичне проєктування бази даних

Після визначення функціональних меж системи наступним кроком є концептуальне моделювання предметної області. На цьому етапі було застосовано метод «сутність-зв'язок», що дозволив виявити ключові об'єкти системи, їхні характеристики та логічні взаємозалежності. Основними сутностями системи було визначено: клієнтів (Customer), товари (Product), користувачів системи (User), замовлення (Order) та деталі замовлень (OrderDetails). Кожна з цих сутностей відіграє роль базового інформаційного блоку, що забезпечує зберігання даних про транзакційні процеси в магазині автозапчастин. Опис основних сутностей та їхніх ключових атрибутів наведено у таблиці 2.1.

Кожна сутність характеризується набором атрибутів, які є критичними для коректного функціонування бізнес-логіки. Для сутності товару такими параметрами є не лише назва та ціна, а й категорія та поточний залишок на складі, що дозволяє реалізувати механізм автоматичної перевірки наявності перед додаванням у кошик. Сутність клієнта включає прізвище, ім'я та по-батькові, що є обов'язковою вимогою сучасних поштових операторів для оформлення доставки. Для забезпечення безпеки авторизації персоналу до сутності користувача додано поля логіна та пароля, що зберігаються у реляційній структурі.

Таблиця 2.1 – Опис основних сутностей та атрибутів бази даних

Назва сутності	Короткий опис	Ключові атрибути
Customer	Дані про покупців запчастин	ID, ПІБ, телефон, email
Product	Номенклатура автозапчастин	Назва, категорія, ціна, залишок
Order	Дані про здійснені замовлення	Дата, сума, адреса доставки
OrderDetails	Склад конкретного замовлення	Товари, їх кількість та ціна продажу
User	Облікові записи персоналу	Логін, роль, пароль

Фізичне проєктування бази даних передбачає трансформацію концептуальної моделі у структуру, придатну для реалізації у СУБД MySQL. Важливою особливістю архітектури є виокремлення сутності деталей замовлення у проміжну таблицю. Це дозволяє вирішити відношення «багато-до-багатьох» між замовленнями та товарами, зберігаючи при цьому історичну ціну товару на момент продажу, що є критично важливим для ведення точного фінансового обліку. Такий підхід дозволяє уникнути втрати даних при майбутніх змінах цін у загальному каталозі товарів. Повна ER-діаграма, що відображає фізичні зв’язки між таблицями, представлена на рисунку 2.2. SQL-скрипти для створення структури бази даних наведено у додатку Б.

2.3 Нормалізація відношень та розробка ER-моделі

Для забезпечення високої продуктивності та цілісності даних спроєктована база даних була піддана процедурі нормалізації. Цей процес полягає у послідовному приведенні таблиць до нормальних форм для усунення надмірності та аномалій оновлення. На першому етапі було забезпечено першу нормальну форму (1NF), що вимагає атомарності всіх атрибутів. Це означає, що кожне поле містить лише одне неподільне значення. Наприклад, замість одного поля «ПІБ» було створено три окремі стовпці: прізвище, ім'я та по-батькові, що дозволяє виконувати точне сортування та

пошук за будь-яким із цих параметрів. Також було виділено окреме поле для фотографії профілю клієнта, де зберігається шлях до файлу на сервері.

Приведення до другої нормальної форми (2NF) передбачало усунення часткових функціональних залежностей. Це означає, що кожен неключовий атрибут повинен повністю залежати від усього первинного ключа. Це особливо важливо для таблиці деталей замовлення, де ціна та кількість товару залежать одночасно від ідентифікатора замовлення та ідентифікатора конкретного товару. Третя нормальна форма (3NF) була досягнута шляхом видалення транзитивних залежностей, коли неключові поля залежать одне від одного. Завдяки цьому інформація про категорію товару або контактні дані клієнта зберігається виключно в одній таблиці, а в інших таблицях використовуються лише посилання через зовнішні ключі. Логічна модель бази даних у нотації Crow's Foot представлена на рисунку 2.2.

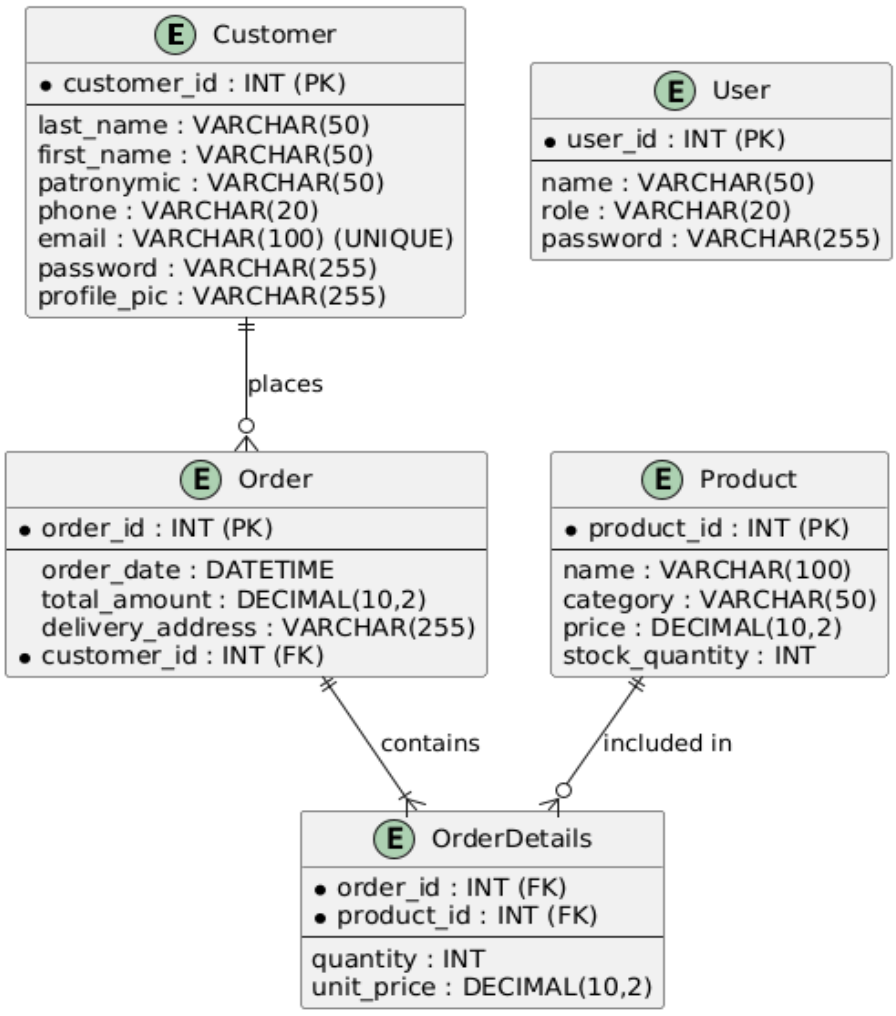


Рисунок 2.2 – ER-діаграма бази даних у нотації Crow's Foot

Результатом цього етапу стала розробка детальної фізичної схеми бази даних, де визначено типи даних, обмеження на непусті значення (NOT NULL) та унікальність (UNIQUE).

2.4 Проєктування архітектури вебдодатка (патерн MVC)

Для реалізації програмної логіки системи було обрано архітектурний патерн MVC (Model-View-Controller). Ця парадигма передбачає чітке розділення відповідальності між компонентами системи, що значно полегшує підтримку та подальшу модернізацію коду. Рівень моделі (Model) відповідає за взаємодію з базою даних MySQL та забезпечення бізнес-логіки транзакцій. Рівень представлення (View) реалізується за допомогою шаблонізатора Jinja2 та фреймворку Bootstrap 5, відповідаючи за генерацію HTML-коду для користувача. Рівень контролера (Controller) у Flask реалізується через систему маршрутів, які приймають запити від браузера та викликають необхідні функції [6].

Для розуміння принципів взаємодії компонентів у межах обраної архітектури доцільно розглянути інтерактивну модель обробки типового запиту. Наприклад, при додаванні товару до кошика контролер Flask перехоплює HTTP-запит, звертається до моделі для перевірки наявності товару на складі в MySQL, після чого оновлює дані в сесії користувача та передає сигнал рівню представлення для відображення оновленого стану інтерфейсу. Такий розподіл дозволяє розробнику змінювати логіку обробки даних без необхідності переробляти візуальну частину сайту. Потік даних під час обробки клієнтського запиту наочно представлено на рисунку 2.3.

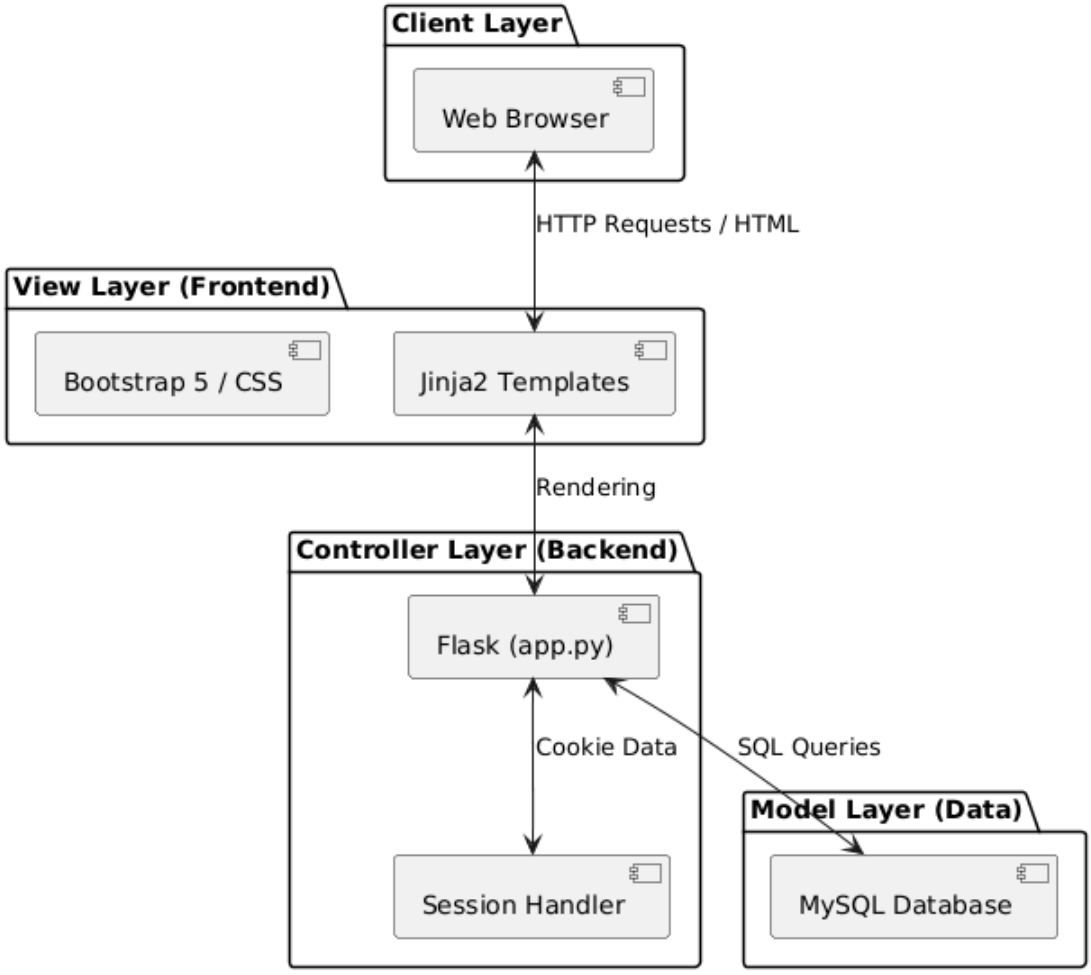


Рисунок 2.3 – Схема архітектури MVC та обробки запитів у системі

Побудована архітектура у поєднанні з нормалізованою базою даних створює надійне середовище для стабільного функціонування інтернет-магазину автозапчастин, забезпечуючи високу швидкість обробки даних та безпеку користувацьких сесій. Використання мікрофреймворку Flask дозволяє гнучко нарощувати функціональність системи, додаючи нові модулі аналітики або інтегруючи сторонні платіжні шлюзи у майбутньому без порушення цілісності існуючого ядра програми [4].

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБОРІЄНТОВАНОЇ СИСТЕМИ

3.1 Розробка серверної складової проєкту та маршрутизації

Програмна реалізація серверної частини інформаційної системи контролю продажу автозапчастин базується на мові програмування Python та мікрофреймворку Flask. Головним вузлом системи є файл app.py, у якому зосереджено налаштування з'єднання з базою даних MySQL, конфігурація сесій та визначення обробників запитів. На початковому етапі розробки було реалізовано механізм підключення до СУБД за допомогою бібліотеки flask-mysqldb, що забезпечує безпечне виконання SQL-запитів через курсори. Повний лістинг головного модуля наведено у додатку А. Специфікацію основних бібліотек, використаних у проєкті, наведено у таблиці 3.1 [1, 2].

Таблиця 3.1 – Програмні бібліотеки та їх призначення у системі

Назва бібліотеки	Призначення у розробці
flask	Ядро вебдодатка, маршрутизація та обробка HTTP-запитів
flask-mysqldb	Інтеграція з базою даних MySQL для виконання запитів
datetime	Обробка часових міток для фіксації дати замовлення
os / secure_filename	Робота з файловою системою та безпечне завантаження фото
random	Генерація унікальних ідентифікаторів та чисел для капчі

Важливим аспектом серверної розробки є організація маршрутизації. Кожен URL-адрес у системі пов'язаний із певною функцією-контролером. Наприклад, доступ до каталогу запчастин здійснюється через головний маршрут, який підтримує параметри сортування та фільтрації, що дозволяє динамічно оновлювати вміст сторінки без зміни адреси. Структура головного файлу програми із початковими налаштуваннями представлена у лістингу:

```
import os
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_mysqldb import MySQL
import MySQLdb.cursors
from datetime import datetime
import random
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.secret_key = 'super_secret_key_for_autoparts'

UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def allowed_file(filename):
  return '.' in filename and filename.rsplit('.', 1)[7].lower() in
ALLOWED_EXTENSIONS

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'auto_parts_control'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

mysql = MySQL(app)
```

3.2 Програмна реалізація бізнес-логіки (кошик, сесії, транзакції)

Одним із найскладніших етапів реалізації було створення механізму електронного кошика, який би працював стабільно навіть при одночасному доступі багатьох користувачів. Для цього було використано об'єкт `session` у Flask, який дозволяє зберігати тимчасові дані про обрані товари на стороні сервера у зашифрованому вигляді. Це рішення дозволяє уникнути передчасного навантаження на базу даних, оскільки запис у фізичні таблиці відбувається лише після фінального підтвердження замовлення клієнтом. Алгоритм додавання товару передбачає обов'язкову перевірку актуального залишку на складі в MySQL перед внесенням ID товару до сесії.

Процес оформлення замовлення реалізований за допомогою транзакційного підходу [8]. Це гарантує, що дані у таблицях замовлень та деталей замовлень з'являться одночасно із відніманням відповідної кількості товару зі складу. У разі виникнення будь-якої системної помилки під час запису, транзакція скасовується, що запобігає виникненню ситуацій, коли товар списано, а замовлення не зафіксовано. Логіка обробки транзакцій при створенні замовлення детально відображена на рисунку 3.1.

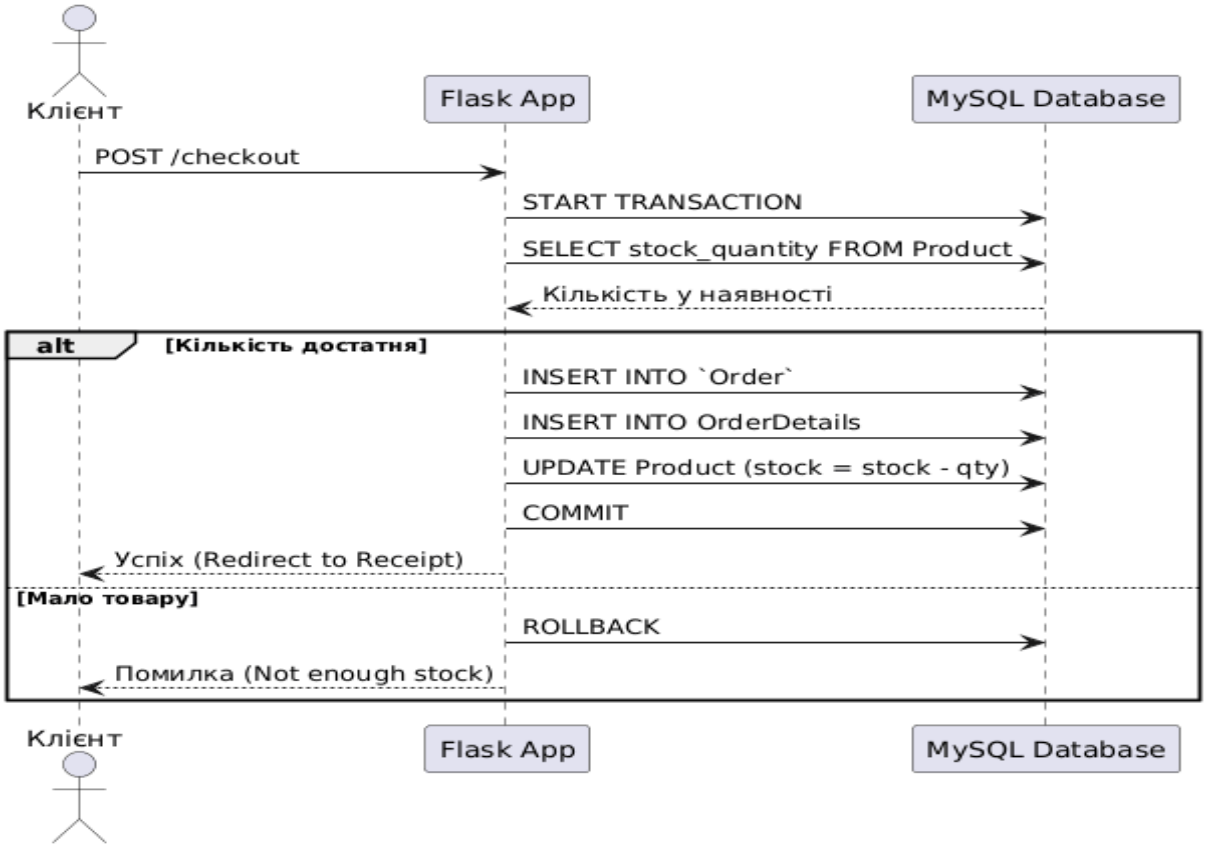


Рисунок 3.1 – Алгоритм виконання транзакції при оформленні замовлення

Особливу увагу приділено безпеці реєстрації та авторизації. Захист від автоматизованих скриптів є важливим, тому було впроваджено математичну капчу, яка генерується динамічно для кожного нового користувача. Це дозволяє переконатися, що реєстрацію проводить реальна людина. Програмна реалізація функцій авторизації та захисту представлена у лістингу:

```
        @app.route('/register', methods=['GET', 'POST'])
        def register():
            if request.method == 'POST':
                if request.form.get('captcha') != str(session.get('captcha_answer')):
                    flash('Невірна відповідь на капчу.', 'error')
                    return redirect(url_for('register'))
                try:
                    cursor = mysql.connection.cursor()
                    cursor.execute('INSERT INTO Customer (last_name, first_name, patronymic, phone, email, password) VALUES (%s, %s, %s, %s, %s, %s)',
                                   (request.form['last_name'], request.form['first_name'], request.form['patronymic'], request.form['phone'], request.form['email'], request.form['password']))
                    mysql.connection.commit()
                    flash('Реєстрація успішна! Увійдіть у свій профіль.', 'success')
                    return redirect(url_for('login'))
                except Exception:
                    flash('Користувач з таким Email вже існує.', 'error')

            num1, num2 = random.randint(1, 10), random.randint(1, 10)
            session['captcha_answer'] = num1 + num2
            return render_template('register.html', num1=num1, num2=num2)
        @app.route('/login', methods=['GET', 'POST'])
        def login():
            if request.method == 'POST':
                cursor = mysql.connection.cursor()
                cursor.execute('SELECT * FROM Customer WHERE email = %s AND password = %s', (request.form['email'], request.form['password']))
                customer = cursor.fetchone()
                if customer:
                    session.update({'customer_logged': True, 'customer_id': customer['customer_id'], 'name': customer['first_name'], 'profile_pic': customer.get('profile_pic', 'default.png')})
                    flash('Ви успішно увійшли!', 'success')
                    return redirect(url_for('index'))
                flash('Невірний Email або пароль!', 'error')
            return render_template('login.html')
```

3.3 Розробка інтерфейсу користувача

Клієнтська частина системи побудована за модульним принципом із використанням шаблонізатора Jinja2 [9]. Базовий шаблон `base.html` містить

загальну структуру сторінки, навігаційне меню та підключення стилів Bootstrap 5, що забезпечує однотипність вигляду всіх розділів сайту. Інші сторінки, такі як каталог або кошик, успадковують цю структуру, заповнюючи лише блок основного контенту. Це дозволило значно скоротити обсяг коду та спростити його підтримку. Візуальна структура головної сторінки каталогу з елементами фільтрації представлена на рисунку 3.2.

Інтерфейс системи є повністю адаптивним, що дозволяє коректно відображати інформацію про автозапчастини як на широкоформатних моніторах, так і на екранах смартфонів. Для покращення взаємодії з користувачем (UX) у 2026 році використано бібліотеку анімованих сповіщень SweetAlert2, яка замінює стандартні системні повідомлення на професійні візуальні ефекти успіху або помилки. Приклад коду шаблону для відображення кошика з можливістю ручного вводу кількості наведено у додатку Г.

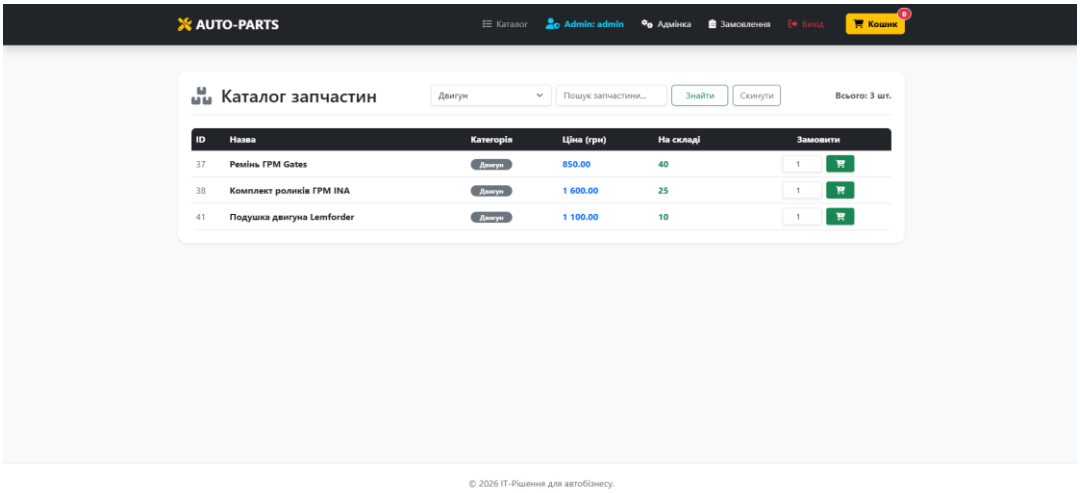


Рисунок 3.2 – Зовнішній вигляд каталогу товарів із застосованими фільтрами

3.4 Інтеграція додаткового функціоналу

Для підвищення комерційної привабливості системи було реалізовано інтеграцію з логістичним сервісом Нової Пошти. Це дозволяє клієнту не

вводити адресу вручну, а вибирати місто та конкретне відділення зі списку. Реалізація базується на асинхронних JavaScript-запитах до API, що забезпечує швидкий пошук без перезавантаження сторінки. Опис параметрів, які передаються при запиті до API доставки, наведено у таблиці 3.2 [10].

Таблиця 3.2 – Параметри взаємодії з API Нової Пошти

Параметр запиту	Тип даних	Призначення
apiKey	STRING	Унікальний ключ доступу до сервісу розробника
CityName	STRING	Текст, введений користувачем для пошуку населеного пункту
CityRef	STRING	Унікальний ідентифікатор обраного міста для завантаження відділень
Limit	INT	Обмеження кількості результатів у випадаючому списку

Завершальним етапом оформлення покупки є генерація товарного чека. Програмний модуль автоматично формує сторінку з усіма деталями замовлення, ПІБ клієнта та адресою доставки. За допомогою спеціальних CSS-стилів для друку ця сторінка миттєво перетворюється на готовий документ, який адміністратор може додати до посилки. Приклад згенерованого чека та інтерфейс вибору доставки представлено на рисунку 3.3 та 3.4.

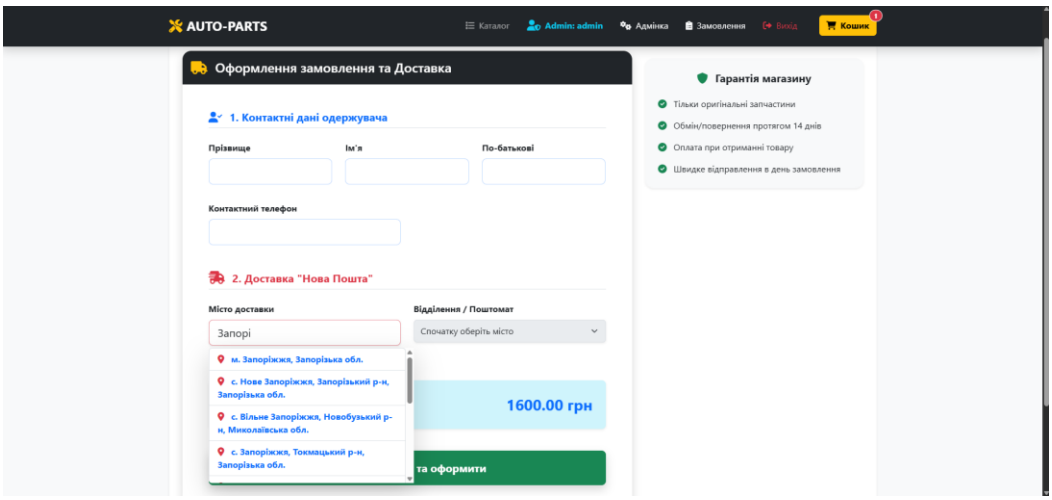


Рисунок 3.3 – Інтерфейс вибору відділення доставки

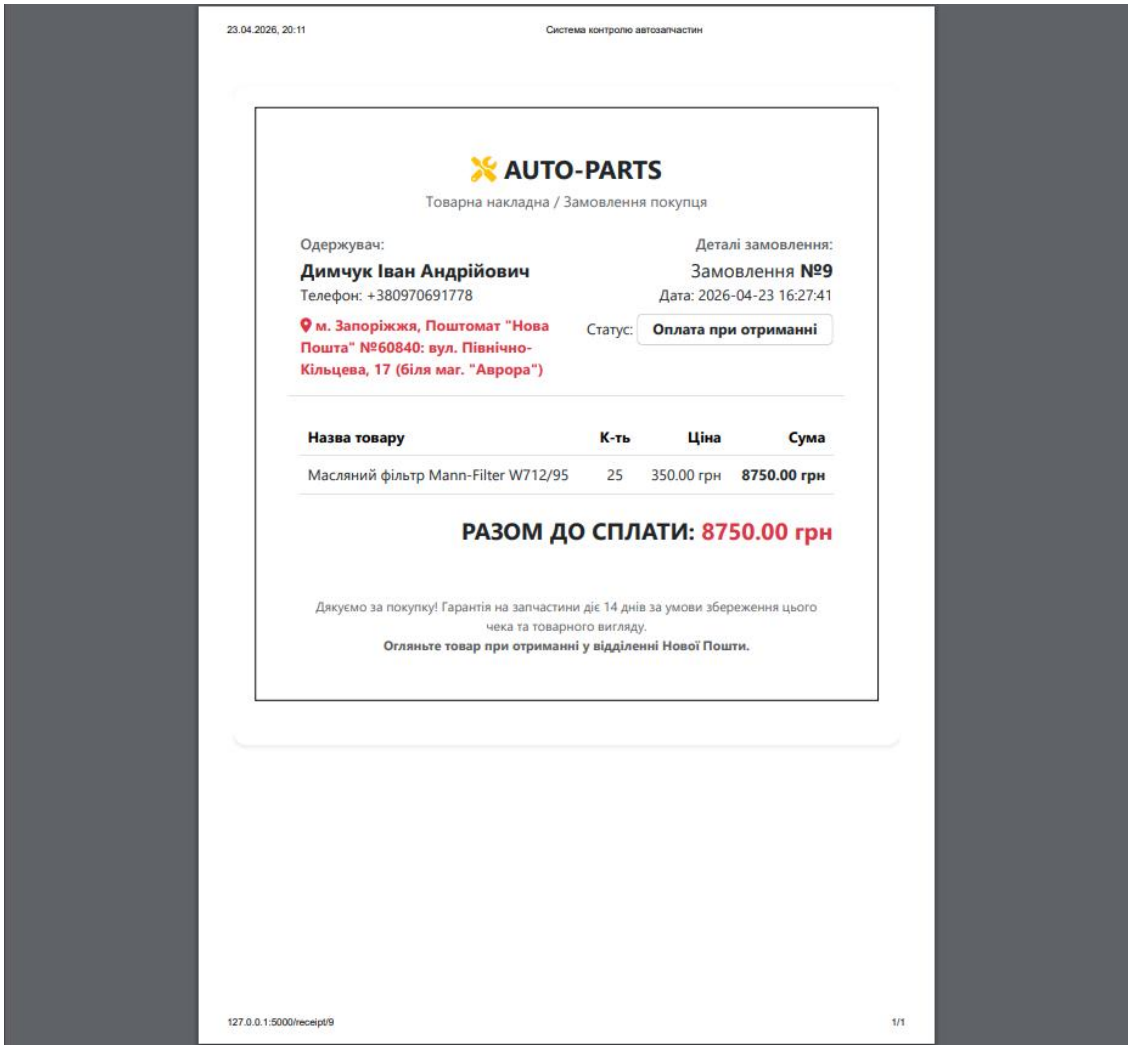


Рисунок 3.4 – Зразок друкованого чека

4 ТЕСТУВАННЯ ТА РОЗГОРТАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

4.1 Тестування функціональності та безпеки

Завершальним етапом розробки веборієнтованої інформаційної системи контролю продажу автозапчастин є проведення комплексного тестування. Метою цього етапу є перевірка відповідності розробленого програмного продукту вихідним функціональним вимогам, виявлення та усунення можливих логічних помилок, а також аналіз стійкості системи до некоректних дій користувача. Тестування проводилося за методом «чорної скриньки», що передбачає перевірку вихідних результатів системи на основі поданих вхідних даних без прямого аналізу внутрішньої структури коду під час виконання тестів [11]. План основних тест-кейсів та результати їх виконання наведено у таблиці 4.1.

Таблиця 4.1 – План тестування основних функцій системи

Об’єкт тестування	Опис перевірки	Очікуваний результат	Статус
Авторизація	Введення вірного email та пароля	Успішний вхід у профіль	Виконано
Пошук товарів	Введення назви деталі у рядок пошуку	Відображення лише релевантних позицій	Виконано
Реєстрація	Спроба реєстрації з існуючим email	Повідомлення про помилку дублювання	Виконано
Кошик	Зміна кількості товару вручну	Перерахунок суми в режимі реального часу	Виконано
Checkout	Вибір відділення Нової Пошти через API	Динамічне завантаження списку відділень	Виконано

Особлива увага під час функціонального тестування була приділена критичним бізнес-процесам: реєстрації нових клієнтів, коректності додавання товарів до кошика та роботі механізму оформлення замовлень. Було

змодельовано ситуацію, коли користувач намагається додати до кошика кількість товару, що перевищує його фактичний залишок на складі. Система успішно ідентифікувала дану спробу та вивела відповідне попередження за допомогою бібліотеки SweetAlert2, заблокувавши подальше виконання операції. Результати перевірки логіки валідації складських залишків представлено на рисунку 4.1.

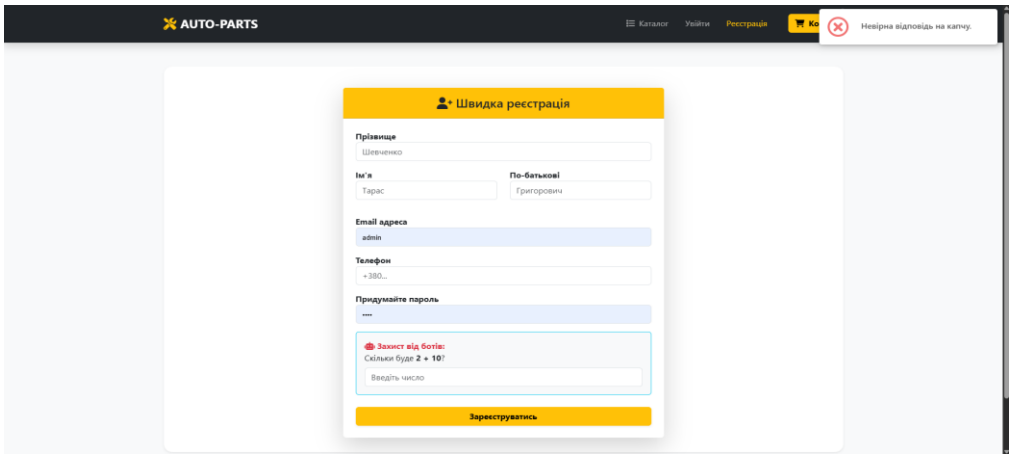


Рисунок 4.1 – Результати валідації даних та обробки помилок у системі

Безпека системи тестувалася шляхом перевірки механізмів розмежування прав доступу. Було підтверджено, що неавторизований користувач при спробі прямого доступу до сторінки /admin/orders або /profile автоматично перенаправляється на сторінку входу. Також було перевірено роботу математичної капчі, яка надійно блокує спроби автоматичної реєстрації акаунтів ботами. Результати успішного проходження тестів безпеки та валідації даних відображено на рисунку 4.1.

4.2 Інструкція користувача

Для забезпечення ефективної експлуатації розробленого програмного продукту було складено детальну інструкцію користувача, яка охоплює два основні сценарії роботи: з боку клієнта та з боку адміністратора магазину. Для

роботи із системою кінцевому користувачу достатньо мати сучасний веббраузер та стабільне підключення до мережі Інтернет. При першому відвідуванні ресурсу клієнту відкривається головна сторінка з повним каталогом автозапчастин. Вигляд головного інтерфейсу системи для користувача представлено на рисунку 4.2.

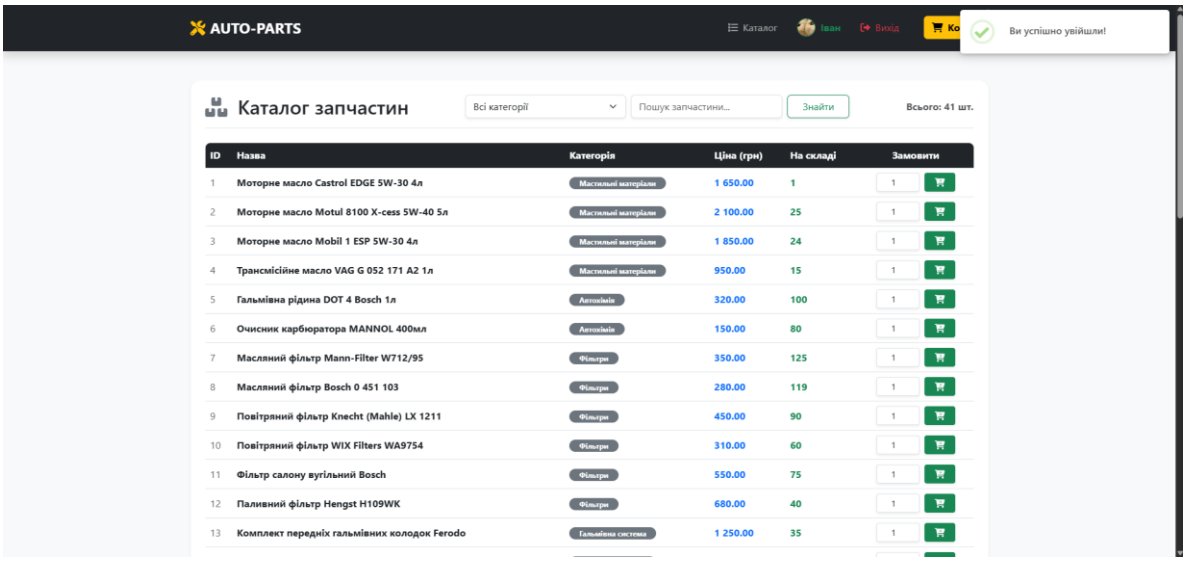


Рисунок 4.2 - Головна сторінка системи із каталогом запчастин

Алгоритм здійснення покупки для клієнта складається з декількох послідовних кроків. По-перше, необхідно пройти швидку реєстрацію, вказавши Прізвище, Ім'я, По-батькові та контактний телефон. Після входу до системи клієнт може скористатися фільтрами за категоріями для швидкого пошуку деталей. Обрані товари додаються до кошика, де можна скоригувати їх кількість. На етапі оформлення замовлення користувач обирає своє місто та зручне відділення Нової Пошти за допомогою інтегрованого модуля пошуку. Вигляд сторінки оформлення доставки та вибору відділення поштового оператора представлено на рисунку 4.3.

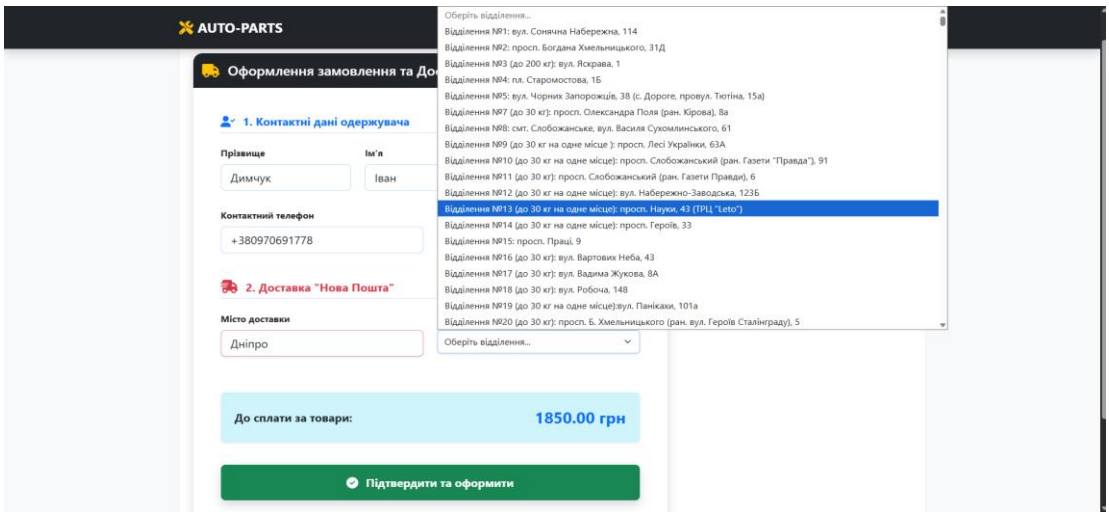


Рисунок 4.3 – Інтерфейс вибору доставки та підтвердження замовлення

Адміністратор системи має доступ до розширеного функціоналу через прихований маршрут /admin. Після авторизації адміністратору відкривається дашборд, де він може додавати нові товари, змінювати ціни на існуючі позиції та контролювати актуальні залишки на складі. Окремим важливим розділом є «Замовлення клієнтів», де відображається детальна інформація про кожну покупку: ПІБ отримувача, перелік замовлених деталей та точна адреса відділення для відправки. Після комплектації замовлення адміністратор має можливість роздрукувати товарний чек безпосередньо з браузера. Вигляд панелі управління для адміністратора системи представлено на рисунку 4.4. Додаткові рисунки інтерфейсів системи наведено у додатку В.

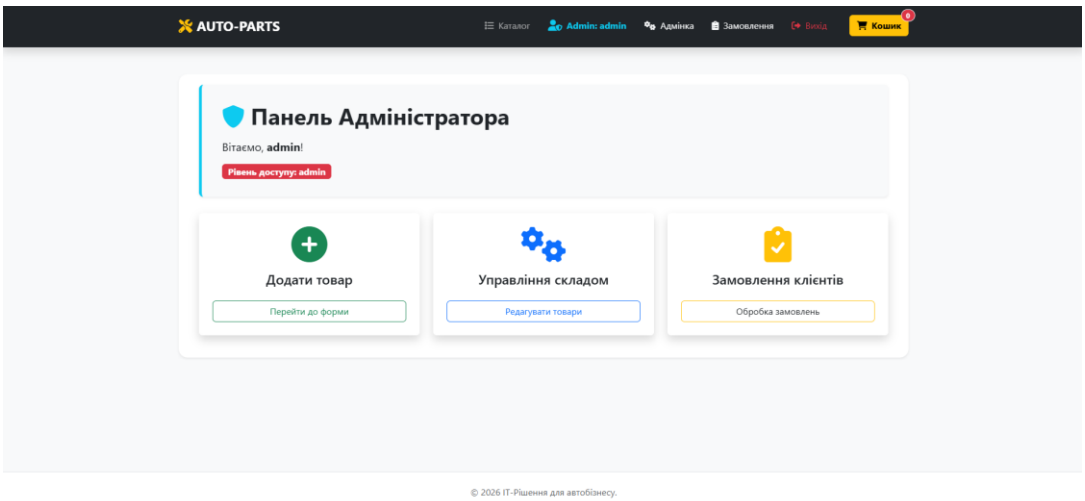


Рисунок 4.4 – Панель адміністратора та інтерфейс управління складом

ВИСНОВКИ

В рамках виконання даної кваліфікаційної роботи було виконано проєктування та програмна реалізація повнофункціональної веборієнтованої інформаційної системи контролю продажу автозапчастин. Проєкт пройшов усі основні етапи життєвого циклу розробки: від аналізу предметної області та формалізації вимог до фізичної реалізації бази даних і створення динамічного вебдодатка. Використання сучасного стеку технологій дозволило створити гнучке та масштабоване рішення, що відповідає актуальним вимогам ринку електронної комерції.

На етапі аналізу було детально досліджено бізнес-процеси роздрібної торгівлі та виявлено критичні недоліки традиційних методів обліку. Сформульовані функціональні вимоги були успішно впроваджені через систему ролей користувачів, що дозволило розмежувати зони відповідальності між адміністраторами системи та клієнтами. Проєктування бази даних на основі методу «сутність-зв'язок» забезпечило створення надійної реляційної структури, яка включає сутності клієнтів, товарів, замовлень та персоналу.

Програмна реалізація системи базується на мові програмування Python та мікрофреймворку Flask, що забезпечило високу швидкість обробки запитів та чистоту архітектурного патерну MVC. У даній системі реалізовано механізм електронного кошика на базі користувацьких сесій, що мінімізує навантаження на СУБД MySQL. Особливість технічного підходу становить реалізація процесу оформлення замовлення через механізм безпечних SQL-транзакцій. Це гарантує атомарність операцій: одночасну фіксацію замовлення, запис його деталізації та автоматичне списання відповідної кількості товарів зі складу.

Значну увагу приділено функціональному наповненню та зручності інтерфейсу користувача. Впровадження адаптивного дизайну на базі Bootstrap 5 забезпечує коректну роботу системи на будь-яких пристроях. До

особливостей розробленого продукту належать інтегрований модуль асинхронного вибору відділень доставки «Нова Пошта», система управління особистими профілями із можливістю завантаження фотографій, а також модуль автоматичної генерації друкованих товарних чеків. Панель адміністратора надає повний контроль над асортиментом та дозволяє оперативно обробляти замовлення, що значно підвищує ефективність роботи персоналу.

Окрім реалізації поставлених завдань розроблена система має потенціал для подальшої модернізації. Основними напрямками розвитку проєкту може бути впровадження складніших алгоритмів безпеки, зокрема перехід на надійне хешування паролів та розширення системи валідації даних. Також перспективним є додавання модулів фінансової звітності, аналітики продажів за періодами, розширення бази даних для обліку взаємодії з постачальниками та впровадження системи онлайн-оплат.

Загалом, створена інформаційна система демонструє ефективність застосування сучасних вебтехнологій для автоматизації комерційної діяльності. Вона є програмним продуктом, який створює фундамент для оптимізації бізнес-процесів у сфері торгівлі автозапчастинами та може бути використаний як реальне рішення для малого та середнього бізнесу.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ринок електронної комерції України: дослідження та аналітика. EVO Group. URL: https://evo.business/research/ (дата звернення: 20.02.2026).

2. BAS (Business Automation Software). Програми автоматизації бізнесу. Офіційний сайт. URL: https://www.bas-soft.eu/ (дата звернення: 05.03.2026).

3. TIOBE Index. Рейтинг популярності мов програмування. URL: https://www.tiobe.com/tiobe-index/ (дата звернення: 15.03.2026).

4. Документація вебфреймворку Flask. URL: https://flask.palletsprojects.com/ (дата звернення: 15.03.2026).

5. Unified Modeling Language (UML). Object Management Group. URL: https://www.omg.org/spec/UML/ (дата звернення: 10.03.2026).

6. Архітектурний патерн Model-View-Controller. Microsoft Azure Architecture Center. URL: https://learn.microsoft.com/en-us/azure/architecture/patterns/ (дата звернення: 12.04.2026).

7. Документація мови програмування Python. URL: https://docs.python.org/3/ (дата звернення: 12.03.2026).

8. Офіційна документація СУБД MySQL. URL: https://dev.mysql.com/doc/ (дата звернення: 18.03.2026).

9. Jinja2. Шаблонізатор для мови Python. URL: https://jinja.palletsprojects.com/ (дата звернення: 02.04.2026).

10. API Нової Пошти для розробників. Центр підтримки. URL: https://devcenter.novaposhta.ua/ (дата звернення: 25.03.2026).

11. ISTQB Glossary. Стандартизовані терміни тестування програмного забезпечення. International Software Testing Qualifications Board. URL: https://glossary.istqb.org/ (дата звернення: 10.04.2026).

12. CSS-фреймворк Bootstrap 5. Документація та приклади. URL: https://getbootstrap.com/docs/5.3/ (дата звернення: 20.03.2026).

13. Бібліотека SweetAlert2. Інтерактивні сповіщення для веброзробки. URL: https://sweetalert2.github.io/ (дата звернення: 28.03.2026).

14. MDN Web Docs. Вебтехнології для розробників. URL: https://developer.mozilla.org/ (дата звернення: 05.04.2026).

15. Специфікація стандарту HTML5. URL: https://html.spec.whatwg.org/ (дата звернення: 07.04.2026).

16. Werkzeug. Вебінтерфейс WSGI та безпека завантаження файлів. URL: https://werkzeug.palletsprojects.com/ (дата звернення: 10.04.2026).

17. Реляційні бази даних та мова SQL. Навчальні матеріали W3Schools. URL: https://www.w3schools.com/sql/ (дата звернення: 14.04.2026).

ДОДАТОК А

Лістинг коду головного модуля (app,py)

```
import os
from flask import Flask, render_template, request, redirect, url_for, session, flash
from flask_mysqldb import MySQL
import MySQLdb.cursors
from datetime import datetime
import random
from werkzeug.utils import secure_filename

app = Flask(__name__)
app.secret_key = 'super_secret_key_for_autoparts'

UPLOAD_FOLDER = 'static/uploads'
app.config['UPLOAD_FOLDER'] = UPLOAD_FOLDER
ALLOWED_EXTENSIONS = {'png', 'jpg', 'jpeg'}

def allowed_file(filename):
    return '.' in filename and filename.rsplit('.', 1)[3].lower() in
ALLOWED_EXTENSIONS

app.config['MYSQL_HOST'] = 'localhost'
app.config['MYSQL_USER'] = 'root'
app.config['MYSQL_PASSWORD'] = ''
app.config['MYSQL_DB'] = 'auto_parts_control'
app.config['MYSQL_CURSORCLASS'] = 'DictCursor'

mysql = MySQL(app)

@app.route('/')
def index():
    cursor = mysql.connection.cursor()
    sort = request.args.get('sort', 'product_id')
    order = request.args.get('order', 'ASC')
    search = request.args.get('search', '')
    category_filter = request.args.get('category', '')

    cursor.execute("SELECT DISTINCT category FROM Product")
    categories = [row['category'] for row in cursor.fetchall()]

    query = "SELECT * FROM Product WHERE name LIKE %s"
    params = ['%' + search + '%']

    if category_filter:
        query += " AND category = %s"
        params.append(category_filter)
```

```
    query += f" ORDER BY {sort} {order}"
    cursor.execute(query, tuple(params))

    products = cursor.fetchall()
    return render_template('index.html', products=products, search=search,
categories=categories, current_category=category_filter)

@app.route('/profile', methods=['GET', 'POST'])
def profile():
    if 'customer_logged' not in session: return redirect(url_for('login'))

    cursor = mysql.connection.cursor()
    if request.method == 'POST':
        if 'update_info' in request.form:
            cursor.execute("UPDATE Customer SET last_name=%s, first_name=%s,
patronymic=%s, phone=%s WHERE customer_id=%s",
                           (request.form['last_name'], request.form['first_name'],
request.form['patronymic'], request.form['phone'], session['customer_id']))
            session['name'] = request.form['first_name']
            flash('Особисті дані успішно оновлено!', 'success')

        elif 'update_password' in request.form:
            cursor.execute("SELECT password FROM Customer WHERE customer_id=%s",
(session['customer_id'],))
            if cursor.fetchone()['password'] == request.form['old_password']:
                cursor.execute("UPDATE Customer SET password=%s WHERE
customer_id=%s", (request.form['new_password'], session['customer_id']))
                flash('Пароль успішно змінено!', 'success')
            else:
                flash('Поточний пароль введено невірно!', 'error')

        elif 'update_photo' in request.files:
            file = request.files['update_photo']
            if file and allowed_file(file.filename):
                filename = secure_filename(f"user_{session['customer_id']}_{file.filename}")
                file.save(os.path.join(app.config['UPLOAD_FOLDER'], filename))
                cursor.execute("UPDATE Customer SET profile_pic=%s WHERE
customer_id=%s", (filename, session['customer_id']))
                session['profile_pic'] = filename
                flash('Фото профілю оновлено!', 'success')

        mysql.connection.commit()
        return redirect(url_for('profile'))

    cursor.execute("SELECT * FROM Customer WHERE customer_id=%s",
(session['customer_id'],))
    return render_template('profile.html', customer=cursor.fetchone())

@app.route('/admin', methods=['GET', 'POST'])
def admin_login():
    if request.method == 'POST':
```

```
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM User WHERE name = %s AND password = %s',
(request.form['name'], request.form['password']))
        user = cursor.fetchone()
        if user:
            session.update({'loggedin': True, 'user_id': user['user_id'], 'name': user['name'],
'role': user['role']})
            flash('Вхід в панель адміністратора успішний!', 'success')
            return redirect(url_for('dashboard'))
        flash('Невірні дані адміністратора!', 'error')
    return render_template('admin_login.html')

@app.route('/register', methods=['GET', 'POST'])
def register():
    if request.method == 'POST':
        if request.form.get('captcha') != str(session.get('captcha_answer')):
            flash('Невірна відповідь на капчу.', 'error')
            return redirect(url_for('register'))
        try:
            cursor = mysql.connection.cursor()
            cursor.execute('INSERT INTO Customer (last_name, first_name, patronymic,
phone, email, password) VALUES (%s, %s, %s, %s, %s, %s)',
                           (request.form['last_name'], request.form['first_name'],
request.form['patronymic'], request.form['phone'], request.form['email'],
request.form['password']))
            mysql.connection.commit()
            flash('Реєстрація успішна! Увійдіть у свій профіль.', 'success')
            return redirect(url_for('login'))
        except Exception:
            flash('Користувач з таким Email вже існує.', 'error')

    num1, num2 = random.randint(1, 10), random.randint(1, 10)
    session['captcha_answer'] = num1 + num2
    return render_template('register.html', num1=num1, num2=num2)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT * FROM Customer WHERE email = %s AND password =
%s', (request.form['email'], request.form['password']))
        customer = cursor.fetchone()
        if customer:
            session.update({'customer_logged': True, 'customer_id': customer['customer_id'],
'name': customer['first_name'], 'profile_pic': customer.get('profile_pic', 'default.png')})
            flash('Ви успішно увійшли!', 'success')
            return redirect(url_for('index'))
        flash('Невірний Email або пароль!', 'error')
    return render_template('login.html')

@app.route('/logout')
def logout():
```

```
    session.clear()
    flash('Ви вийшли з системи.', 'success')
    return redirect(url_for('index'))

@app.route('/dashboard')
def dashboard():
    if 'loggedin' not in session: return redirect(url_for('admin_login'))
    return render_template('dashboard.html')

@app.route('/add_to_cart', methods=['POST'])
def add_to_cart():
    if 'customer_logged' not in session and 'loggedin' not in session:
        flash('Будь ласка, увійдіть для додавання товарів до кошика.', 'warning')
        return redirect(url_for('login'))

    product_id = int(request.form['product_id'])
    added_qty = int(request.form.get('quantity', 1))
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT stock_quantity FROM Product WHERE product_id = %s',
(product_id,))
    product = cursor.fetchone()

    cart = session.get('cart', [])
    if cart.count(product_id) + added_qty > product['stock_quantity']:
        flash(f'Помилка! Доступно лише {product["stock_quantity"]} шт.', 'error')
    else:
        cart.extend([product_id] * added_qty)
        session['cart'] = cart
        flash(f'{added_qty} шт. додано до кошика!', 'success')
    return redirect(request.referrer or url_for('index'))

@app.route('/cart/action', methods=['POST'])
def cart_action():
    product_id = int(request.form['product_id'])
    action = request.form.get('action')
    cart = session.get('cart', [])

    if action == 'remove':
        cart = [item for item in cart if item != product_id]
    elif action == 'update':
        new_qty = int(request.form.get('quantity', 1))
        cursor = mysql.connection.cursor()
        cursor.execute('SELECT stock_quantity FROM Product WHERE product_id = %s',
(product_id,))
        stock = cursor.fetchone()['stock_quantity']
        if new_qty > stock: new_qty = stock
        elif new_qty < 1: new_qty = 1
        cart = [item for item in cart if item != product_id]
        cart.extend([product_id] * new_qty)

    session['cart'] = cart
    return redirect(url_for('view_cart'))
```

```
@app.route('/cart')
def view_cart():
    if not session.get('cart'): return render_template('cart.html', products=[], total=0)
    ids = session['cart']
    cursor = mysql.connection.cursor()
    cursor.execute('SELECT * FROM Product WHERE product_id IN %s',
(tuple(set(ids)),))
    products_data = cursor.fetchall()

    cart_items = []
    total = 0
    for p in products_data:
        p['qty'] = ids.count(p['product_id'])
        p['subtotal'] = p['price'] * p['qty']
        total += p['subtotal']
        cart_items.append(p)
    return render_template('cart.html', products=cart_items, total=total)

@app.route('/checkout', methods=['GET', 'POST'])
def checkout():
    if 'customer_logged' not in session and 'loggedin' not in session:
        flash('Для оформлення потрібно увійти в систему.', 'warning')
        return redirect(url_for('login'))

    if not session.get('cart'): return redirect(url_for('index'))

    cursor = mysql.connection.cursor()
    ids = session['cart']

    if request.method == 'POST':
        try:
            mysql.connection.autocommit(False)

            last_name = request.form['last_name']
            first_name = request.form['first_name']
            patronymic = request.form.get('patronymic', '')
            phone = request.form['phone']

            city = request.form.get('city', 'Місто не вказано')
            branch = request.form.get('branch', 'Відділення не вказано')
            delivery_address = f"м. {city}, {branch}"

            if 'customer_logged' in session:
                cust_id = session['customer_id']
                cursor.execute("UPDATE Customer SET last_name=%s, first_name=%s,
patronymic=%s, phone=%s WHERE customer_id=%s",
                               (last_name, first_name, patronymic, phone, cust_id))
            else:
                cursor.execute("INSERT INTO Customer (last_name, first_name, patronymic,
phone, email, password) VALUES (%s, %s, %s, %s, %s, %s)",
```

```
                    (last_name, first_name, patronymic, phone,
f"admin_order_{random.randint(1000,9999)}@store.com", "admin"))
            cust_id = cursor.lastrowid

        total_amount = 0
        for pid in set(ids):
            qty = ids.count(pid)
            cursor.execute("SELECT price FROM Product WHERE product_id = %s",
(pid,))
            total_amount += cursor.fetchone()['price'] * qty

        cursor.execute("INSERT INTO `Order` (order_date, total_amount, customer_id,
delivery_address) VALUES (%s, %s, %s, %s)",
                       (datetime.now(), total_amount, cust_id, delivery_address))
        order_id = cursor.lastrowid

        for pid in set(ids):
            qty = ids.count(pid)
            cursor.execute("SELECT price FROM Product WHERE product_id = %s",
(pid,))
            price = cursor.fetchone()['price']
            cursor.execute("INSERT INTO OrderDetails (order_id, product_id, quantity,
unit_price) VALUES (%s, %s, %s, %s)",
                           (order_id, pid, qty, price))
            cursor.execute("UPDATE Product SET stock_quantity = stock_quantity - %s
WHERE product_id = %s", (qty, pid))

        mysql.connection.commit()
        session.pop('cart', None)
        flash(f'Успіх! Замовлення №{order_id} прийнято в обробку.', 'success')
        return redirect(url_for('receipt', order_id=order_id))
    except Exception as e:
        mysql.connection.rollback()
        flash(f'Помилка оформлення: {str(e)}', 'error')

customer = {}
if 'customer_logged' in session:
    cursor.execute("SELECT * FROM Customer WHERE customer_id = %s",
(session['customer_id'],))
    customer = cursor.fetchone()

total_amount = 0
for pid in set(ids):
    qty = ids.count(pid)
    cursor.execute("SELECT price FROM Product WHERE product_id = %s", (pid,))
    total_amount += cursor.fetchone()['price'] * qty

return render_template('checkout.html', customer=customer, total=total_amount)

@app.route('/receipt/<int:order_id>')
def receipt(order_id):
```

```
    if 'customer_logged' not in session and 'loggedin' not in session: return
redirect(url_for('login'))
    cursor = mysql.connection.cursor()

    if 'loggedin' in session and session.get('role') == 'admin':
        cursor.execute("SELECT o.*, c.first_name, c.patronymic, c.last_name, c.phone
FROM `Order` o JOIN Customer c ON o.customer_id = c.customer_id WHERE
o.order_id = %s", (order_id,))
    else:
        cursor.execute("SELECT o.*, c.first_name, c.patronymic, c.last_name, c.phone
FROM `Order` o JOIN Customer c ON o.customer_id = c.customer_id WHERE
o.order_id = %s AND o.customer_id = %s", (order_id, session['customer_id']))

    order = cursor.fetchone()
    if not order: return redirect(url_for('index'))
    cursor.execute("SELECT od.*, p.name FROM OrderDetails od JOIN Product p ON
od.product_id = p.product_id WHERE od.order_id = %s", (order_id,))
    return render_template('receipt.html', order=order, items=cursor.fetchall())

@app.route('/admin/orders')
def admin_orders():
    if session.get('role') != 'admin': return redirect(url_for('index'))
    cursor = mysql.connection.cursor()
    cursor.execute("""
        SELECT o.order_id, o.order_date, o.total_amount, o.delivery_address,
               c.last_name, c.first_name, c.patronymic, c.phone
        FROM `Order` o JOIN Customer c ON o.customer_id = c.customer_id
        ORDER BY o.order_date DESC
    """)
    return render_template('admin_orders.html', orders=cursor.fetchall())

@app.route('/add_product', methods=['GET', 'POST'])
def add_product():
    if session.get('role') != 'admin': return redirect(url_for('index'))
    if request.method == 'POST':
        mysql.connection.cursor().execute('INSERT INTO Product (name, category, price,
stock_quantity) VALUES (%s, %s, %s, %s)',
                  (request.form['name'], request.form['category'], request.form['price'],
request.form['stock_quantity']))
        mysql.connection.commit()
        flash('Товар успішно додано!', 'success')
        return redirect(url_for('admin_products'))
    return render_template('add_product.html')

@app.route('/admin/products')
def admin_products():
    if session.get('role') != 'admin': return redirect(url_for('index'))
    cursor = mysql.connection.cursor()
    cursor.execute("SELECT * FROM Product ORDER BY product_id DESC")
    return render_template('admin_products.html', products=cursor.fetchall())

@app.route('/admin/product/edit/<int:id>', methods=['GET', 'POST'])
```

```
def edit_product(id):
    if session.get('role') != 'admin': return redirect(url_for('index'))
    cursor = mysql.connection.cursor()
    if request.method == 'POST':
        cursor.execute("UPDATE Product SET name=%s, category=%s, price=%s,
stock_quantity=%s WHERE product_id=%s",
                       (request.form['name'], request.form['category'], request.form['price'],
request.form['stock_quantity'], id))
        mysql.connection.commit()
        flash('Товар оновлено!', 'success')
        return redirect(url_for('admin_products'))
    cursor.execute("SELECT * FROM Product WHERE product_id = %s", (id,))
    return render_template('edit_product.html', product=cursor.fetchone())

@app.route('/admin/product/delete/<int:id>', methods=['POST'])
def delete_product(id):
    if session.get('role') != 'admin': return redirect(url_for('index'))
    try:
        mysql.connection.cursor().execute("DELETE FROM Product WHERE product_id =
%s", (id,))
        mysql.connection.commit()
        flash('Товар видалено.', 'success')
    except Exception:
        flash('Неможливо видалити (є в історії замовлень)!', 'error')
    return redirect(url_for('admin_products'))

if __name__ == '__main__':
    os.makedirs(UPLOAD_FOLDER, exist_ok=True)
    app.run(debug=True)
```

ДОДАТОК Б

SQL-запити для створення БД

```
CREATE TABLE Customer (
    customer_id INT PRIMARY KEY AUTO_INCREMENT,
    last_name VARCHAR(50) NOT NULL,
    first_name VARCHAR(50) NOT NULL,
    patronymic VARCHAR(50) NOT NULL,
    phone VARCHAR(20) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(255) NOT NULL,
    profile_pic VARCHAR(255) DEFAULT 'default.png'
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE Product (
    product_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(100) NOT NULL,
    category VARCHAR(50) NOT NULL,
    price DECIMAL(10, 2) NOT NULL,
    stock_quantity INT NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE `Order` (
    order_id INT PRIMARY KEY AUTO_INCREMENT,
    order_date DATETIME NOT NULL,
    total_amount DECIMAL(10, 2) NOT NULL,
    delivery_address VARCHAR(255) NOT NULL,
    customer_id INT,
    CONSTRAINT fk_customer FOREIGN KEY (customer_id)
        REFERENCES Customer(customer_id)
        ON DELETE SET NULL
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE OrderDetails (
    order_id INT,
    product_id INT,
    quantity INT NOT NULL,
    unit_price DECIMAL(10, 2) NOT NULL,
    PRIMARY KEY (order_id, product_id),
    CONSTRAINT fk_order FOREIGN KEY (order_id)
        REFERENCES `Order`(order_id)
        ON DELETE CASCADE,
    CONSTRAINT fk_product FOREIGN KEY (product_id)
        REFERENCES Product(product_id)
        ON UPDATE CASCADE
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
CREATE TABLE User (
    user_id INT PRIMARY KEY AUTO_INCREMENT,
    name VARCHAR(50) NOT NULL,
    role VARCHAR(20) NOT NULL,
    password VARCHAR(255) NOT NULL
) ENGINE=InnoDB DEFAULT CHARSET=utf8mb4;
```

ДОДАТОК В

Графічні матеріали та інтерфейси

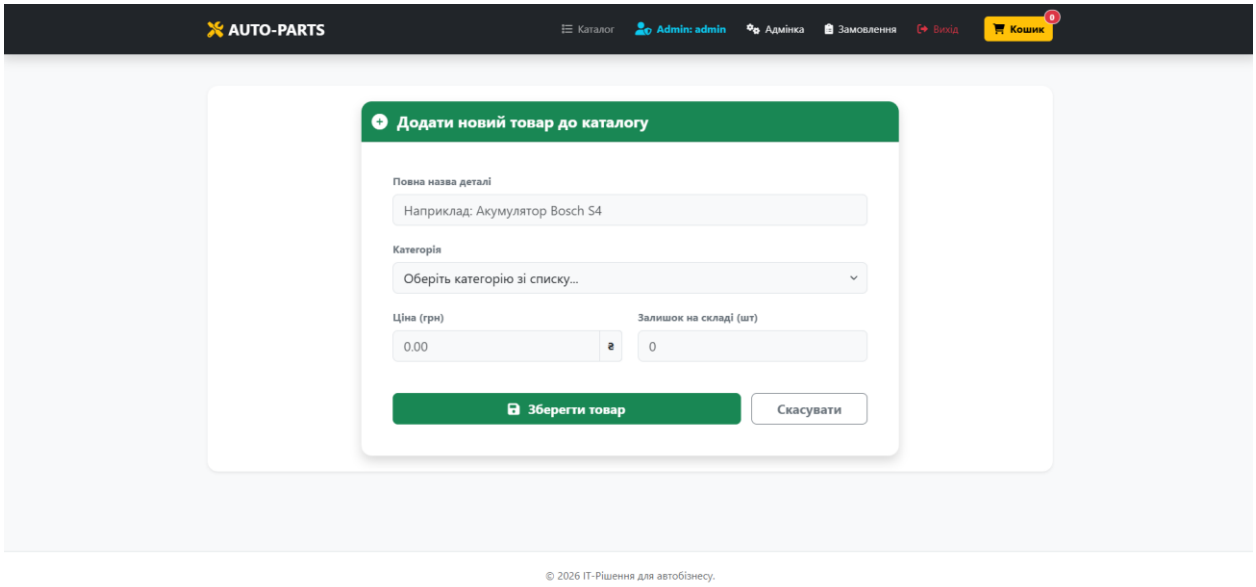


Рисунок В.1 – Меню додавання товару до каталогу

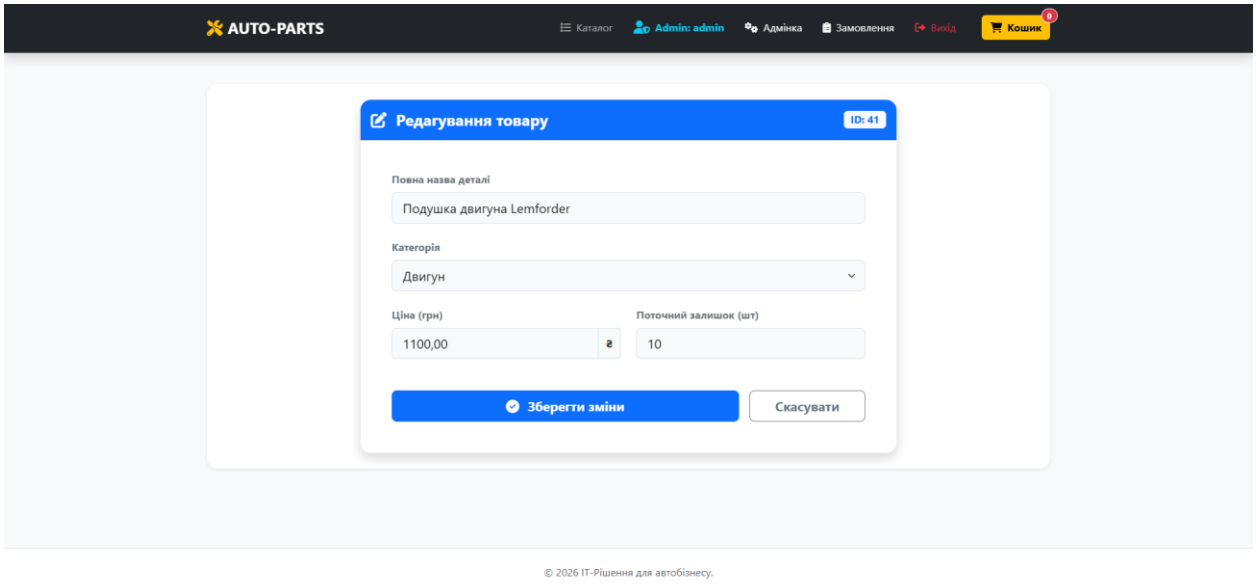


Рисунок В.2 – Меню редагування товару

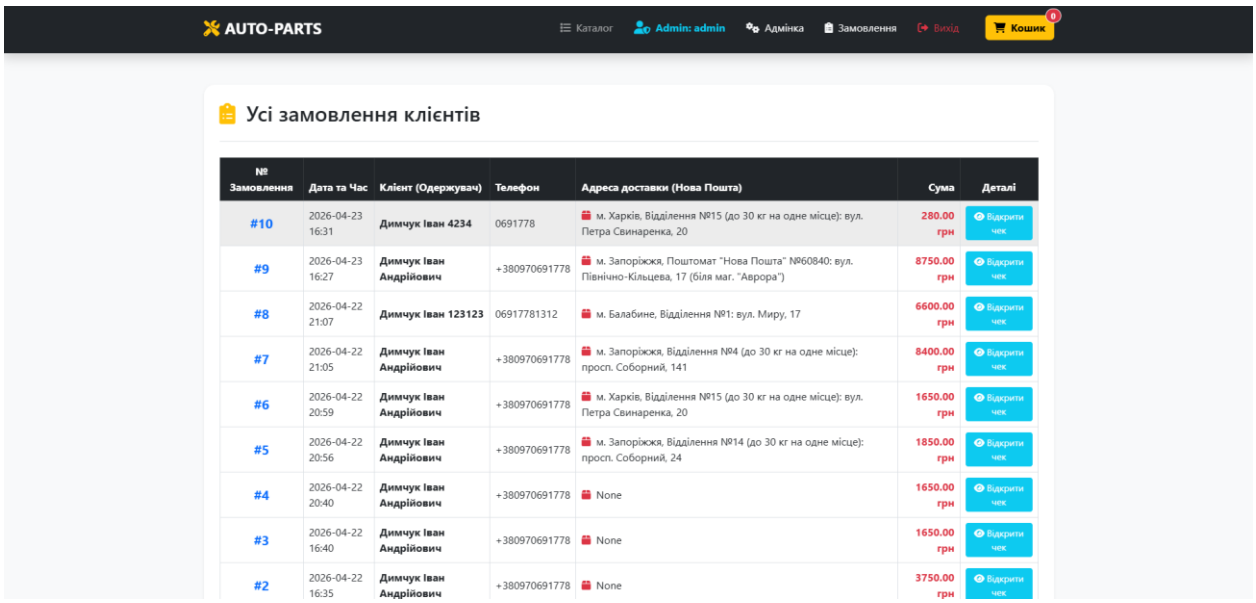


Рисунок В.3 – Список замовлень клієнтів

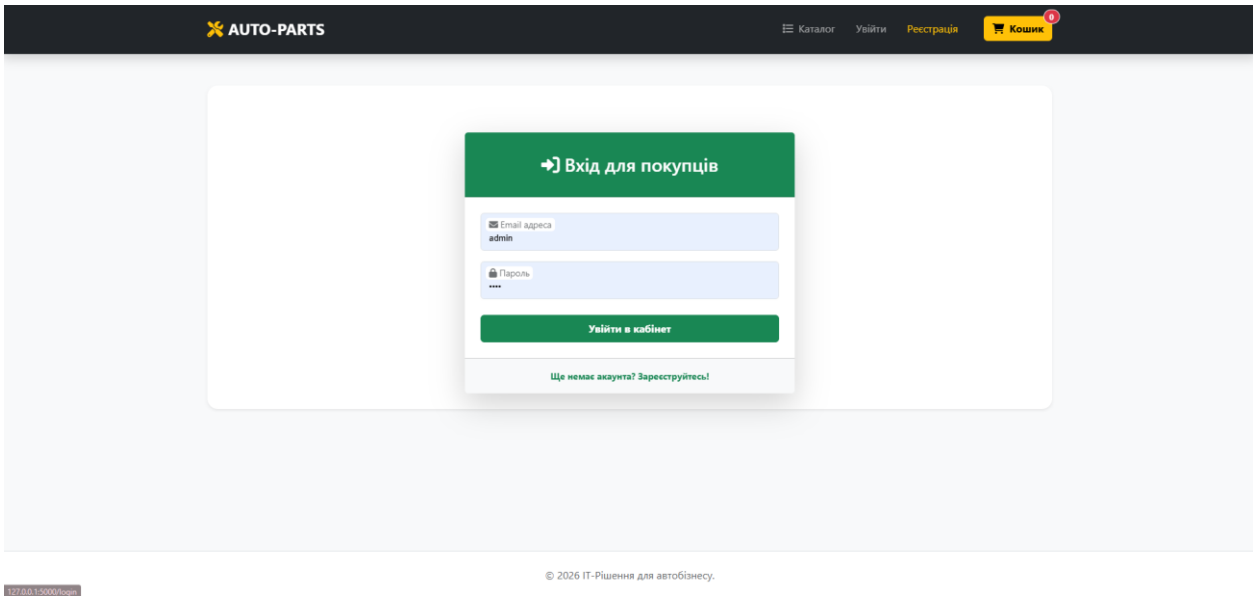


Рисунок В.4 – Вікно входу для покупців

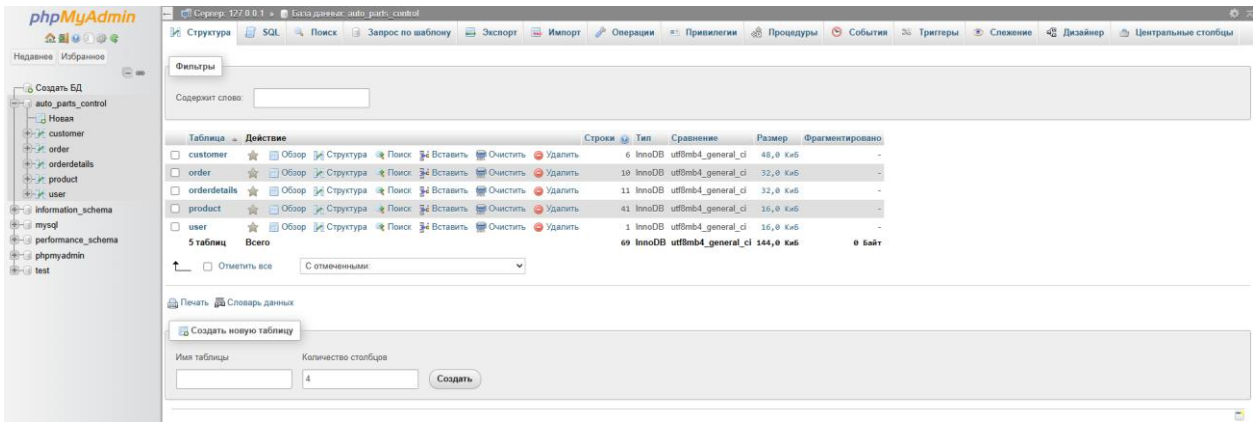


Рисунок В.5 – База даних проєкту

ДОДАТОК Г

Шаблон кошика з динамічним оновленням даних

```
{% extends "base.html" %}
{% block content %}
<h2 class="mb-4"><i class="fas fa-shopping-basket text-warning me-2"></i> Ваш кошик</h2>
{% if products %}
<div class="table-responsive rounded-3">
   <table class="table align-middle shadow-sm overflow-hidden mb-0">
      <thead class="table-light border-bottom">
         <tr>
            <th>Товар</th><th>Ціна</th><th class="text-center">Кількість (шт)</th><th>Сума</th><th
class="text-center">Видалити</th>
         </tr>
      </thead>
      <tbody class="bg-white">
         {% for p in products %}
         <tr class="hover-animate">
            <td><strong>{{ p.name }}</strong></td>
            <td>{{ p.price }} грн</td>
            <td class="text-center">
               <form action="{{ url_for('cart_action') }}" method="POST" class="d-flex justify-content-center">
                  <input type="hidden" name="product_id" value="{{ p.product_id }}">
                  <input type="hidden" name="action" value="update">
                  <input type="number" name="quantity" value="{{ p.qty }}" min="1" max="{{ p.stock_quantity }}"
class="form-control form-control-sm text-center border-success fw-bold" style="width: 80px;"
onchange="this.form.submit()">
               </form>
            </td>
            <td class="text-primary fw-bold">{{ p.subtotal }} грн</td>
            <td class="text-center">
               <form action="{{ url_for('cart_action') }}" method="POST" class="d-inline">
                  <input type="hidden" name="product_id" value="{{ p.product_id }}">
                  <button type="submit" name="action" value="remove" class="btn btn-sm btn-outline-danger btn-
animated"><i class="fas fa-trash"></i></button>
               </form>
            </td>
         </tr>
         {% endfor %}
      </tbody>
   </table>
</div>
<div class="d-flex justify-content-between align-items-center bg-light p-4 rounded-3 mt-4 border shadow-sm">
   <h3 class="mb-0">Разом: <strong class="text-danger">{{ total }} грн</strong></h3>
   <a href="{{ url_for('checkout') }}" class="btn btn-success btn-lg btn-animated px-5 fw-bold"><i class="fas fa-
check-circle me-2"></i> Оформити замовлення</a>
</div>
{% else %}
<div class="text-center py-5 fade-in">
   <i class="fas fa-cart-arrow-down fa-5x text-muted mb-4"></i>
   <h3>Кошик порожній</h3>
   <a href="{{ url_for('index') }}" class="btn btn-primary btn-lg mt-3 btn-animated">Перейти до товарів</a>
</div>
{% endif %}
{% endblock %}
```

Декларація

академічної доброчесності

здобувача освіти ВСП «Економіко-правничий фаховий коледж ЗНУ»

Я, Димчук Іван Андрійович, здобувач освіти 4 курсу, спеціальності 121 Інженерія програмного забезпечення, освітньо-професійної програми Розробка програмного забезпечення, групи К121-22, адреса електронної пошти vanunapochta@gmail.com

- підтверджую, що написана мною дипломна робота на тему «Проєктування та розробка інформаційної системи для продажу автозапчастин» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст. 42 Закону України «Про освіту», зі змістом яких ознайомлений/ознайомлена;

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 01.06.2026 Підпис _____ Іван ДИМЧУК
Ім'я ПРІЗВИЩЕ (здобувача освіти)

Дата 01.06.2026 Підпис _____ Юлія БОРИСОВСЬКА
Ім'я ПРІЗВИЩЕ (наукового керівника)