Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматики
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

РОЗРОБКА WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИН ПОБУТОВИХ ТОВАРІВ»

Виконала: студентка 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Нілова М.Г.
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О.В.
(прізвище та ініціали)
«04» червня 2025 р.
Рецензент: к.т.н., доцент каф. САІТ
Горячев Г.В.
(прізвище та ініціали)
«05» червня 2025 р.

Допущено до захисту
Зав. кафедри Юрій А.А.
«06» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо-професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«05» травня 2025 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Ніловій Марії Геннадіївні

1. Тема роботи: Розробка WEB-додатку «Інтернет магазин побутових товарів» .
керівник роботи: Сілагін Олексій Віталійович, к.т.н., доцент кафедри КН.
затверджені наказом вищого навчального закладу «20» березня 2025 року №97

2. Термін подання студентом роботи 30 травня 2025 р.

3. Вихідні дані до роботи: кількість сутностей – не менше 4; кількість сторінок – не менше 4; реляційна база даних; відповідність стандарту ODBC; дизайн інтерфейсу має бути в білому та синіх тонах, інтуїтивно зрозумілий.

4. Зміст текстової частини (перелік питань, які потрібно розробити):
вступ; обґрунтування доцільності розробки WEB-додатку «Інтернет магазин побутових товарів», аналіз відомих рішень сфері створення WEB-додатку, проектування WEB-додатку «Інтернет магазин побутових товарів», реалізація, тестування роботи програми та аналіз результатів, висновки; список використаних джерел; додатки.

5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень): UML-діаграма прецедентів WEB-додатку «Інтернет магазин побутових товарів»; UML-діаграма послідовності WEB-додатку «Інтернет магазин побутових товарів»; UML-діаграма розгортання WEB-додатку «Інтернет магазин побутових товарів»; ER-модель предметної області WEB-додатку «Інтернет магазин побутових товарів»; схема бази даних WEB-додатку «Інтернет

магазин побутових товарів», схема алгоритму роботи WEB-додатку «Інтерне магазин побутових товарів»; приклад роботи програми.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1–3	Сілагін О.В., к.т.н., доцент каф. КН		

7. Дата видачі завдання 05 травня 2025 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Обґрунтування доцільності розробки WEB-додатку «Інтернет магазин побутових товарів»	05.05.25	07.05.25	
2	Аналіз відомих рішень сфері створення WEB-додатку	08.05.25	11.05.25	
3	Проектування WEB-додатку «Інтернет магазина побутових товарів»	12.05.25	15.05.25	
4	Програмна реалізація WEB-додатку «Інтернет магазина побутових товарів»	16.05.25	26.05.25	
5	Тестування роботи програми та аналіз результатів	27.05.25	29.05.25	
6	Розробка інструкції користувача	30.05.25	01.06.25	
7	Оформлення матеріалів до захисту БДР	02.06.25	04.06.25	

Студентка _____ **Нілова М.Г.**
(підпис) (прізвище та ініціали

Керівник роботи _____ **Сілагін О.В.**
(підпис) (прізвище та ініціал

АНОТАЦІЯ

Нілова М.Г. WEB-додаток «Інтернет магазин побутових товарів» Бакалаврська кваліфікаційна робота зі спеціальності 122 – комп’ютерні науки, освітня програма – комп’ютерні науки. Вінниця: ВНТУ, 2025.

Бакалаврська кваліфікаційна робота складається з 93 сторінок формату А4, на яких є 21 рисунків, 3 таблиці, список використаних джерел містить 21 найменування.

У цій бакалаврській роботі досліджено етапи створення WEB-додатку «Інтернет-магазин побутових товарів». Проведено аналіз існуючих платформ з визначенням їхніх сильних і слабких сторін, сформовано функціональні та нефункціональні вимоги до системи й запропоновано рішення на їх основі. За допомогою UML-діаграм (прецедентів, класів, послідовності та розгортання) спроєктовано архітектуру додатку та розроблено алгоритми основних бізнес-процесів. Реляційну базу даних реалізовано у 3НФ, а серверну частину — на Spring Boot із використанням Spring Security і Spring Data JPA; клієнтську — із Thymeleaf і Bootstrap. Результати модульного й інтеграційного тестування підтвердили відповідність розробленого рішення поставленим вимогам.

Ключові слова: WEB-додаток, інтернет-магазин, побутові товари, Spring Boot, Thymeleaf, UML, реляційна база даних, тестування.

ABSTRACT

Nilova M.H. WEB Application «Online Store of Household Goods»: Bachelor's Qualification Work in Computer Science (Educational Program – Computer Science). Vinnytsia: VNTU, 2025.

The Bachelor's qualification work comprises 93 A4 pages, including 21 figures and 3 tables. The list of references contains 21 items.

This study investigates the stages of developing a web application titled Online Store of Household Goods. An analysis of existing platforms was conducted to identify their strengths and weaknesses, and both functional and non-functional requirements were formulated with corresponding solutions proposed. Using UML diagrams (use-case, class, sequence, and deployment), the application architecture was designed and core business-process algorithms developed. A relational database in 3NF was implemented, the server side built on Spring Boot with Spring Security and Spring Data JPA, and the client side implemented using Thymeleaf and Bootstrap. Results from unit and integration testing confirmed that the developed solution meets the specified requirements.

Keywords: web application, online store, household goods, Spring Boot, Thymeleaf, UML, relational database, testing.

ЗМІСТ

ВСТУП 8

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ РОЗРОБКИ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИН ПОБУТОВИХ ТОВАРІВ» 10

1.1 Аналіз відомих рішень сфері створення WEB-додатку 12

1.2 Вибір технології та інструментарії для створення WEB-додатку 17

1.3 Формулювання вимог та постановка задач на розробку WEB-додатку 18

1.4 Висновок 20

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИНА ПОБУТОВИХ ТОВАРІВ» 21

2.1 Обґрунтування та вибір архітектури 21

2.2 Розробка структури та UML-діаграм 22

2.3 Розробка бази даних для WEB-додатку 30

2.4 Розробка схеми алгоритму роботи для WEB-додатку побутових товарів. 35

2.4 Висновок 41

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИНА ПОБУТОВИХ ТОВАРІВ» 42

3.1 Обґрунтування та вибір мов середовищ та бібліотек 42

3.2 Обґрунтування вибору мови та бібліотек програмування 42

3.3 Тестування роботи програми та аналіз результатів 47

3.4 Висновок 53

ВИСНОВКИ 54

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 56

ДОДАТКИ 59

ДОДАТОК А (ОБОВ’ЯЗКОВИЙ) ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ 60

ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ) ЛІСТИНГ ПРОГРАМИ 61

ДОДАТОК Б (ОБОВ’ЯЗКОВИЙ) ГРАФІЧНА ЧАСТИНА..... 76

ДОДАТОК Г (ДОВІДНИКОВИЙ) ІНСТРУКЦІЯ КОРИСТУВАЧА 88

ДОДАТОК Д (ДОВІДНИКОВИЙ) ДОВІДКА ПРО ВПРОВАДЖЕННЯ 93

ВСТУП

Актуальність досліджень.

У сучасних умовах стрімкого розвитку електронної комерції все більше споживачів віддають перевагу онлайн-покупкам побутових товарів — від засобів гігієни та побутової хімії до дрібної кухонної техніки й аксесуарів для дому. Зручність, економія часу та можливість порівняти десятки пропозицій одним кліком стимулюють подальший розвиток цієї галузі. Водночас масові маркетплейси (Rozetka, Allo.ua, Prom.ua) часто пропонують надто широкий асортимент, перевантажений інтерфейс, фрагментарну підтримку клієнтів та відсутність чіткої спеціалізації саме на побутових товарах.

Багато платформ-лідерів мають слабкі місця, що заважають користувачам швидко і комфортно здійснювати замовлення. Серед поширених недоліків — заплутані меню й фільтри, нерозвинена програма лояльності, обмежена можливість персоналізації профілю та неповна прозорість інформації про товар. Окрім того, власники бізнесу змушені вкладати значні ресурси в оренду складів, ручне адміністрування асортименту та роботу служби підтримки.

Для оптимізації процесу продажу побутових товарів актуальним є створення спеціалізованого WEB-додатка з вузькою фокусованістю. Такий застосунок поєднає інтуїтивно зрозумілий інтерфейс, гнучкі фільтри пошуку, прозоре відображення характеристик, а також ефективні інструменти адміністрування з чітким розподілом ролей (Admin / Client). Інтеграція сучасного технологічного стеку забезпечить масштабованість, безпеку й простоту подальшого розширення функціоналу.

Метою дослідження є покращення та розширення функціональних можливостей WEB-додатку «Інтернет-магазин побутових товарів» із одночасним підвищенням безпеки його роботи та захисту даних користувачів.

Об’єктом дослідження є процес створення WEB-додатка інтернет магазину побутових товарів.

Предметом дослідження є алгоритми та програмно-технічні засоби для реалізації WEB-додатка «Інтернет-магазин побутових товарів».

Завдання дослідження:

1. Обґрунтувати доцільність розробки WEB-додатка для продажу побутових товарів;
2. Проаналізувати аналіз відомих рішень по продажу побутових товарів;
3. Спроектувати WEB-додаток інтернет магазин побутових товарів;
4. Реалізувати серверну частину та клієнтський інтерфейс WEB-додатка інтернет магазин побутових товарів;
5. Виконати тестування програми та провести аналіз отриманих результатів;

Апробація роботи. Результати досліджень було представлено та апробовано на Міжнародній науково-практичній інтернет-конференції «Молодь в науці: дослідження, проблеми, перспективи» (МН ВНТУ - 2025) м. Вінниці у 2025 р.

Публікації. За основними результатами досліджень бакалаврської кваліфікаційної роботи було опубліковано тези доповіді на науково-технічній конференції [1].

1 ОБҐРУНТУВАННЯ ДОЦІЛЬНОСТІ СТВОРЕННЯ РОЗРОБКИ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИН ПОБУТОВИХ ТОВАРІВ»

Сучасний ринок бізнесу та електронної комерції розвивається та змінюється дуже стрімкими темпами: щороку обсяг онлайн-продажів зростає на десятки відсотків, і ця тенденція лише посилюється завдяки безперервному вдосконаленню технологій та росту інтернет-покриття. Багато бізнес-організацій сьогодні використовують WEB-додаток не просто як додатковий канал продажів, а як основний майданчик для здійснення комерційних операцій та взаємодії з клієнтами. Це особливо актуально для сегмента побутових товарів – від засобів для дому та господарства до дрібної кухонної техніки й великої побутової техніки для сільського господарства, адже саме ці категорії товарів найчастіше купуються саме в інтернет-режимі.

Поворотним моментом у розвитку електронної торгівлі стала пандемія COVID-19, яка кардинально змінила підходи споживачів до покупок. Мільйони людей, змушені залишатися вдома, відкрили для себе всі переваги онлайн-шопінгу: зручність, мінімізацію контактів, швидкість оформлення замовлень та економію часу. Навіть після закінчення карантинних обмежень значна частина користувачів не повернулася до виключно офлайн-покупок, оскільки звикла до комфорту та простоти, які дарує WEB-додаток, і цінує можливість порівнювати десятки пропозицій буквально за кілька кліків.

Сам по собі WEB-додаток поєднує в собі найкращі риси прямого маркетингу та традиційного фізичного магазину. На відміну від звичайних торговельних точок, він дозволяє запропонувати клієнту значно ширший асортимент продукції: зручну навігацію за категоріями, докладні описи, технічні характеристики, огляди, фото- та відеоматеріали, а також відгуки інших покупців. Завдяки цьому кожен користувач отримує повний комплекс інформації, необхідний для прийняття обґрунтованого рішення, навіть якщо раніше він полюбляв протестувати товар у живу. Додатково WEB-додаток

підтримує персоналізовані профілі з історією замовлень, що спрощує повторні покупки, обробку гарантійних випадків та рекомендаційні системи.

Для власників бізнесу онлайн-торгівля відкриває можливості суттєвого скорочення витрат: необхідність оренди великих площ і найму значної кількості працівників відпадає, оскільки обробку замовлень і ведення обліку можна автоматизувати. Автоматизація управління складом, обробки замовлень та реалізації маркетингових активностей дозволяє мінімізувати людський фактор, знизити ризики помилок і підвищити швидкість обслуговування клієнтів, що в результаті позитивно впливає на загальну рентабельність та репутацію підприємства.

Водночас одним із найважливіших викликів у розробці та впровадженні WEB-додатків є необхідність поєднати передові інтернет-технології з традиційними торговельними підходами. У фізичному магазині покупець може безпосередньо оцінити якість, вагу та матеріали товару, тоді як в онлайні він позбавлений такої можливості. Хоча сучасні WEB-додатки надають візуальну інформацію, фотографії високої роздільної здатності й відеоогляди, все ж іноді трапляються випадки, коли недобросовісні продавці вказують неточні характеристики або приховують дефекти, особливо якщо ціни суттєво нижчі за середньоринкові.

Разом із тим кількість WEB-додатків та маркетплейсів постійно зростає: це вигідно й зручно як для продавців, так і для покупців. На відміну від звичайного магазину, WEB-додаток має можливість працювати цілодобово, без вихідних чи перерв, а багато процесів — від прийому замовлення до формування вантажу — можуть виконуватися без прямої участі співробітників завдяки інтегрованим системам обробки даних. Ще однією перевагою є мінімальні інвестиції в товарні запаси: продавцю достатньо налагодити постачання «just-in-time», замовляючи продукцію під конкретний запит клієнта, що суттєво знижує ризики «завалів» на складі.

Географія продажів WEB-додатку розширюється далеко за межі одного регіону чи міста. Доставка здійснюється будь-якими логістичними каналами:

кур’єрськими службами, поштовими операторами, авіа- та морським транспортом, що відкриває нові ринки та дозволяє підприємствам ефективно масштабувати бізнес.

Метою даної кваліфікаційної роботи є розробка WEB-додатку «Інтернет-магазин побутових товарів», який виступить головною платформою для здійснення продажу побутових товарів у рамках певного підприємства, забезпечуючи стабільну, безпечну і комфортну взаємодію між клієнтом і продавцем.

Хоча на ринку вже існує чимало аналогічних рішень, вони мають типові недоліки: загальний асортимент занадто розширений і не має чіткого фокусу на побутових товарах, користувачу складно швидко знайти потрібну категорію; канали комунікації з клієнтами часто мають обмежену гнучкість — довгі черги на «гарячих лініях» або недостатній рівень підтримки у складних або гарантійних випадках.

Саме тому завданням цієї роботи є створення зручного WEB-додатку «Інтернет-магазин побутових товарів», який забезпечить безперешкодний пошук і швидкий вибір товарів для клієнтів, а співробітникам підприємства — ефективну систему обробки замовлень і передачі даних до служби доставки для оперативного й надійного виконання замовлень.

1.1 Аналіз відомих рішень сфері створення WEB-додатку

Для аналізу було обрано декілька сайтів зі схожою тематикою і призначенням.

Одним із таких сайтів є WEB-додаток інтернет магазин Rozetka [2], хоча останнім часом він став маркетплейсом, для порівняння можна взяти його категорії де є побутові товари. WEB-додаток інтернет магазин Rozetka з вибраною категорією «Побутова техніка» представлений на рис.1.1.

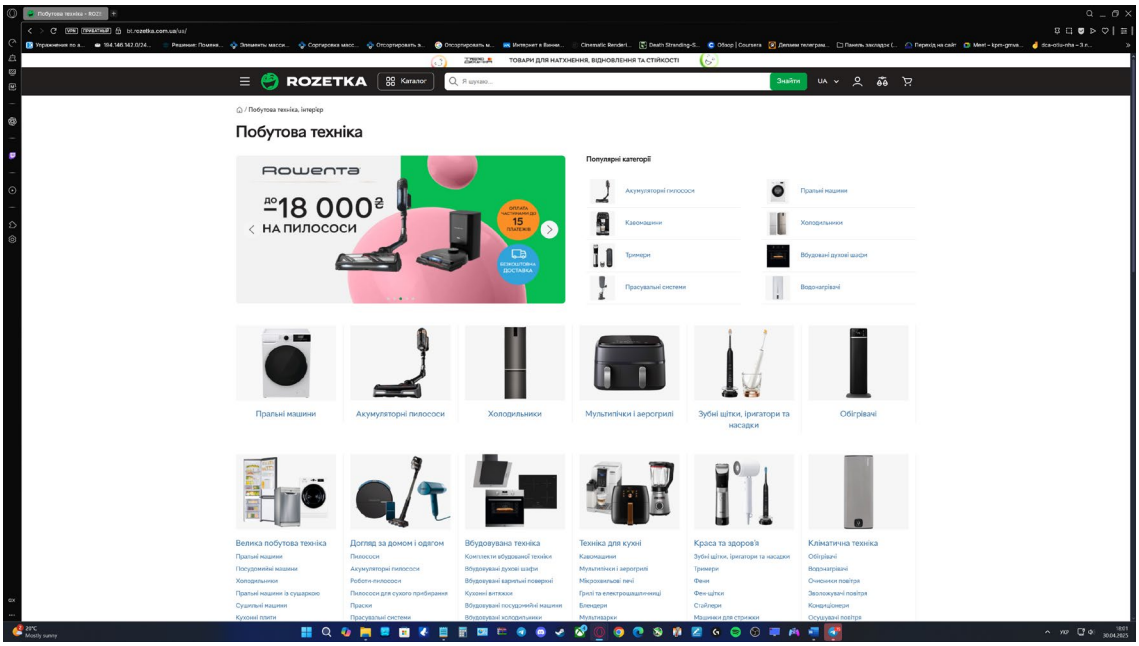


Рисунок 1.1 – Сторінка «Побутова техніка» сайту Rozetka

Розетка залишається лідером українського ринку електронної комерції, насамперед завдяки комплексному підходу до сервісу та розвитку додаткових можливостей для покупців. Її інтерфейс, хоч і вражає багатофункціональним, іноді сприймається як перевантажений, особливо новачками: величезна кількість категорій, фільтрів і акцій можуть відволікати від швидкого пошуку потрібного товару.

Водночас сильна сторона платформи — власна логістика. Завдяки кур'єрській службі Rozetka доставка до великих міст займає лише 1–2 дні, а в регіонах — до 3–4 днів. Це забезпечує конкурентну перевагу над маркетплейсами, які покладаються тільки на сторонні служби доставки.

Ще один мотиватор для покупців — активні маркетингові кампанії: яскраві розпродажі на «чорну п'ятницю», регулярні акції та кешбек за допомогою фірмової картки, що стимулює неодноразово повертатися на сайт та збільшувати середній чек.

Схожу тематику також має сайт Allo.ua, представлений на рис. 1.2.

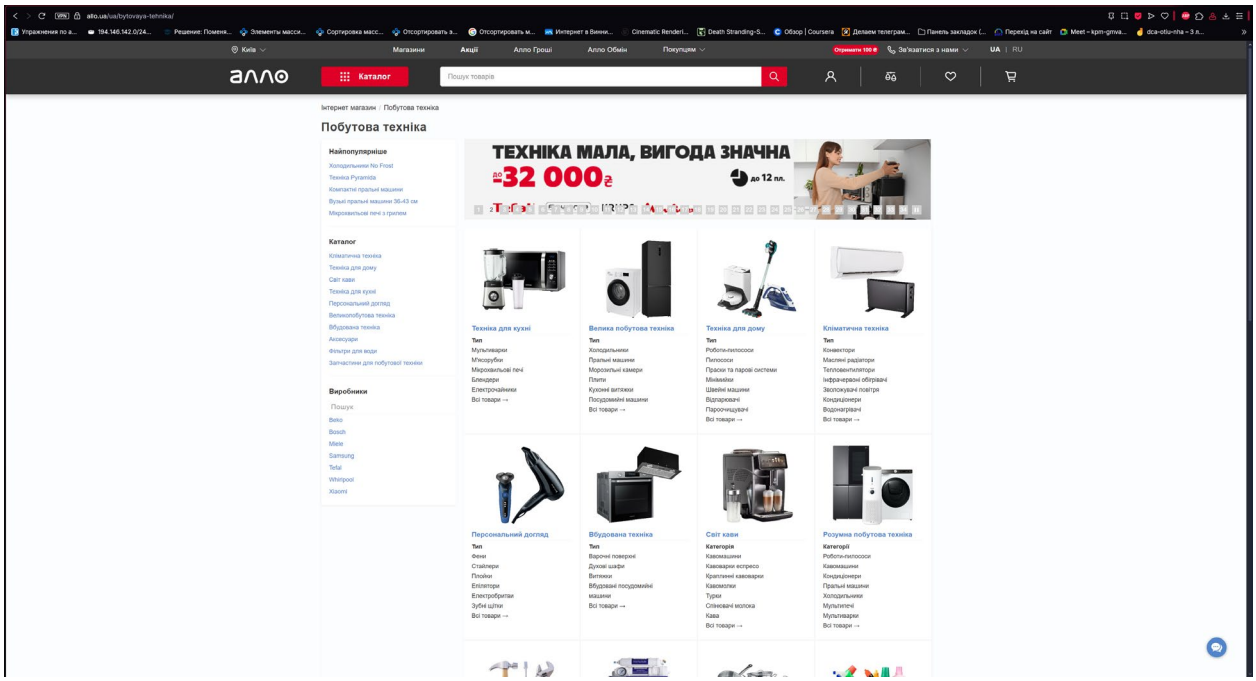


Рисунок 1.2 – Сторінка «Побутова техніка» сайту Allo.ua

Allo.ua [3] сфокусована здебільшого на електроніці та побутовій техніці, що дозволяє їй глибше працювати саме з цими категоріями товарів. Інтеграція з банківськими сервісами дає змогу клієнтам оформити покупку в розстрочку буквально в кілька «кліків», а партнерство з такими перевізниками, як Нова Пошта та Укрпошта, гарантує швидку доставку і можливість самовивозу.

Однак вузька спеціалізація означає, що асортимент побутових дрібниць і хімії у них представлений не в повному обсязі, а програма лояльності менш гнучка, ніж у великих маркетплейсів: бонуси переважно прив’язані до банківських карт, а варіантів знижок на комбіновані набори тут небагато. Це створює простір для нового вузькоспеціалізованого рішення — з інтуїтивним UX, жорсткими стандартами якості та розширеними можливостями кастомізації замовлень.

Для аналізу також був обраний сайт маркетплейсу Prom.ua [4] основна тематика якого продаж не тільки побутової техніки, а також і багато інших товарів.

Сайт маркетплейсу Prom.ua з категорією «Побутова техніка» представлений на рис. 1.3.

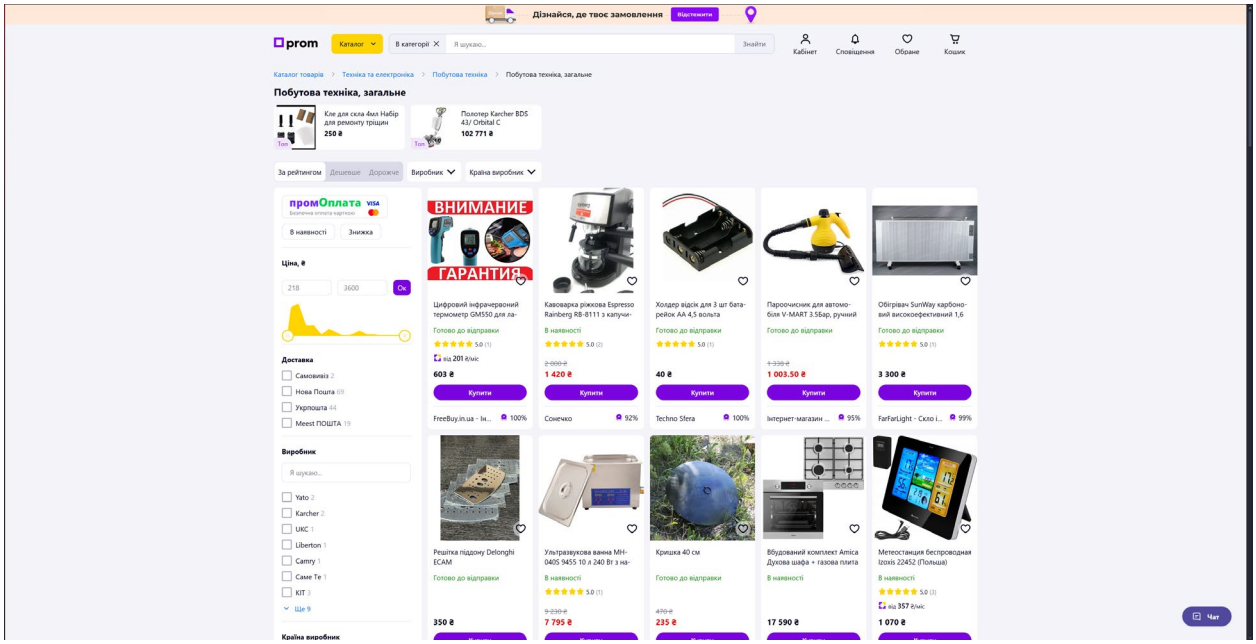


Рисунок 1.3 – Сторінка «Побутова техніка» сайту Prom.ua

Prom.ua займає іншу нішу: це відкрита платформа-маркетплейс, де будь-хто може стати продавцем, а покупець отримує доступ до тисяч оголошень з різних країн — від Китаю до Туреччини.

Для багатьох продавців саме невисока комісія та інтуїтивно зрозумілі інструменти створення картки товару стають вирішальними факторами при виборі Prom.ua як майданчика для продажів. Проте така відкритість призводить до нестабільності якості: нерідко покупець стикається з некоректними описами, затримками або невідповідністю товару заявленому.

Слабка модерація та відсутність жорстких вимог до постачальників знижують довіру користувачів і змушують шукати альтернативи або додатково перевіряти кожну пропозицію.

На сьогоднішній день створена велика кількість веб-додатків подібного плану, але вони мають певні недоліки.

Проаналізувавши більшість вже створених WEB-додатків інтернет магазин можна відокремити декілька недоліків, а саме:

1. Дизайн і UX: інтерфейси багатофункціональні, але через це іноді здається перевантаженим.

2. Модерація: на маркетплейсах часто погано модерується якість оголошень, клієнт не завжди отримує товар згідно опису.
3. Якість: за частіше на маркетплейсах нестабільна якість товару, та часте не співпадіння з заявленими характеристиками на сайті.
4. Замовлення та логістика: неквапліва обробка замовлення та доставка.

Розглянемо детальніше інформацію про WEB-додатки інтернет магазинів побутових товарів та проведемо загальний аналіз (табл. 1.1).

Таблиця 1.1 – Аналіз розглянутих сайтів для продажу побутових товарів

Критерії	Rozetka	Allo.ua	Prom.ua
Відсутність непотрібної реклами	-	+	-
Неактуальна інформація	+	-	+
Велика база продавців	+	-	+
Широкий асортимент, логістика	+	-	-
Розстрочка, інтеграція з банками	+	+	-
Застарілий дизайн	+	-	+

Отже, проаналізувавши сайти зі схожою тематикою і призначенням було вирішено розробити сайт, який не матиме недоліків виявлених на WEB-додатках представлених в таблиці 1.1. Дана розробка матиме всі переваги даних WEB-додатків, а також зручний менеджмент додавання нових товарів для працівників магазину, і зручний інтерфейс для клієнтів.

1.2 Вибір технології та інструментарії для створення WEB-додатку

При виборі технологій для розробки WEB-додаток інтернет магазин побутових товарів головними критеріями були надійність, простота підтримки, масштабованість та швидкість виходу продукту на ринок. Серверну частину було реалізовано на основі Java із застосуванням Spring Boot, оскільки цей фреймворк поєднує переваги доконфігурованої платформи та гнучкості Spring Framework. Автоматичне підключення Web MVC, підтримка REST, вбудовані засоби безпеки Spring Security і зручна робота з базою даних через Spring Data JPA суттєво скоротили час налаштування проєкту й написання «шаблонного» коду.

Для зберігання даних на етапі розробки обрана вбудована база H2, яка не потребує додаткового розгортання й дозволяє легко виконувати юніт-тести. У продакшн-середовище передбачена можливість заміни на PostgreSQL без змін у бізнес-логіці завдяки ORM-шару Hibernate.

Клієнтська частина реалізована з використанням Thymeleaf — серверного шаблонізатора, що щільно інтегрується зі Spring Boot, а також Bootstrap 5 для побудови адаптивного інтерфейсу. Такий підхід дозволив швидко створити сучасний UX/UI, який автоматично підлаштовується під мобільні пристрої без необхідності складного JavaScript-фреймворку.

У процесі розробки активно використовувалися інструменти автоматизації та підвищення продуктивності: Maven для керування залежностями, Lombok — для зменшення шаблонності в коді, ModelMapper — для зручного мапінгу між сутностями та DTO, JUnit [16] і Mockito [17] — для модульного тестування

сервісів і контролерів. Для контролю версій застосовано Git із розміщенням репозиторію на GitHub, а для тестування API — Postman.

Усі ці рішення у комплексі забезпечують швидку розробку, високу якість коду та можливість подальшого масштабування WEB-додатку під зростаючі потреби підприємства.

1.3 Формулювання вимог та постановка задач на розробку WEB-додатку

Розробка WEB-додатка побутових товарів потребує чіткого визначення функціональних та нефункціональних вимог, що дозволить спроектувати зручний, безпечний і масштабований сервіс. У цьому підрозділі описано ключові сценарії використання та завдання, яких необхідно досягти в процесі реалізації проекту.

WEB-додаток має забезпечувати ефективну взаємодію двох ролей — Client (покупець) і Admin (адміністратор) — та містити такі основні функції:

Реєстрація та авторизація: користувачі створюють облікові записи з підтвердженням електронної пошти, авторизуються для доступу до персонального кабінету та оформлення замовлень;

− Персональний кабінет: кожен покупець має історію замовлень, список бажань і можливість оновлювати свої контактні й платіжні дані;

− Каталог товарів: інтерактивний перелік із фотографіями, назвою, описом, ціною та наявністю кожної позиції;

− Пошук і фільтрація: швидкий пошук за ключовими словами (назва, бренд), фільтрація за категоріями, ціною, рейтингом і наявністю на складі;

− Кошик: управління вибраними товарами — додавання, видалення, зміна кількості, розрахунок загальної вартості з урахуванням знижок;

− Оформлення замовлення: вибір способу доставки й оплати (онлайн-платіж, накладений платіж), введення адреси й підтвердження покупки;

− Адміністративна панель: для Admin — повний доступ до CRUD-операцій над товарами, категоріями, виробниками, замовленнями та користувачами;

− Безпека та валідація: хешування паролів (BCrypt), захист від CSRF, валідація форм на фронтенді й бекенді;

− Адаптивний дизайн: інтерфейс, що коректно працює на ПК, планшетах і смартфонах;

− Локалізація: підтримка кількох мов інтерфейсу (українська, англійська).

Нефункціональні вимоги включають: час відгуку сервера до 2 секунд, забезпечення 99,8 % uptime, можливість обробляти щонайменше 500 одночасних користувачів, масштабованість архітектури, підтримку CI/CD та логування бізнес-подій.

На підставі зазначених вимог поставлено такі завдання:

1. Провести аналіз ринку онлайн-продажів побутових товарів і виявити основних конкурентів;

2. Сформулювати функціональні та нефункціональні вимоги до системи;

3. Обґрунтувати вибір архітектурного підходу, технологічного стеку та інструментів розробки;

4. Спроектувати структуру бази даних (ER-модель), UML-діаграми (Use Case, Class) та алгоритми основних бізнес-процесів;

5. Реалізувати серверну частину на Spring Boot із модулем безпеки Spring Security та ORM Spring Data JPA;

6. Розробити клієнтський інтерфейс із Thymeleaf та Bootstrap, забезпечивши адаптивність і локалізацію;

7. Виконати модульне й інтеграційне тестування компонентів;

8. Підготувати користувацьку та адміністративну документацію.

1.4 Висновок

У першому розділі було обґрунтовано доцільність створення спеціалізованого WEB-додатку «Інтернет-магазин побутових товарів», що закладає основу для подальшої розробки. Проведено аналіз існуючих платформ (Rozetka, Allo.ua, Prom.ua), виявлено їхні обмеження та потреби користувачів, що дозволило чітко сформулювати функціональні та нефункціональні вимоги до системи. Обґрунтовано вибір багаторівневої архітектури з використанням Spring Boot для бекенду та Thymeleaf для фронтенду, а також визначено ключові технології та інструменти. Сформульовані завдання охоплюють увесь цикл розробки — від аналізу до модульного й інтеграційного тестування та документування. Отже, розглянуті підходи та вимоги забезпечують надійну та гнучку основу для реалізації WEB-додатку і досягнення поставленої мети.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИНА ПОБУТОВИХ ТОВАРІВ»

2.1 Обґрунтування та вибір архітектури

Вибір архітектурного підходу є одним із найважливіших етапів при створенні веб-застосунку, оскільки саме він визначає загальну структуру, масштабованість, гнучкість і підтримуваність проєкту. З огляду на цілі даної кваліфікаційної роботи — розробку WEB-додатку інтернет магазину побутових товарів — було прийнято рішення використати багаторівневу (багатошарову) архітектуру, яка традиційно застосовується у веб-додатках на основі фреймворку Spring Boot [5].

Переваги багаторівневої архітектури:

Багаторівнева архітектура дозволяє чітко розділити логіку програми на окремі шари відповідальності, серед яких можна виділити:

- Презентаційний шар (Presentation layer**)** — відповідає за взаємодію з користувачем. У проєкті використовується шаблонізатор Thymeleaf, який інтегрується зі Spring MVC [6] і дозволяє генерувати динамічні HTML-сторінки з підтримкою інтернаціоналізації.

- Шар бізнес-логіки (Service layer) — містить основну логіку обробки даних, валідацію, розрахунки, управління замовленнями тощо. Саме тут реалізується більшість функціональних вимог, включаючи обробку замовлень, фільтрацію товарів, перевірку прав доступу.

- Шар доступу до даних (Repository/Data Access layer) — взаємодіє з базою даних за допомогою Spring Data JPA, що дозволяє скоротити кількість SQL-запитів і зосередитись на логіці.

- Модельний шар (Domain layer) — включає сутності (Entity) і об’єкти передачі даних (DTO), які представляють структуру предметної області: товари, користувачі, ролі, замовлення тощо.

Поділ проєкту на шари забезпечує високу модульність, полегшує тестування, розширення функціональності та обслуговування проєкту у майбутньому.

Обраний підхід: архітектура MVC

На основі багаторівневої моделі у застосунку реалізовано шаблон MVC (Model–View–Controller), який є стандартним у Spring-застосунках. Цей шаблон передбачає:

- Model — описує дані та правила роботи з ними (наприклад, Product, Order, User).
- View — шаблони, що відображають дані кінцевому користувачеві (HTML-сторінки з Thymeleaf).
- Controller — посередник, який отримує запити, обробляє їх та повертає відповідні представлення.

Застосування MVC дозволяє чітко відокремити логіку інтерфейсу користувача від логіки обробки запитів, що робить систему гнучкою до змін і адаптацій.

2.2 Розробка структури та UML-діаграм

У процесі розробки інформаційної системи важливою складовою є створення структурної моделі предметної області. Для візуалізації логіки та структури системи використовуються UML-діаграми [7], які дозволяють формалізовано описати функціональність, взаємодію користувачів із системою, а також внутрішні залежності між класами. У межах даної роботи було побудовано діаграму варіантів використання «Use Case Diagram» та діаграму класів «Class Diagram», що відображають архітектуру та функціональні можливості WEB-додатку інтернет магазину побутових товарів.

На рисунку 2.1 представлено UML-діаграма класів побутового магазину.

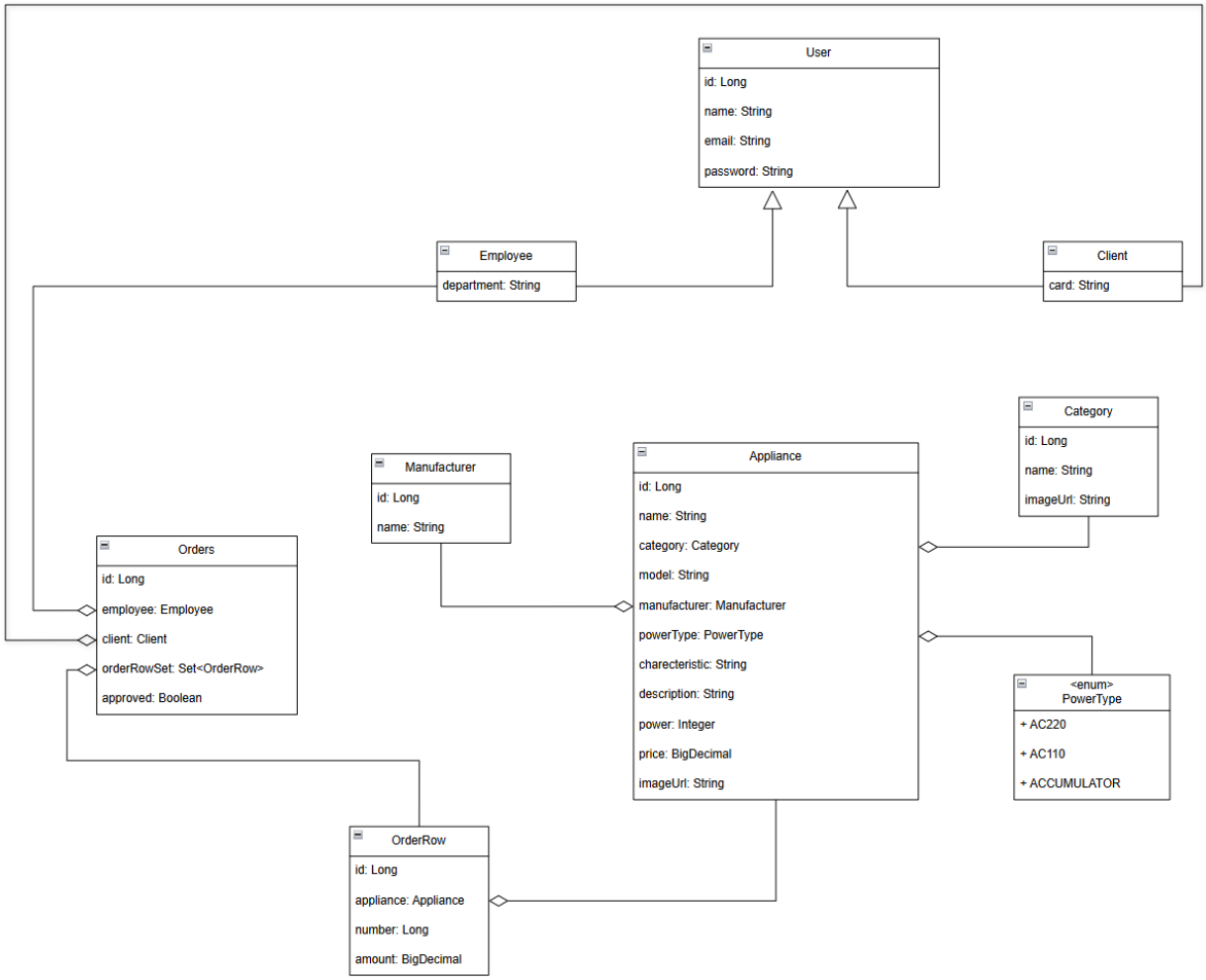


Рисунок 2.1 – UML-діаграма класів побутового магазину

Діаграма класів моделює статичну структуру системи, відображаючи основні сутності, їхні атрибути, зв’язки й асоціації між об’єктами.

User — базовий клас, від якого наслідуються Client та Employee. Містить загальні поля: id, name, email, password.

Client має додаткове поле card для оплати, а Employee — поле department.

Appliance — головна сутність, яка описує побутовий товар. Містить інформацію про назву, категорію, виробника, модель, опис, ціну, зображення та тип живлення (PowerType — як enum).

Category та Manufacturer — окремі сутності, зв’язані з Appliance через асоціацію.

Orders — замовлення, що містить посилання на клієнта, співробітника, список товарних позицій (orderRowSet) та статус підтвердження (approved).

OrderRow — представляє окрему позицію в замовленні: товар, кількість і суму.

Усі зв’язки між класами оформлені відповідно до логіки предметної області: асоціації багато до одного, наслідування, а також композиція для включення рядків замовлення в об’єкт замовлення (Order → OrderRow).

Діаграма варіантів використання моделює функціональну поведінку системи з точки зору користувачів. У нашому випадку розглядаються два основні актори: Client (клієнт) та Employee (співробітник магазину). Кожен з них взаємодіє з системою відповідно до своїх повноважень.

На рисунку 2.2 представлено UML-діаграма прецедентів побутового магазину.

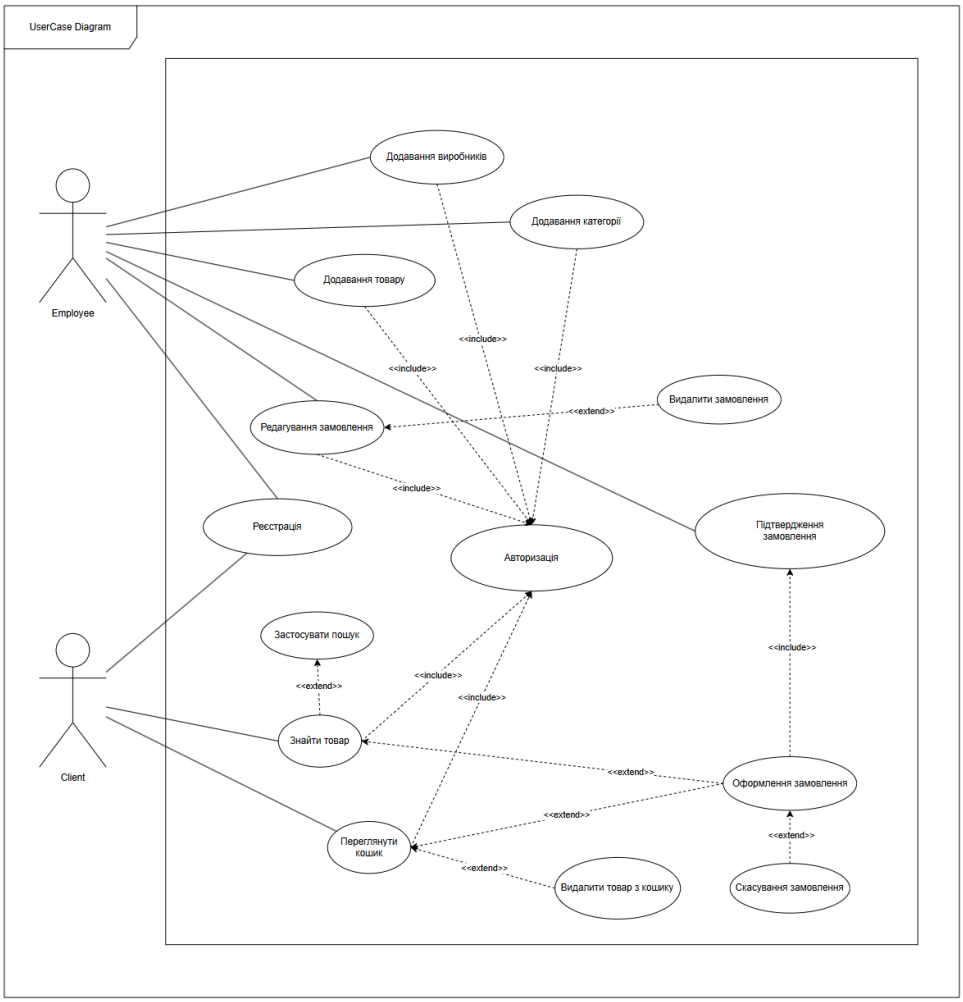


Рисунок 2.2 – UML-діаграма прецедентів побутового магазину

Клієнт може зареєструватися, авторизуватись, шукати товари, переглядати кошик, оформляти замовлення, видаляти товари з кошика, скасовувати замовлення.

Співробітник має доступ до додавання нових товарів, категорій і виробників, редагування та підтвердження замовлень.

Функція Авторизація є обов'язковою (include) для більшості дій користувача, які вимагають автентифікації.

Зв'язки (extend) вказують на дії, які виконуються додатково за певних умов (наприклад, видалення товару з кошика під час перегляду або зміна замовлення під час оформлення).

Ця діаграма допомагає чітко відобразити функціональні вимоги до системи з точки зору кінцевих користувачів, а також визначити обсяг відповідальності кожної ролі.

Діаграма послідовності (Sequence Diagram) є потужним інструментом UML-моделювання, що дозволяє наочно показати, як об'єкти системи обмінюються повідомленнями в часі для реалізації конкретного сценарію. На такій діаграмі можна простежити логічний потік управління між клієнтом, компонентами сервісів і базою даних, що значно полегшує аналіз взаємодії та відлагодження архітектурних рішень.

Основні елементи діаграми послідовності:

– Об'єкти (Objects): екземпляри класів або актори, задіяні у сценарії (Client, WEB-додаток, CartService, OrderService, Database). Вони розташовані у верхній частині діаграми як прямокутники з іменами.

– Лінії життя (Lifelines): вертикальні пунктирні лінії під кожним об'єктом, що відображають його існування протягом виконання сценарію.

– Повідомлення (Messages): горизонтальні стрілки, які показують виклики методів або передавання даних між об'єктами. Синхронні повідомлення показуються суцільними стрілками, асинхронні — з відкритим наконечником.

– Фокус керування (Activation Box): тонкі вертикальні прямокутники на лінії життя, що позначають час, коли об'єкт активно обробляє запит.

Представлена UML-діаграма послідовності на рисунку 2.3 показує взаємодію між користувачем та WEB-додатком інтернет магазину побутових товарів.

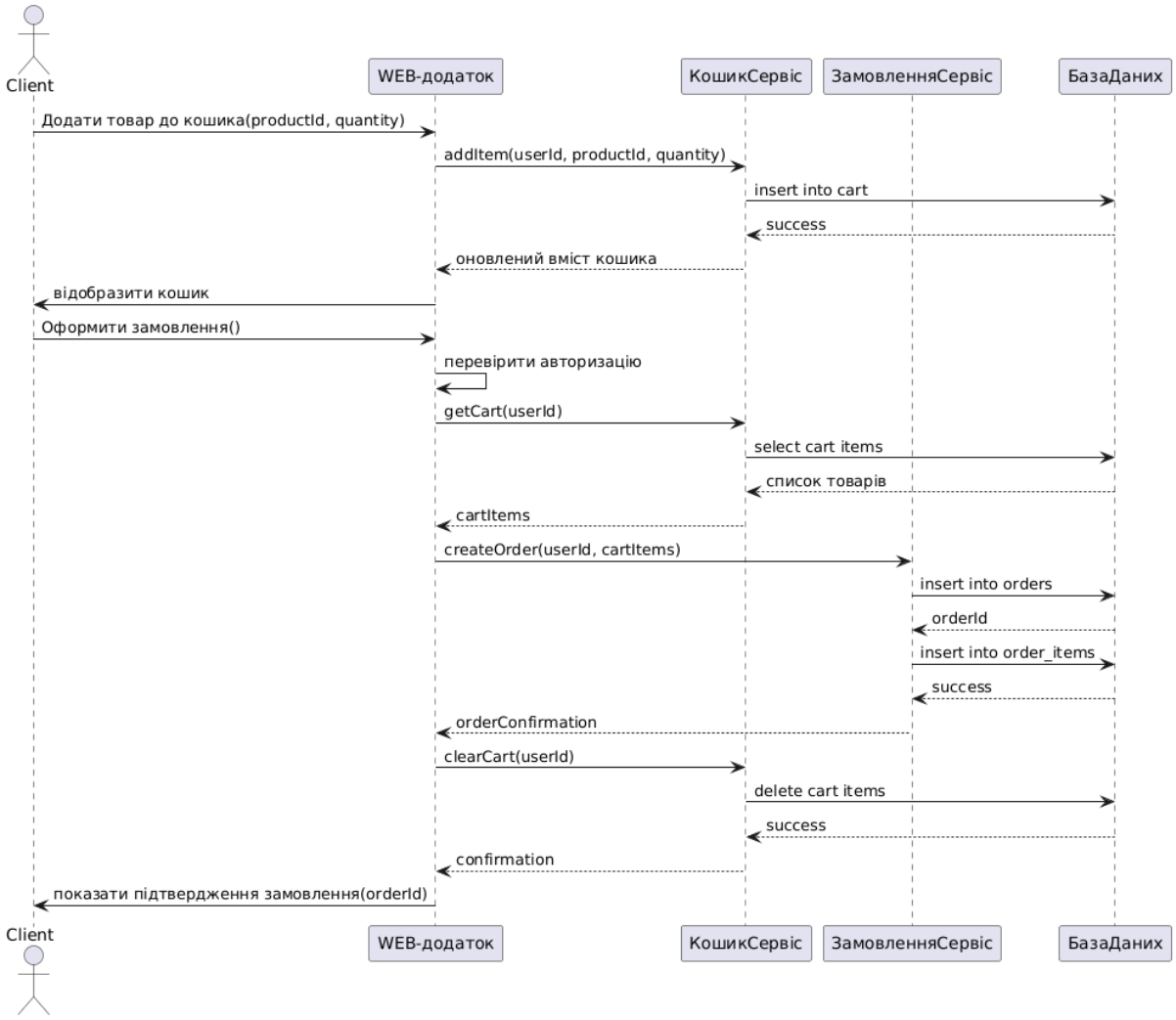


Рисунок 2.3 – UML-діаграма послідовності побутового магазину

Нижче приведено детальний опис двох ключових сценаріїв для WEB-додатку «Інтернет-магазин побутових товарів».

Сценарій 1. Додавання товару до кошика:

1. Client натискає кнопку «Додати до кошика» на сторінці товару.

2. WEB-додаток приймає запит і формує повідомлення addItem(userId, productId, quantity) до CartService.

3. CartService створює фокус керування, звертається до Database з SQL-запитом для додавання позиції в таблицю кошика.

4. Database повертає результат операції (успіх або помилка).

5. CartService передає оновлений вміст кошика назад у WEB-додаток.

6. WEB-додаток оновлює вигляд кошика на інтерфейсі користувача та відправляє відповідь Client.

Сценарій 2. Оформлення замовлення:

1. Client обирає «Оформити замовлення» у кошику.

2. WEB-додаток перевіряє, чи користувач авторизований; якщо ні — генерує запит на авторизацію.

3. Після успішної авторизації WEB-додаток надсилає getCart(userId) до CartService для отримання поточних товарів.

4. CartService запитує Database список позицій кошика; Database повертає їх CartService, який передає їх до WEB-додатка.

5. WEB-додаток формує об’єкт замовлення й надсилає createOrder(userId, cartItems) до OrderService.

6. OrderService активує свій фокус керування, записує дані до таблиці orders, отримує згенерований orderId, а потім додає рядки замовлення в order_items через Database.

7. Після успішного збереження OrderService повертає підготовлене підтвердження (orderConfirmation) до WEB-додатка.

8. WEB-додаток викликає clearCart(userId) у CartService, який видаляє всі записи кошика через Database.

9. WEB-додаток надсилає Client фінальне повідомлення із номером замовлення і статусом успіху.

Обидва сценарії демонструють, як за допомогою UML-діаграми послідовності можна відобразити чіткий порядок викликів між клієнтом,

сервісами та базою даних, а також визначити зони високої навантаженості та точки розширення для майбутньої оптимізації.

Діаграма розгортання (Deployment Diagram) — це різновид UML-діаграм, створений для відображення фізичної інфраструктури, в якій працює програмний продукт. Вона демонструє, які апаратні або віртуальні вузли використовуються, куди розгортаються артефакти додатку та як між собою взаємодіють ці компоненти в реальному середовищі. Завдяки такій діаграмі можна оцінити необхідні обчислювальні ресурси, обсяг пам’яті, пропускну здатність мережі та налаштування середовища виконання для WEB-додатку «Інтернет-магазин побутових товарів».

Основні елементи розгортання:

1. Вузли (Nodes) — фізичні сервери або віртуальні машини, на яких розміщуються компоненти системи. У нашій діаграмі виділено три основні вузли:

- Client Browser — середовище користувача, яке виконує HTTP(S)-запити до додатку;
- Web Server — вузол із розгорнутим компонентом Spring Boot Application, що обробляє бізнес-логіку та HTTP-запити;
- Database Server — окремий вузол із СУБД (PostgreSQL у продакшн або H2 під час розробки), що зберігає всі дані про товари, користувачів, кошик і замовлення.

2. Артефакти (Artifacts) — виконувані файли й бібліотеки, які розгортаються на вузлах. Для WEB-додатку це, зокрема, JAR-файл Spring Boot із вбудованим Tomcat, а також SQL-скрипти ініціалізації бази даних.

3. Зв’язки (Communication Paths) — мережеві з’єднання між вузлами. На діаграмі вони позначені стрілками з позначенням протоколу (HTTPS між Client Browser та Web Server, JDBC між Web Server і Database Server), що допомагає зрозуміти, як відбувається передача даних і які порти/протоколи налаштовуються.

Особливості такої діаграми розгортання:

– Окреме поділ середовища: дозволяє чітко розмежувати фронтенд-клієнта, бекенд-логіку та шар зберігання даних.

– Незалежність від внутрішньої реалізації: діаграма не деталізує класи чи методи, а лише показує фізичне розташування компонентів.

– Масштабованість і висока доступність: за потреби Web Server можна масштабувати горизонтально (додати кілька вузлів), а база даних — кластеризувати, що легко внести до діаграми.

Нижче наведена UML-діаграма розгортання, яка ілюструє розміщення компонентів WEB-додатку «Інтернет-магазин побутових товарів» в типовому продакшн-середовищі рисунку. 2.4.

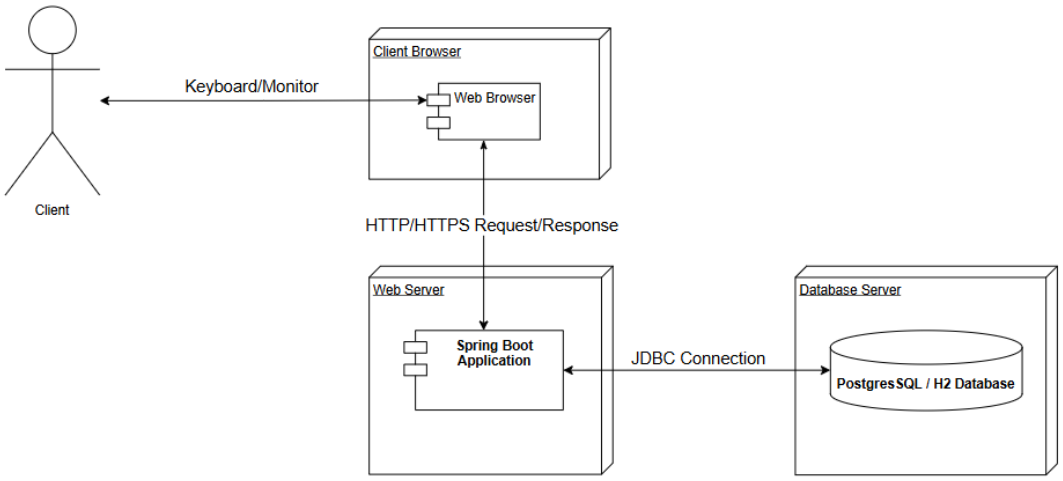


Рисунок 2.4 – UML-діаграма розгортання для WEB-додатку «Інтернет-магазин побутових товарів»

Представлена діаграма розгортання ілюструє фізичну архітектуру WEB-додатку «Інтернет-магазин побутових товарів», що складається з трьох ключових вузлів: Client Browser, Web Server і Database Server.

Користувач взаємодіє з системою через Client Browser, відправляючи HTTP(S)-запити та отримуючи відповідь у вигляді веб-сторінок і динамічного

контенту. У браузері може бути відкрито декілька вкладок, в яких користувач здійснює перегляд каталогу, фільтрацію товарів, керує кошиком і оформлює замовлення.

Запити з Client Browser надходять до Web Server, на якому розгорнуто компонент Spring Boot Application. Цей вузол обробляє бізнес-логіку: маршрутизацію запитів, валідацію даних, виклики сервісів кошика та замовлень, а також генерацію динамічних HTML-сторінок через Thymeleaf. Комунікація між браузером і сервером відбувається по захищеному протоколу HTTPS, що гарантує конфіденційність та цілісність передаваних даних.

Spring Boot Application встановлює з'єднання з Database Server через JDBC (протокол TCP/IP). На сервері бази даних розгорнуто СУБД — у середовищі розробки це H2, а в продакшні може використовуватися PostgreSQL. Саме тут зберігаються сутності каталогу товарів, записи кошика, інформація про користувачів і історію замовлень.

Такий розподіл ролей гарантує:

− Ізоляцію клієнтської частини від даних і логіки — браузер виконує лише представлення і взаємодію з користувачем;

− Централізацію бізнес-логіки на Web Server із можливістю масштабування горизонтально (додавання нових інстансів Spring Boot);

− Надійне зберігання й управління даними на окремому Database Server, що дозволяє використовувати реплікацію та резервне копіювання.

− Завдяки такій схемі розгортання WEB-додаток «Інтернет-магазин побутових товарів» забезпечує високу доступність, безпечну обробку запитів та ефективне зберігання даних.

2.3 Розробка бази даних для WEB-додатку

Розробка бази даних є невід'ємною частиною створення WEB-додатку «Інтернет-магазин побутових товарів». При формуванні схеми зберігання даних необхідно враховувати логічну модель предметної області, типи та формат

атрибутів, а також забезпечити високий рівень безпеки й цілісності інформації для коректної роботи системи.

База даних — це організована структура даних, що зберігається на дискових носіях і доступна програмним компонентам через спеціалізовані інтерфейси. Вибір СУБД визначає продуктивність і масштабованість рішення: реляційні СУБД (SQL) забезпечують строгі схеми і складні зв’язки між таблицями, тоді як нереляційні СУБД (NoSQL) дають гнучкість у зберіганні напівструктурованих або неструктурованих даних.

Реляційні бази даних організовують інформацію у вигляді таблиць, де кожний рядок відповідає одному запису, а стовпці описують атрибути цих записів. Зв’язки між таблицями реалізуються через первинні та зовнішні ключі, що гарантує узгодженість і підтримує цілісність транзакцій. Для роботи з такими СУБД використовується мова SQL (Structured Query Language), яка дозволяє виконувати складні запити, агрегації та оновлення даних. Перевагами реляційного підходу є формальна валідація структури та можливість складної аналітики, тоді як недолік — обмежена горизонтальна масштабованість під дуже великий навантаженням без розподілу даних.

Нереляційні бази даних представ-лені кількома моделями зберігання: документ орієнтованою, «ключ-значення», графовою тощо. Вони не вимагають попереднього визначення схеми, що дає змогу швидко еволюціонувати структуру даних, і забезпечують високу швидкість запису/читання при роботі з великими обсягами інформації. Проте відсутність жорсткої схеми ускладнює підтримку цілісності та уніфіковане виконання складних зв’язків.

З огляду на бізнес-вимоги WEB-додатку «Інтернет-магазин побутових товарів» — необхідність зберігання каталогів товарів, інформації про користувачів, замовлення, історію операцій і підтримки складних запитів (фільтрація, звітність, зведені дані) — оптимальним рішенням є використання реляційної СУБД. Реляційна модель дозволить гарантувати цілісність транзакцій, забезпечить підтримку складних JOIN-операцій і полегшить подальшу підтримку та розширення схеми даних.

Для візуалізації логічної структури бази даних використовується ER-модель (Entity-Relationship Model). Концептуальна ER-діаграма відображає основні сутності (Entity) предметної області — Товар, Користувач, Замовлення, Категорія тощо — у вигляді прямокутників; атрибути цих сутностей позначені овальними елементами; а зв’язки між сутностями позначаються ромбами з підписами, що описують тип взаємодії (наприклад, «Оформляє», «Містить»). Кратність зв’язків (1:1, 1:Б, Б:Б) вказується поряд із ребрами, що допомагає чітко визначити бізнес-правила й гарантує правильність подальших реалізаційних етапів. Представлення ER-моделі предметної області WEB-додатку інтернет-магазин побутових товарів на рисунку 2.5.

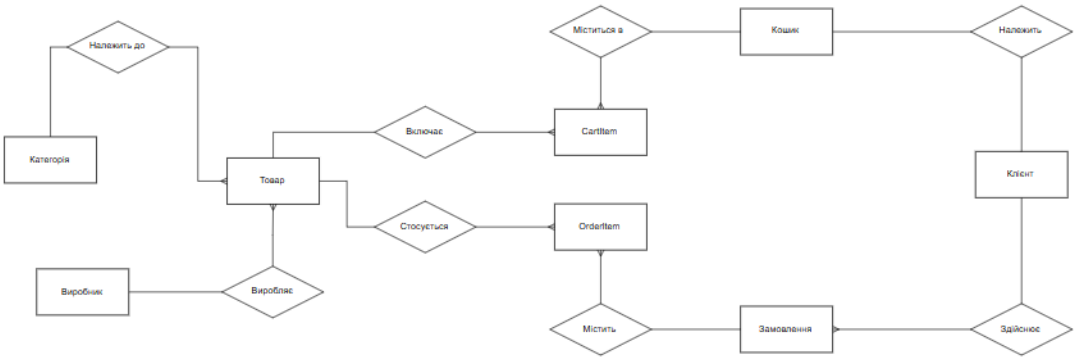


Рисунок 2.5 – ER-моделі предметної області WEB-додатку інтернет-магазин побутових товарів

У рамках розробки WEB-додатку «Інтернет-магазин побутових товарів» ключовими складовими предметної області виступають сім основних сутностей, кожна з яких виконує свою роль у забезпеченні безперебійної та зручної роботи системи. Сутність Category відповідає за класифікацію товарів за групами, вона має ідентифікатор та назву, що дозволяє клієнтам швидко зорієнтуватися у великому асортименті побутової продукції. Сутність Manufacturer зберігає інформацію про виробників — також за допомогою унікального ідентифікатора та назви — і служить джерелом даних для відображення брендів у каталозі.

Основна сутність Product включає в себе набір атрибутів, необхідних для повного опису товару: від назви й докладного текстового опису до ціни, зображення та зовнішніх ключів, які пов’язують його з категорією та виробником.

Сутність Client з її атрибутами ідентифікатора, імені, електронної пошти, захищеного пароля і платіжних даних відтворює користувача-покупця. Для кожного клієнта формується особистий Cart, що містить записи про обрані товари (CartItem) — кожен із яких включає кількість та ідентифікатор товару. Після завершення вибору товарів покупець оформлює Order, який зберігає дату й час оформлення та поточний статус, а деталізацією замовлення займається пов’язана з ним сутність OrderItem, що вказує на кількість та ціну кожного товару в замовленні.

Універсальне відношення для цієї предметної області описується як:

R(id_Category, name_Category, id_Manufacturer, name_Manufacturer, id_Product, name_Product, description, price, image_url, category_id, manufacturer_id, id_Client, name_Client, email, password, payment_info, id_Cart, client_id_Cart, id_CartItem, cart_id, product_id_CartItem, quantity_CartItem, id_Order, client_id_Order, order_date, status, id_OrderItem, order_id, product_id_OrderItem, quantity_OrderItem, unit_price)

Загальний ступінь універсального відношення становить 31, що відображає кількість всіх атрибутів, необхідних для повного опису предметної області інтернет-магазину побутових товарів.

На основі описаного було спроектовано схему бази даних WEB-додатку інтернет-магазин побутових товарів, яка зображена на рисунку 2.6.

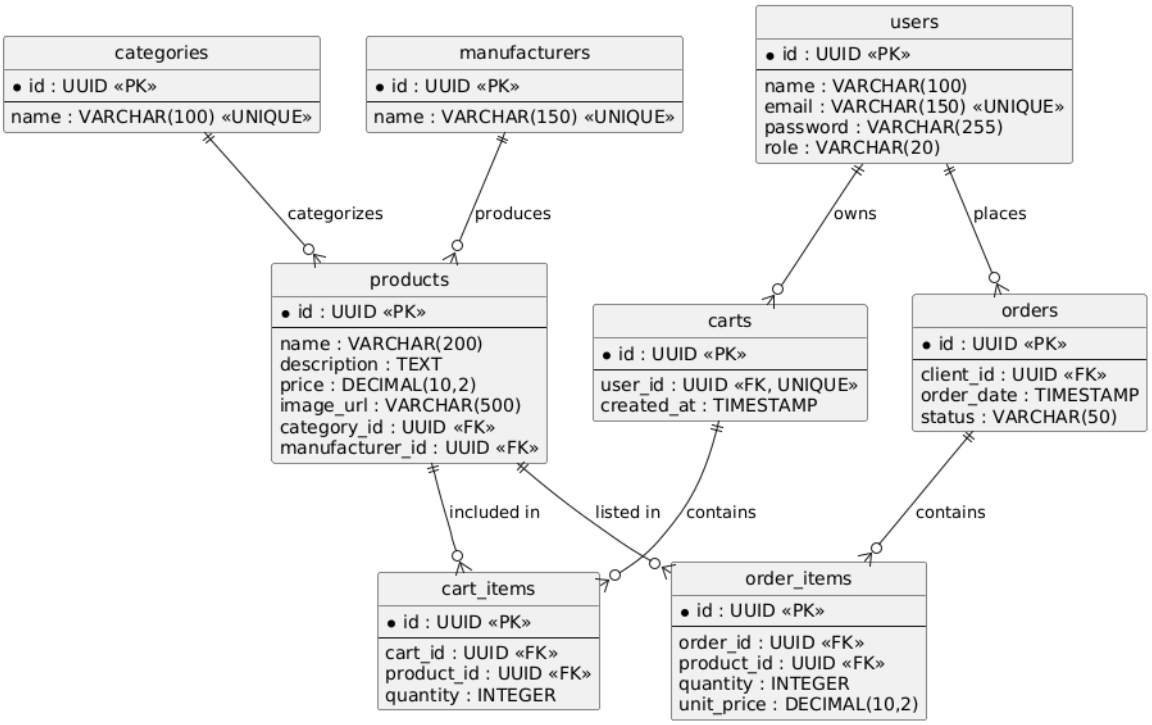


Рисунок 2.6 – Схема бази даних WEB-додатку інтернет-магазин побутових товарів

Наведена схема бази даних WEB-додатку «Інтернет-магазин побутових товарів» складається з восьми таблиць: users, categories, manufacturers, products, carts, cart_items, orders та order_items. Первинні ключі позначені як PK (Primary Key), а зовнішні ключі — як FK (Foreign Key). Кожна таблиця має чітко визначений набір атрибутів, що забезпечує однозначне зберігання та зв’язність даних:

– users (PK: id) — містить дані про всіх користувачів (клієнтів і адміністраторів), з унікальними полями email і role;

– categories (PK: id) та manufacturers (PK: id) — довідкові таблиці для класифікації товарів за категоріями та брендами;

– products (PK: id, FK: category_id → categories.id, FK: manufacturer_id → manufacturers.id) — описує товари зі зв’язками до категорій і виробників;

– carts (PK: id, FK: user_id → users.id) — зберігає поточний кошик кожного клієнта;

− cart_items (PK: id, FK: cart_id → carts.id, FK: product_id → products.id) — деталізує вміст кошика;

− orders (PK: id, FK: client_id → users.id) — фіксує офіційно оформлені замовлення;

− order_items (PK: id, FK: order_id → orders.id, FK: product_id → products.id) — деталізує товари та їхні кількості в кожному замовленні.

Нормалізація бази даних є ключовим етапом проєктування для зменшення надлишковості й запобігання аномаліям оновлення. Усі відношення відповідають вимогам Першої нормальної форми (1НФ): кожне поле містить лише атомарні значення, а повторюваних груп даних у рядках немає. Таблиці, що мають складені ключі (cart_items та order_items), не містять неключових атрибутів, тому вони також задовольняють вимогам Другої нормальної форми (2НФ), а в інших таблицях відсутні складені ключі. Для відповідності Третій нормальній формі (3НФ) необхідно, щоб жоден неключовий атрибут не мав транзитивних залежностей від первинного ключа. У нашій моделі всі атрибути безпосередньо залежать лише від свого PK і не містять транзитивних зв’язків, отже всі таблиці знаходяться в 3НФ.

Таким чином, розроблена схема реляційної бази даних є збалансованою, легко підтримуваною та готовою до масштабування відповідно до бізнес-вимог WEB-додатку «Інтернет-магазин побутових товарів»

2.4 Розробка схеми алгоритму роботи для WEB-додатку побутових товарів

Алгоритм — це впорядкована послідовність чітко визначених дій, які виконуються для досягнення певної мети. У контексті WEB-додатку алгоритм описує логіку взаємодії користувача з системою на всіх етапах: від входу до оформлення замовлення. Схематичне представлення алгоритму дозволяє наочно відобразити структуру програми, полегшує розуміння процесів та спрощує реалізацію логіки у коді.

У схемі використовуються стандартні елементи:

- овал — для початку та завершення алгоритму;
- прямокутник — для дій або процесів;
- ромб — для умовних перевірок;
- стрілки — для напрямку виконання.

I випадок

Алгоритм у випадку, коли користувач не зареєстрований складається з таких кроків:

Крок 1. Запуск головного вікна;

Крок 2. Вибір опції реєстрації;

Крок 3. Введення даних для реєстрації (ім’я, пошта, пароль);

Крок 4. Перевірка: чи реєстрація успішна? Якщо ні — повернення до кроку 2;

Крок 5. Перевірка: чи користувач є адміністратором?

Крок 6. Якщо так — перехід до гілки адміністратора, інакше — до гілки клієнта.

II випадок

Алгоритм у випадку, коли користувач зареєстрований складається з таких кроків:

Крок 1. Запуск головного вікна;

Крок 2. Відображення форми входу;

Крок 3. Введення логіну та паролю;

Крок 4. Перевірка: авторизація успішна? Якщо ні — крок 5;

Крок 5. Введення email для скидання паролю;

Крок 6. Надсилання токену на пошту;

Крок 7. Введення нового паролю;

Крок 8. Перевірка: зміна паролю успішна? Якщо ні — повернення до кроку 7;

Крок 9. Успішна авторизація → перевірка на роль (адміністратор чи клієнт).

Схема алгоритму роботи WEB-додатку інтернет магазину побутових товарів наведена на рис. 2.3.

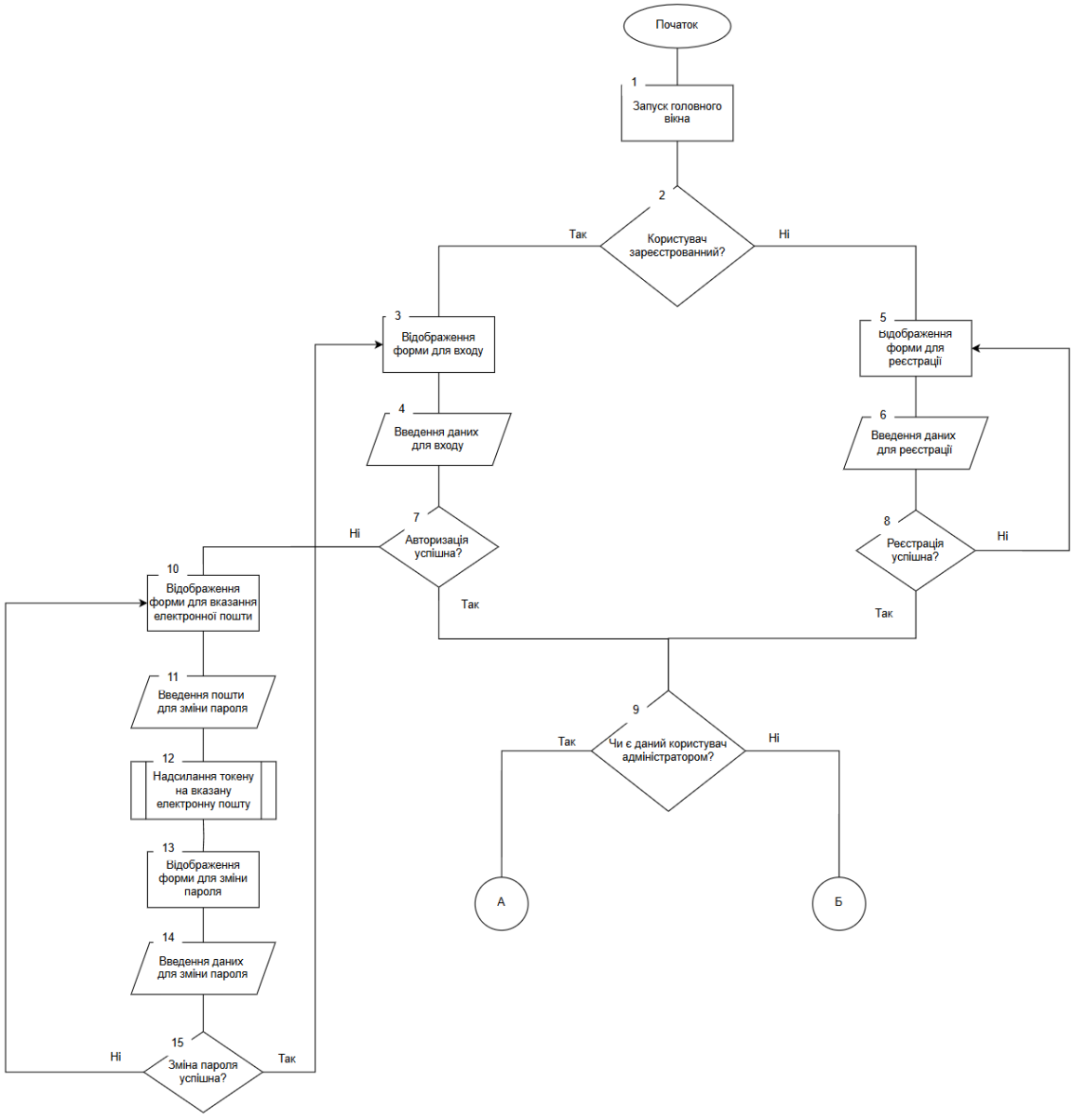


Рисунок 2.3 – Схема алгоритму роботи WEB-додатку інтернет магазину побутових товарів

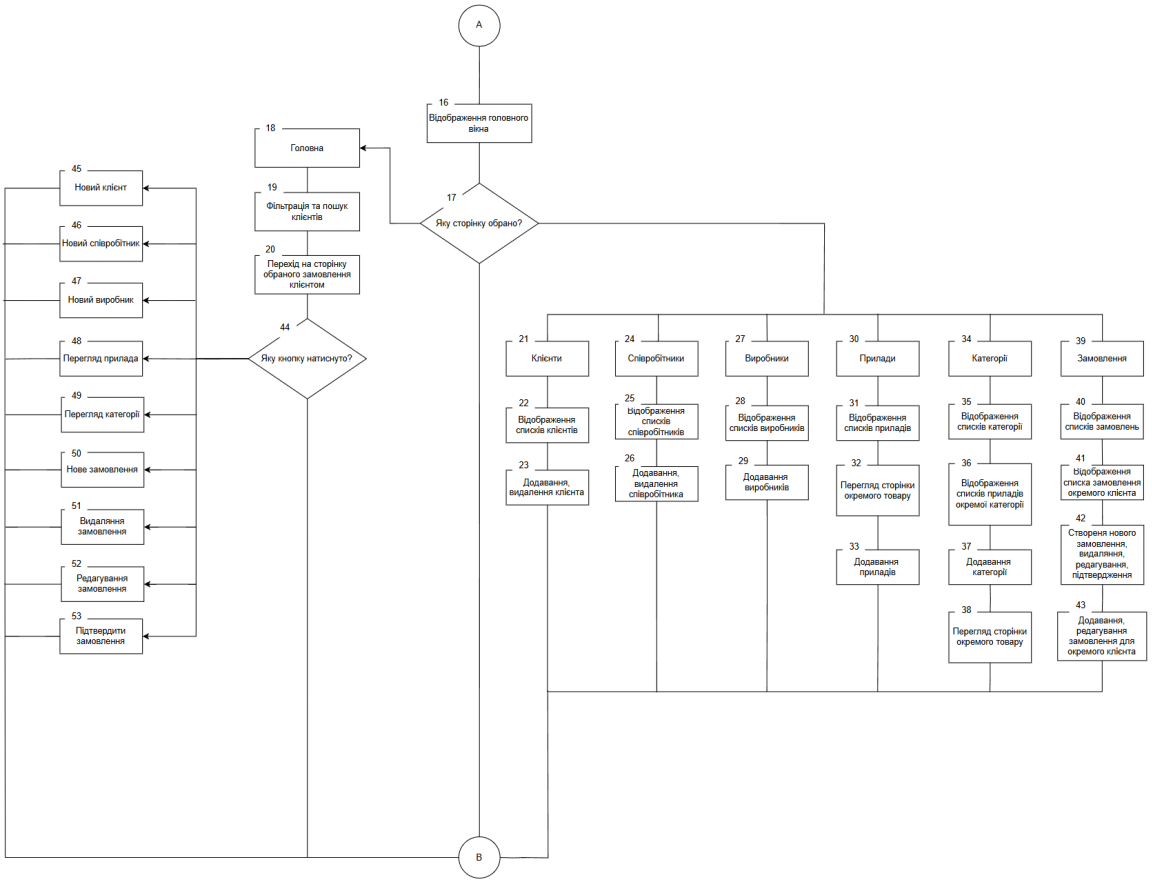


Рисунок 2.3 – Продовження

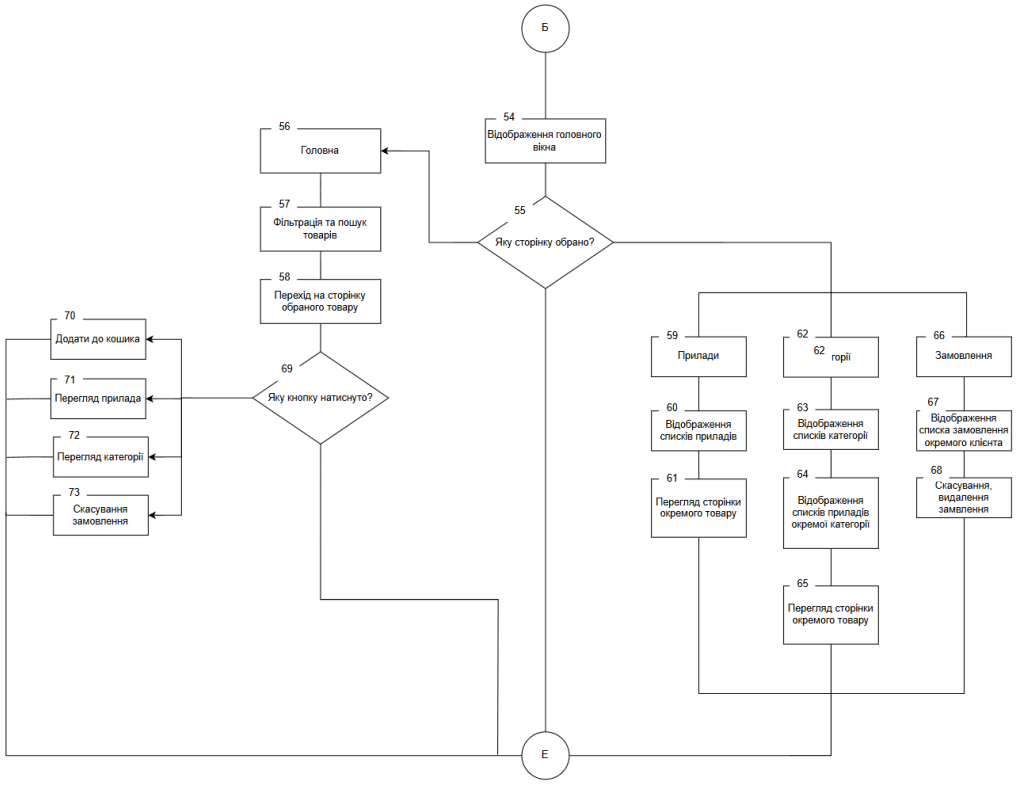


Рисунок 2.3 – Продовження

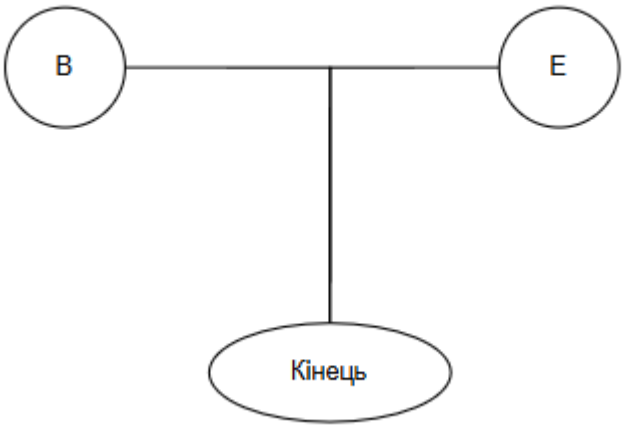


Рисунок 2.3 – Продовження

III випадок

Алгоритм у випадку, коли адміністратор взаємодіє з системою (гілка А):

Крок 1. Відображення головного вікна;

Крок 2. Вибір сторінки: "Клієнти", "Співробітники", "Виробники", "Прилади", "Категорії", "Замовлення";

Крок 3. Відповідно до вибору — відображення списків або форм;

Крок 4. Додавання/редагування/видалення даних у відповідних розділах;

Крок 5. Або адміністратор натискає на кнопку — наприклад: "Додати клієнта", "Нове замовлення", "Переглянути прилади" тощо;

Крок 6. Відбувається дія — наприклад: перегляд, додавання чи редагування сутностей;

Крок 7. Повернення до головного меню.

IV випадок

Алгоритм у випадку, коли клієнт взаємодіє з системою (гілка Б):

Крок 1. Відображення головного вікна після входу;

Крок 2. Вибір сторінки: "Прилади", "Категорії", "Замовлення";

Крок 3. Відображення вибраного розділу — список товарів, категорій або історії замовлень;

Крок 4. Вибір конкретного товару/категорії/замовлення — відкриття сторінки детального перегляду;

Крок 5. Можливість скасування або перегляду замовлення, додавання товару до кошика;

Крок 6. Повернення до головного меню або завершення взаємодії.

V випадок

Алгоритм у випадку, коли користувач здійснює дію з товаром (перегляд, додавання до кошика, скасування замовлення):

Крок 1. Пошук або фільтрація товарів;

Крок 2. Вибір конкретного товару з каталогу;

Крок 3. Перехід на сторінку товару;

Крок 4. Натискання кнопки: "Додати до кошика", "Переглянути", "Скасувати замовлення";

Крок 5. Виконання відповідної дії;

Крок 6. Вивід результату (наприклад, повідомлення про успішне скасування);

Крок 7. Повернення до головної сторінки.

VI випадок

Завершення роботи:

Крок 1. Користувач завершує взаємодію з системою (натискає "Вийти" або закриває додаток);

Крок 2. Повернення до початкового вікна (початковий стан);

Крок 3. Завершення роботи алгоритму — фіксується в схемі як "Кінець".

2.4 Висновок

У другому розділі було виконано всебічне проєктування WEB-додатку «Інтернет-магазин побутових товарів», що заклало міцну основу для подальшої реалізації. Спершу обґрунтовано багатошарову архітектуру клієнт–сервер–дані з використанням Spring Boot на сервері та Thymeleaf/Bootstrap на клієнті, що гарантує чітке розподілення відповідальностей, легке масштабування й можливість паралельної розробки фронтенду та бекенду.

Наступним етапом стала деталізація структури системи за допомогою UML-моделювання. Діаграма прецедентів окреслила основні ролі (Client, Admin) і сценарії взаємодії, діаграма класів описала ключові сутності й їх атрибути, а діаграми послідовності продемонстрували обмін повідомленнями між компонентами при виконанні критичних операцій. Діаграма розгортання показала фізичне середовище розміщення додатку, зокрема веб-сервер і СУБД.

Важливою частиною проєктування стала розробка реляційної бази даних. Було побудовано ER-модель із основними сутностями (users, categories, manufacturers, products, carts, cart_items, orders, order_items) та чітко визначеними зв’язками між ними. На її основі створена схема таблиць із первинними та зовнішніми ключами, що підтримують цілісність даних. Усі відношення приведено до третьої нормальної форми, що мінімізує надлишковість і запобігає аномаліям оновлення.

На завершальному етапі побудовані блок-схеми алгоритмів для ключових бізнес-процесів — реєстрації, керування кошиком, оформлення замовлення та адміністрування — завдяки чому логіка роботи додатку стала абсолютно прозорою. Таким чином, другий розділ забезпечив повний набір проектних артефактів: від архітектури й UML-моделей до структури бази даних і алгоритмічних описів, що дозволяє команді розробників розпочати реалізацію з чітким уявленням про всі компоненти системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИНА ПОБУТОВИХ ТОВАРІВ»

3.1 Обґрунтування та вибір мов середовищ та бібліотек

У процесі розробки програмного забезпечення надзвичайно важливим етапом є вибір відповідного технологічного стеку — тобто мов програмування, середовищ розробки та бібліотек. Такий вибір має базуватись на специфіці задачі, функціональних і нефункціональних вимогах до системи, досвіді розробника та наявності підтримки обраних рішень у спільноті.

Проєкт створення WEB-додатку інтернет магазину побутових товарів передбачає реалізацію клієнтської та серверної частин, роботу з базами даних, організацію авторизації та автентифікації користувачів, а також забезпечення гнучкого й адаптивного інтерфейсу. З огляду на це, було обрано мову програмування Java як основну для серверної логіки, фреймворк Spring Boot для створення REST API [8] та управління безпекою, а також Thymeleaf [9] для генерації динамічних HTML-сторінок. Для зберігання даних на етапі розробки використовується вбудована база H2 [10], а для остаточного розгортання можливе підключення до PostgreSQL [11] або іншої реляційної СКБД.

3.2 Обґрунтування вибору мови та бібліотек програмування

На сучасному етапі розвитку інформаційних технологій існує велика кількість мов програмування, кожна з яких має свої переваги, недоліки та галузі застосування. Для реалізації серверної частини веб-додатку необхідно було обрати таку мову, яка забезпечить високу стабільність, безпеку, масштабованість, а також підтримку великої спільноти та широкого спектра готових бібліотек.

В таблиці 3.1 наведено порівняльну характеристику популярних мов програмування, які найбільш часто використовуються для розробки веб-додатків:

Таблиця 3.1 - Порівняльну характеристику популярних мов програмування

Критерій	Java	Python	JavaScript (Node.js)	C# (.NET)
Висока продуктивність	+	-	+	+
Масштабованість	+	-	+	+
Безпека	+	-	-	+
Популярність у бізнесі (B2B)	+	-	-	+
Велика кількість фреймворків	+	+	+	+
Широка активна спільнота	+	+	+	-
Простота вивчення	-	+	+	+
Кросплатформеність	+	+	+	-
Підтримка інструментів для тестування	+	+	+	+

На основі порівняння можна зробити висновок, що Java є найкращим вибором для реалізації проєкту WEB-додатку інтернет магазину побутових товарів, оскільки вона забезпечує:

- Надійність та стабільність. Java використовується в критично важливих сферах: банківських системах, телекомунікаціях, корпоративному програмному забезпеченні. Вона має перевірену роками архітектуру та зворотну сумісність.
- Масштабованість та модульність.Завдяки об'єктно-орієнтованій природі Java, систему легко масштабувати, додавати нові функції без значних змін до існуючого коду. Це особливо важливо для WEB-додатку, де можуть згодом з’являтись додаткові ролі, аналітика, акції, API.
- Розвинена екосистема та спільнота. Java має одну з найстаріших і найпотужніших екосистем — тисячі бібліотек і рішень на будь-який випадок. Спільнота активна, тож рішення на більшість проблем легко знайти.
- Безпека. Spring Security [12] — компонент у екосистемі Java, який дозволяє швидко впровадити сучасні стандарти безпеки (аутентифікація, рольовий доступ, захист від CSRF, HTTPS тощо).
- Міжплатформеність. Java-програми можуть запускатися на будь-якій ОС, що підтримує JVM. Це дозволяє розгорнути веб-додаток як на Windows, так і на Linux-серверах.
- Підтримка багатьма фреймворками. У Java-екосистемі [13] є перевірені часом фреймворки: Spring Boot, Hibernate [14], Maven[15], які прискорюють розробку та полегшують супровід коду.

У сучасній розробці веб-застосунків важливе місце посідає не лише вибір мови програмування, але й фреймворку, який забезпечує основу для побудови архітектури, обробки запитів, роботи з базами даних, авторизації, валідації та іншої прикладної логіки. Фреймворк дозволяє зосередитися на бізнес-логіці, а не на вирішенні рутинних технічних завдань. Саме тому вибір фреймворку є стратегічно важливим і потребує уважного аналізу.

Для реалізації серверної частини WEB-додатку було розглянуто кілька популярних фреймворків, таких як:

- Spring Boot (Java);
- Django (Python);

- Express.js (JavaScript/Node.js);
- ASP.NET Core (C#);
- Laravel (PHP);
- Ruby on Rails (Ruby).

Кожен із зазначених фреймворків має свої переваги та історично сформовану нішу застосування. Наприклад, Django [19] популярний у стартап-середовищі завдяки швидкій розробці MVP-продуктів. Express.js використовується у проектах із фокусом на front-end, де серверна частина є лише легким API-шаром. Laravel і Rails активно застосовуються у середніх і малих веб-проектах, проте менш придатні для великого масштабування. ASP.NET від Microsoft найчастіше використовується в комерційних рішеннях для Windows-платформи.

Однак з урахуванням поставлених завдань — побудова стабільного, масштабованого та безпечного веб-застосунку з багатою структурою, підтримкою ролей, форм, замовлень і інтеграцією з базами даних — найбільш обґрунтованим вибором став Spring Boot.

Spring Boot став оптимальним вибором для реалізації серверної частини WEB-додатку завдяки своїй універсальності, гнучкості та широким можливостям інтеграції. Однією з найважливіших переваг Spring Boot є його здатність суттєво спростити конфігурування проєкту та зменшити кількість рутинного коду. Завдяки механізму автоматичної конфігурації, розробник має змогу швидко створити повноцінний веб-застосунок, не витрачаючи час на ручне налаштування великої кількості параметрів. Усе, що потрібно — це правильно підключити залежності, і Spring Boot самостійно ініціалізує потрібні компоненти.

Крім того, Spring Boot є частиною великої екосистеми Spring, яка вже понад два десятиліття є стандартом у корпоративній розробці. Це дозволяє безперешкодно використовувати інші модулі Spring, зокрема Spring MVC для реалізації архітектури Model-View-Controller, Spring Security для побудови

механізмів автентифікації й авторизації, Spring Data JPA для роботи з базами даних, а також модулі для валідації, обробки помилок, логування та тестування. Всі ці складові тісно інтегровані між собою, що дає змогу будувати масштабовані, гнучкі та легко підтримувані системи.

Неможливо не згадати і про те, що Spring Boot надає розробнику високий рівень безпеки. Зокрема, інтеграція з модулем Spring Security дозволяє реалізовувати різні рівні доступу на основі ролей користувача, захист від міжсайтових атак (CSRF), шифрування паролів за допомогою алгоритмів типу BCrypt [20] та інші важливі заходи безпеки. У контексті WEB-додатку, де передбачено обробку особистих даних клієнтів та замовлень, наявність таких функціональностей є критично важливою.

Ще однією визначною перевагою Spring Boot є підтримка REST API, шаблонізаторів (наприклад, Thymeleaf), а також зручні інструменти для локалізації інтерфейсу. Це особливо актуально для даного проєкту, оскільки WEB-додаток повинен бути доступним кількома мовами (українською, англійською) і Spring Boot дозволяє реалізувати багатомовність із мінімальними зусиллями.

Таким чином, Spring Boot об'єднує в собі всі ті характеристики, які є необхідними для створення сучасного, надійного, безпечного й масштабованого веб-додатку. Його широкі можливості, розвинена екосистема, активна спільнота та висока якість документації роблять його логічним вибором для реалізації кваліфікаційного проєкту з розробки WEB-додатку інтернет магазину побутових товарів.

З урахуванням поставлених вимог до системи, Java виявилась найоптимальнішою мовою програмування для реалізації WEB-додатку інтернет магазину побутових товарів. У поєднанні з фреймворком Spring Boot вона дозволяє побудувати надійний, безпечний і масштабований застосунок, що відповідає сучасним стандартам розробки програмного забезпечення. Такий вибір також забезпечує легкість підтримки, хорошу документацію і можливість у майбутньому інтегрувати нові функції без істотних змін в архітектурі проєкту.

3.3 Тестування роботи програми та аналіз результатів

Тестування є найважливішою частиною розробки WEB-додатку побутових товарів, о дозволяє знайти та виправити помилки, перевірити функціонал заданим вимогам та забезпечити надійну роботу веб-застосунку.

Для реєстрації користувача потрібно вказати ім’я, пошту, пароль, підтвердження паролю та прив’язати номер карти. Також для вільного перегляду товарів або категорії необов’язково реєструватися або авторизуватися на сайті, це потрібно тільки для того, щоб здійснити покупку зі сторони клієнта, або для адміністрування зі сторони адміністратора. Сторінка для реєстрації користувача зображено на рисунку 3.1, а для входу на рисунку 3.2.

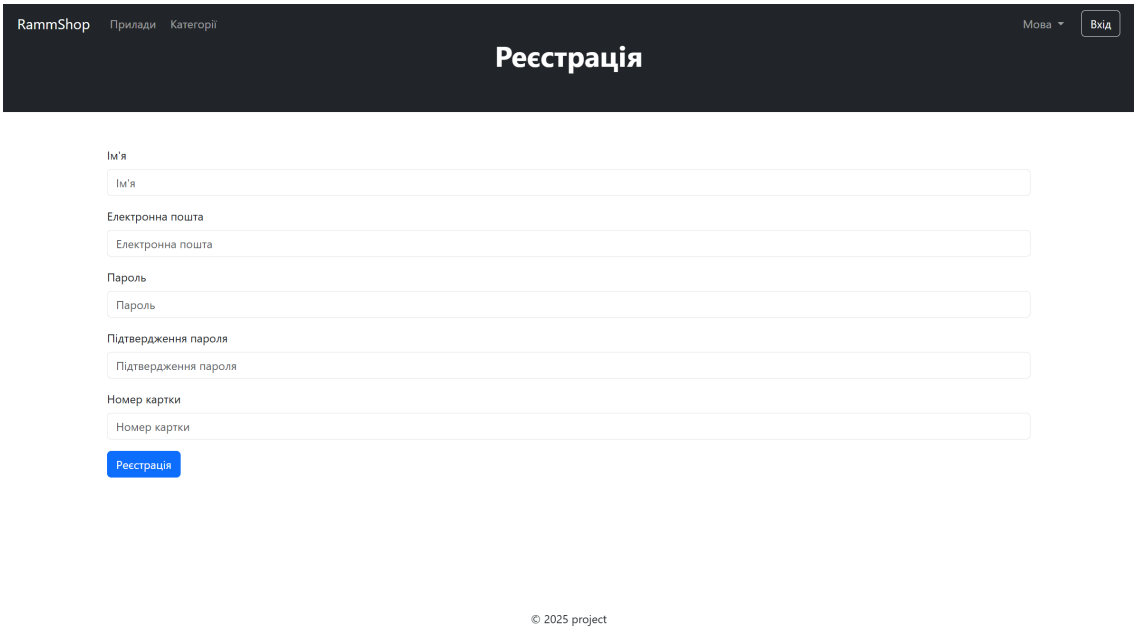


Рисунок 3.1 – Сторінка для реєстрації нового користувача

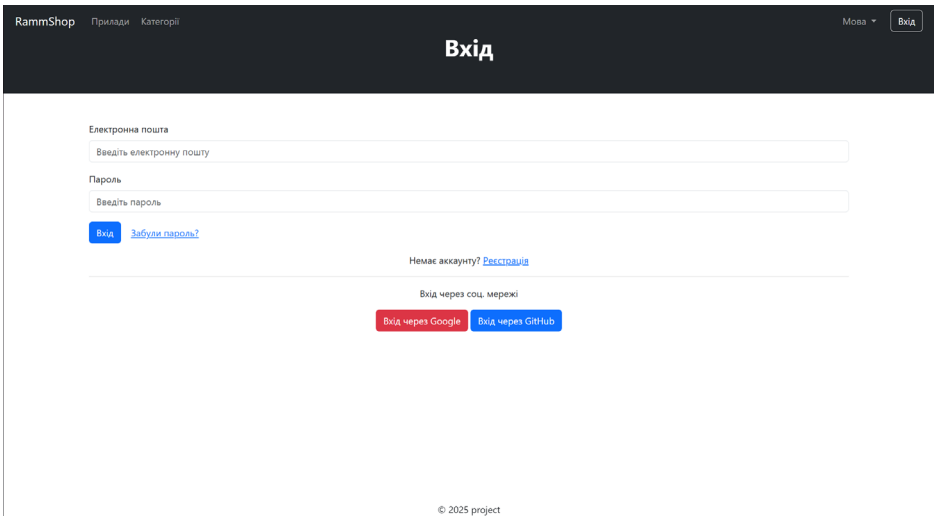


Рисунок 3.2 – Сторінка для авторизації

Після успішної авторизації відображається головна сторінка WEB-додатку, де користувач, в ролі клієнта може переходити на сторінку «Прилади», «Категорії» та «Мої Замовлення». Клієнт може продивлятися товар, шукати товар за категорією, та за назвою, виконувати фільтрацію, здійснювати покупки. Після натискання кнопки «Додати до кошика», обраний товар та обрана кількість товару потрапляє до кошика клієнта, де клієнт може подивитися своє замовлення та видалити його за потреби, та перевірити статус свого замовлення. Сторінки «Прилади», сторінка окремого товару, «Категорії» та «Мої Замовлення» зображенні на рисунках 3.3, 3.4, 3.5 та 3.6.

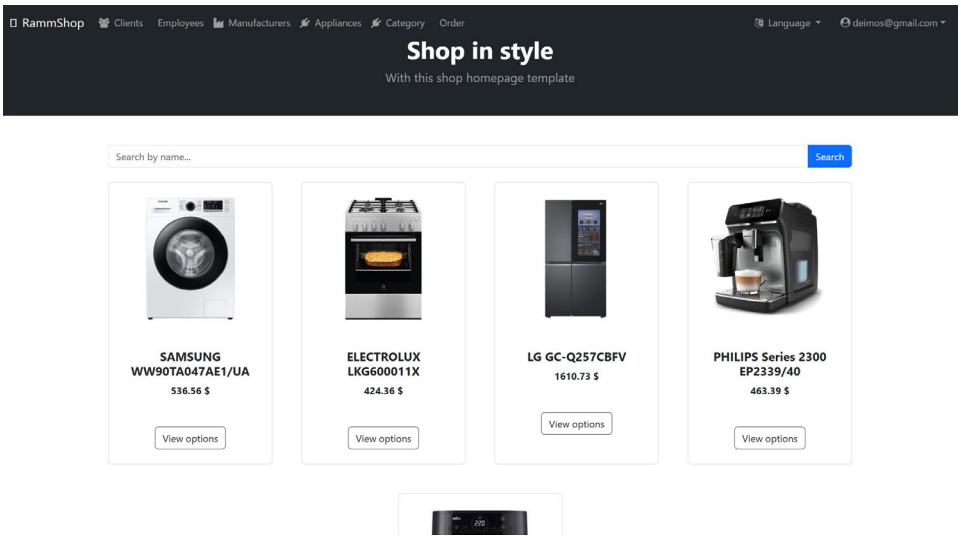


Рисунок 3.3 – Сторінка «Прилади»

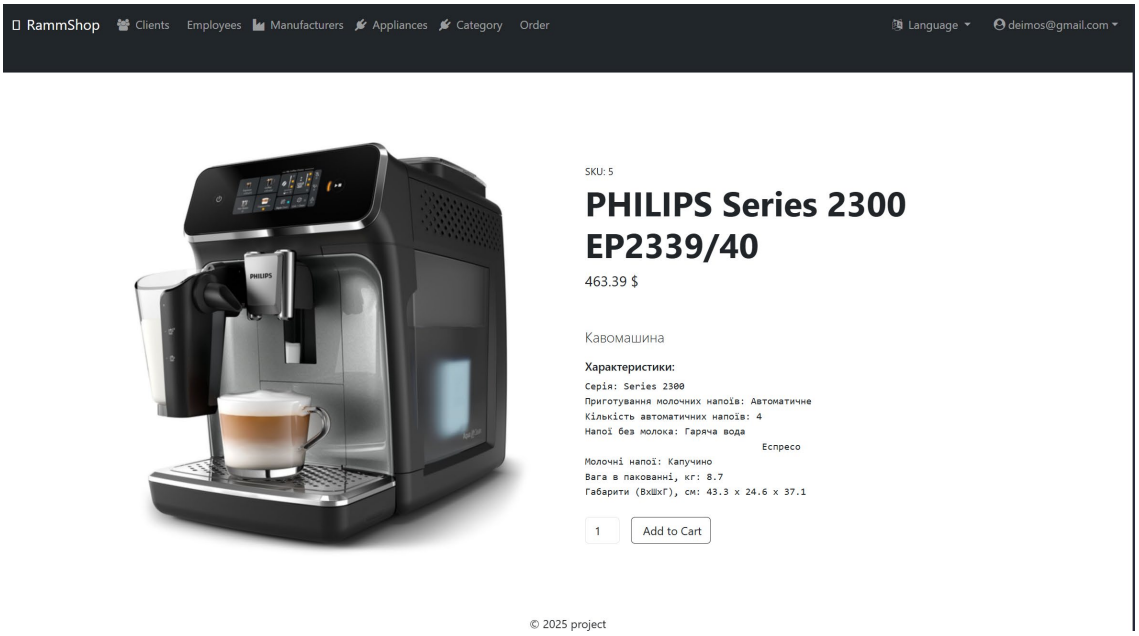


Рисунок 3.4 – Сторінка окремого товару

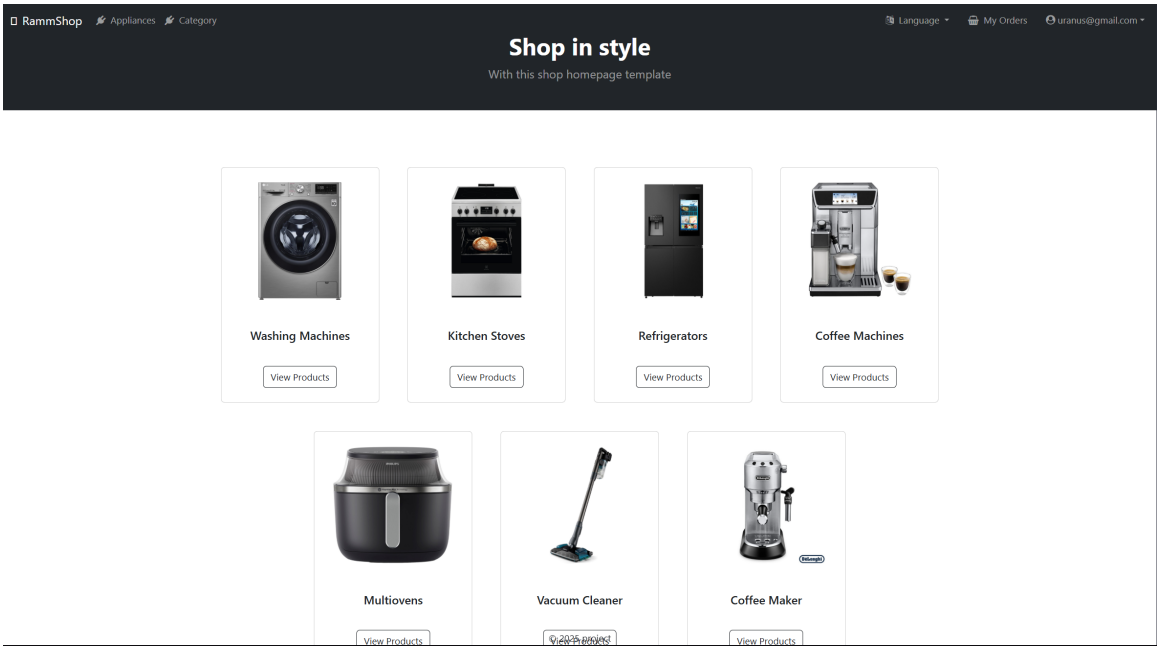


Рисунок 3.5 – Сторінка «Категорії»

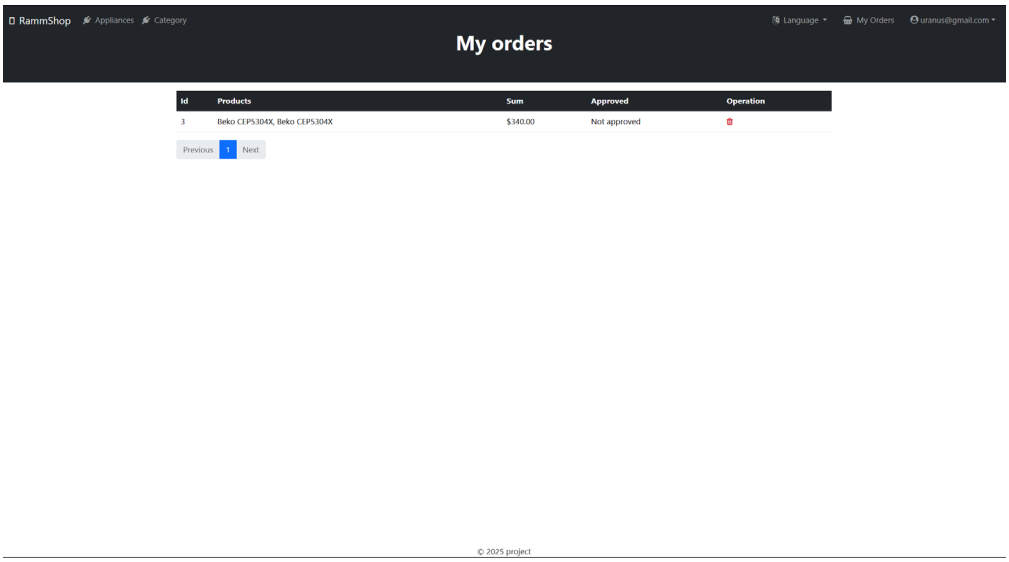


Рисунок 3.6 – Сторінка «Мої Замовлення»

Для адміністратора сайту, звісно доступно набагато більше функціоналу, ніж для клієнта. Адміністратор може повністю керувати замовленням клієнтів, скасовувати, редагувати та підтверджувати. Також найважливішою частиною адміністрування є додавання виробників, категорії, товарів. Адміністратор може додавати робочих(адміністраторів), призначати їм певний відділ, реєструвати нових клієнті за потреби, наприклад, якщо у клієнта немає акаунта і в магазині йому запропонували зареєструватися на місці. На рисунку 3.7 зображено приклад менеджменту замовлення та 3.8 зображено приклад додання товару.

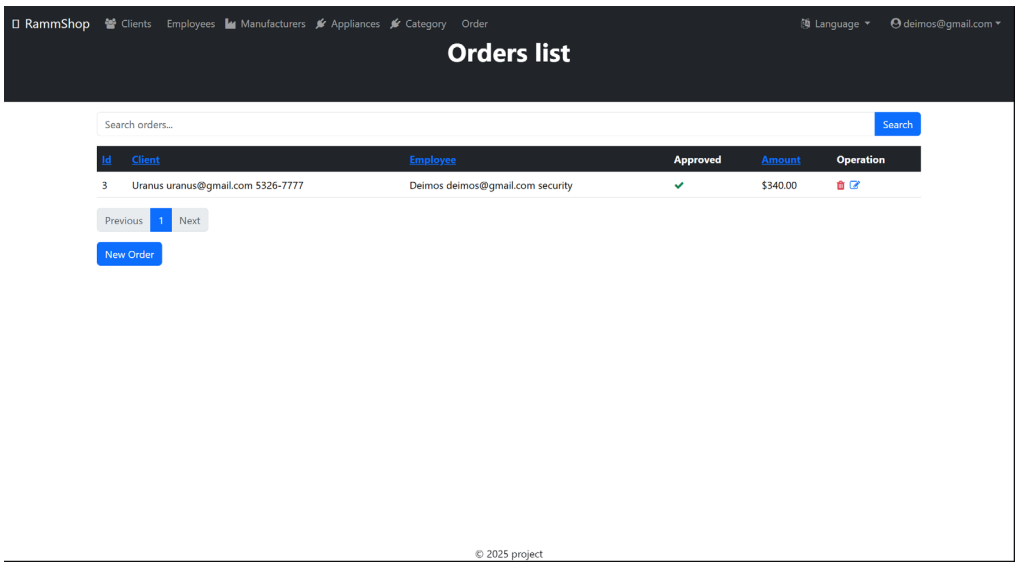


Рисунок 3.7 – Сторінка менеджменту замовлення для адміністратора

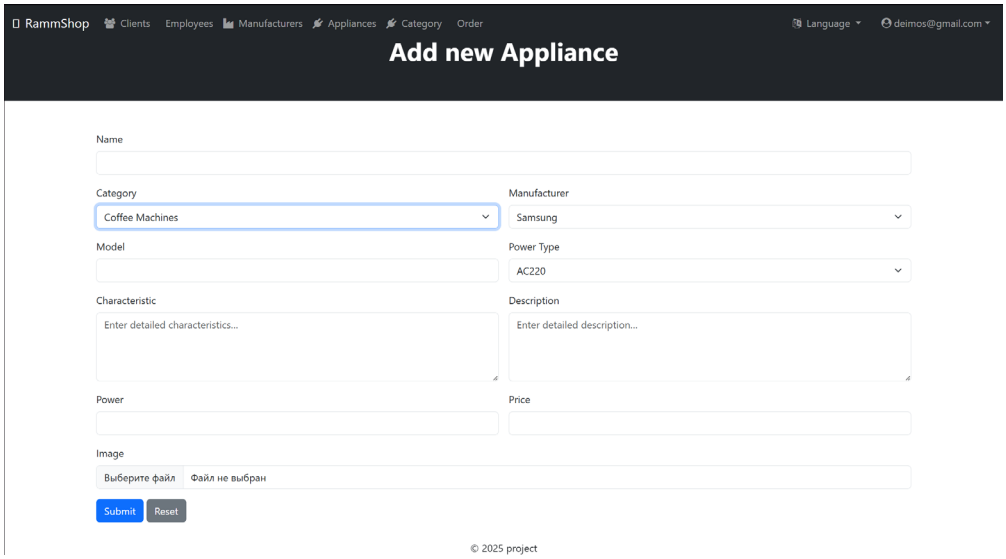


Рисунок 3.8 – Приклад додання товару

Під час тестування WEB-додатку побутових товарів було проведено реєстрацією для декількох клієнтів користувачів, кожен з них здійснив замовлення обраних товарів, з різних категорії, та замовлення були проадміністрованні адміністратором сайту. Було ретельно перевірено функціональність фільтрація та пошук за різними параметрами, список замовлення, а також було протестовано скасування замовлення клієнтом та адміністратором. Окрім цього, зі сторони адміністратора декілька раз тестувалася можливість редагування замовлення та видалення всіх клієнтів разом із замовленнями. Додатково, була перевірена валідація форм для введення даних та для задання унікального паролю, щоб гарантувати достовірність та безпеку інформації, яку вводить користувач. Проведене тестування дало в висновку, що функціональність WEB-додатку працює належним чином та повністю відповідає всім функціональним та нефункціональним вимогам, які були визначені.

Порівняльна характеристика розробленого WEB-додатка інтернет магазин побутових товарів з Веб-ресурсами наведено в таблиці 3.2.

Таблиця 3.2 - Порівняльна характеристика розробленого WEB-додатку

Критерій	Rozetka	Allo.ua	Prom.ua	Розроблений WEB-додаток
Орієнтація саме на побутові товари	-	-	-	+
Простість і зручність інтерфейсу	-	+	-	+
Якість клієнтської підтримки	-	-	-	+
Гнучкість управління замовленнями	+	+	-	+
Прозорість структури товарів	-	-	-	+
Стабільність та швидкість роботи	+	+	+	+

Отже, за результатами таблиці можна зрозуміти, що розроблений WEB-додатку магазину побутових товарів має розширені функції по відношенню до інших WEB-ресурсів, в висновку можна зробити, що мету дослідження було досягнуто.

3.4 Висновок

У третьому розділі було всебічно досліджено й обґрунтовано вибір технологічного стеку та засобів розробки для створення WEB-додатку «Інтернет-магазин побутових товарів». На початку було проведено порівняльний аналіз мов програмування та фреймворків, в якому Java і Spring Boot були визнані оптимальними для реалізації бекенд-логіки через їхню стабільність, масштабованість і широку екосистему. Для забезпечення інтуїтивного інтерфейсу клієнтської частини обрано серверний шаблонізатор Thymeleaf у поєднанні з CSS-фреймворком Bootstrap [18], що гарантує адаптивність та швидкий старт розробки.

Детально розглянуто середовище розробки IntelliJ IDEA як провідний інструмент для роботи з Java-проєктами, а також налаштування збірки та керування залежностями за допомогою Maven. Здійснено вибір вбудованої бази даних H2 для швидкого прототипування та проведення юніт-тестів із можливістю подальшої заміни на PostgreSQL у продуктивному середовищі. Для роботи з даними в ORM-шарі використано Spring Data JPA, а для безпеки — Spring Security із BCrypt-хешуванням паролів та CSRF-захистом.

Крім того, виділено роль допоміжних бібліотек: Lombok для зменшення шаблонності коду, ModelMapper для мапінгу DTO, JUnit [16] і Mockito [17] для модульного тестування. Було проаналізовано інструменти автоматизації (Git, Postman) і налаштовано логування бізнес-подій. Результатом стало створення чіткого плану впровадження, що охоплює проєктування ER-моделі, UML-діаграм (Use Case, Class), розробку прототипу бекенду та фронтенду, а також комплексне тестування функціональних компонентів.

Отже, у межах цього розділу сформовано ґрунтовну платформу для реалізації проєкту, яка забезпечує зручність, безпеку й високу продуктивність WEB-додатку «Інтернет-магазин побутових товарів», а також готовність до подальшого масштабування та інтеграції додаткових сервісів.

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи було розроблено повнофункціональний WEB-додаток «Інтернет-магазин побутових товарів». Усі поставлені завдання були успішно реалізовані:

- обґрунтовано доцільність розробки WEB-додатка для продажу побутових товарів;
- проаналізовано відомих рішень по продажу побутових товарів;
- спроектовано WEB-додаток інтернет магазин побутових товарів;
- програмно реалізовано серверну частину а також клієнтський інтерфейс WEB-додатка інтернет магазин побутових товарів;
- виконано тестування програми та проведено аналіз отриманих результатів

Під час проєктування обґрунтовано вибір багаторівневої архітектури з використанням Spring Boot і Thymeleaf, створено ER-модель бази даних у 3НФ та розроблено схему таблиць, що гарантує цілісність і узгодженість даних. UML-діаграми прецедентів, класів, послідовності та розгортання надали візуальне уявлення про структуру й взаємодію компонентів, а блок-схеми алгоритмів окреслили логіку виконання ключових сценаріїв.

Для реалізації обрано IntelliJ IDEA, Java, Spring Security, Spring Data JPA, а на фронтенді—Thymeleaf із Bootstrap, що забезпечило адаптивність інтерфейсу. Проведене тестування підтвердило відповідність розробленого WEB-додатку «Інтернет-магазин побутових товарів» заданим функціональним та безпековим вимогам, що свідчить про досягнення мети дослідження.

Розроблений програмний продукт успішно пройшов тестування. Робота виконана згідно вимог та методичних рекомендацій щодо підготовки бакалаврських кваліфікаційних робіт [21].

Мета дослідження, яка полягала в покращенні та розширенні функціональних можливостей WEB-додатку «Інтернет-магазин побутових

товарів» із одночасним підвищенням безпеки його роботи та захисту даних користувачів, досягнута завдяки впровадженню наступних переваг у порівнянні з аналогами: забезпечено інтуїтивний інтерфейс для швидкого пошуку й фільтрації товарів, реалізовано персоналізований кабінет із історією замовлень і захищеним обміном даних, а також створено гнучку систему керування кошиком і оформлення замовлень. Крім того, застосовано багаторівневу систему безпеки з BCrypt-хешуванням паролів, CSRF-захистом і розмежуванням прав доступу для клієнтів та адміністраторів, що гарантує конфіденційність і цілісність інформації користувачів.

Отже, всі поставлені задачі було виконано, а мету роботи досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Нілова М.Г., Сілагін О.В. Розробка web-додатку «Інтернет магазин побутових товарів» [Електронний ресурс] // Молодь в науці: дослідження, проблеми, перспективи (МН-2025): Інтелектуальні інформаційні технології та автоматизація. – Режим доступу: https://conferences.vntu.edu.ua/index.php/mn/mn2025/paper/view/25335 – Дата звернення: 07.06.2025.

2. Rozetka. ВТ. Інтернет-магазин “Rozetka” [Електронний ресурс]. – Режим доступу: https://bt.rozetka.com.ua/ua/ – Дата звернення: 21.05.2025.

3. Allo. Побутова техніка в м. Вінниця [Електронний ресурс]. – Режим доступу: https://allo.ua/ua/vinnicja/bytovaya-tehnika/ – Дата звернення: 21.05.2025.

4. PROM.UA. Техніка і електроніка [Електронний ресурс]. – Режим доступу: https://prom.ua/ua/Tehnika-i-elektronika?utm_source=google_pmax&utm_medium=cpc&utm_content=pmax&utm_campaign=Pmax_cpa_1_50_tehnika_i_elektronika_5297199152&gad_source=1&gad_campaignid=20983228748&gclid=Cj0KCQjw0LDBBhCnARIsAMpYlAqQMyf9xqnMM9k7E_g0i_Q9e5HuCp8KgscD3ME7xwZE6-3NUAANOsAaAnzBEALw_wcB – Дата звернення: 21.05.2025.

5. Pivotal Software, Inc. Spring Boot Reference Documentation [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-boot/index.html – Дата звернення: 21.05.2025.

6. VMware, Inc. Spring Framework Reference Documentation: Web MVC [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-framework/reference/web/webmvc.html – Дата звернення: 21.05.2025.

7. Каграманова Ю. Як будувати UML-діаграми. Розбираємо три найпопулярніші варіанти [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/40575/ – Дата звернення: 21.05.2025.

8. Liew Z. Як правильно працювати з REST API [Електронний ресурс]. – Режим доступу: https://itvdn.com/ua/blog/article/rest-api-18 – Дата звернення: 21.05.2025.

9. Thymeleaf. Thymeleaf: Java XML/XHTML/HTML5 template engine [Електронний ресурс]. – Режим доступу: https://www.thymeleaf.org – Дата звернення: 21.05.2025.

10. H2 Database Engine. H2 Database Engine: Official Documentation [Електронний ресурс]. – Режим доступу: https://www.h2database.com/html/main.html – Дата звернення: 21.05.2025.

11. PostgreSQL Global Development Group. PostgreSQL Documentation [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/ – Дата звернення: 21.05.2025.

12. Pivotal Software, Inc. Spring Security [Електронний ресурс]. – Режим доступу: https://spring.io/projects/spring-security – Дата звернення: 21.05.2025.

13. Petryk A. Java Digest #11: State of Java Ecosystem ‘24 & Spring, Spring, Spring [Електронний ресурс]. – Режим доступу: https://dou.ua/forums/topic/49182/ – Дата звернення: 21.05.2025.

14. Hibernate ORM. Hibernate ORM: Relational persistence for Java [Електронний ресурс]. – Режим доступу: https://hibernate.org – Дата звернення: 21.05.2025.

15. Apache Software Foundation. Apache Maven [Електронний ресурс]. – Режим доступу: https://maven.apache.org – Дата звернення: 21.05.2025.

16. JUnit 5 [Електронний ресурс] // Режим доступу: https://junit.org/junit5/ ; Дата звернення: 29.05.2025.

17. Mockito [Електронний ресурс] // Режим доступу: https://site.mockito.org ; Дата звернення: 29.05.2025.

18. Bootstrap [Електронний ресурс] // Режим доступу: https://getbootstrap.com ; Дата звернення: 29.05.2025.

19. Django Software Foundation. Django [Електронний ресурс]. – Режим доступу: https://www.djangoproject.com – Дата звернення: 31.05.2025.

20. Pivotal Software, Inc. Spring Security Reference Documentation: Crypto – BCrypt class [Електронний ресурс]. – Режим доступу: https://docs.spring.io/spring-security/site/docs/current/api/org/springframework/security/crypto/bcrypt/BCrypt.html – Дата звернення: 31.05.2025.

21. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. Вінниця : ВНТУ, 2023. (PDF, 54 с.).

ДОДАТКИ

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка WEB-додатку «Інтернет магазин побутових товарів»

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 4,33 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✓ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

☐ У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

☐ У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____ _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Сілагін О.В.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Нілова М.Г.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ)
ЛІСТИНГ ПРОГРАМИ

```
package com.epam.rd.autocode.assessment.appliances.controller;


import com.epam.rd.autocode.assessment.appliances.dto.ApplianceDTO;
import com.epam.rd.autocode.assessment.appliances.dto.CategoryDTO;
import com.epam.rd.autocode.assessment.appliances.dto.ManufacturerDTO;
import com.epam.rd.autocode.assessment.appliances.model.Appliance;
import com.epam.rd.autocode.assessment.appliances.model.Category;
import com.epam.rd.autocode.assessment.appliances.model.PowerType;
import com.epam.rd.autocode.assessment.appliances.service.ApplianceService;
import com.epam.rd.autocode.assessment.appliances.service.CategoryService;
import com.epam.rd.autocode.assessment.appliances.service.ManufacturerService;
import lombok.RequiredArgsConstructor;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.IOException;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;
```

```
import java.util.List;

@Controller
@RequestMapping("/appliances")
@RequiredArgsConstructor
public class ApplianceController {


    private final ApplianceService applianceService;
    private final ManufacturerService manufacturerService;
    private final CategoryService categoryService;
    private final String uploadDir = "./uploads/images/appliances/";

    @GetMapping
    public String listAppliances(
            Model model,
            @RequestParam(value = "categoryId", required = false) Long categoryId,
            @RequestParam(value = "search", required = false) String search,
            @PageableDefault(size = 8, sort = {"id", "name", "category", "model", "manufacturer",
"powerType", "power", "price"}, direction = Sort.Direction.ASC) Pageable pageable) {
        Page<ApplianceDTO> appliancesPage;
        if (categoryId != null) {
            appliancesPage = applianceService.getByCategory(categoryId, pageable);
            model.addAttribute("selectedCategoryId", categoryId);
        }
        else if (search != null && !search.trim().isEmpty())
            appliancesPage = applianceService.searchAppliances(search, pageable);
        else
            appliancesPage = applianceService.getAppliances(pageable);

        List<CategoryDTO> allCats = categoryService
                .getAllCategories(PageRequest.of(0, 10))
                .getContent();

        model.addAttribute("appliancesPage", appliancesPage);
```

```
        model.addAttribute("search", search);
        model.addAttribute("categories", allCats);
        return "appliance/appliances";
    }

    @GetMapping("/add")
    public String showAddApplianceForm(Model model,
                                       @RequestParam(value = "size", defaultValue = "100") int size) {

        Pageable pageable = PageRequest.of(0, size, Sort.by("id").ascending());
        List<ManufacturerDTO> manufacturerDTO =
manufacturerService.getAllManufacturers(pageable).getContent();
        List<CategoryDTO> categoriesDTO = categoryService
                .getAllCategories(PageRequest.of(0, size, Sort.by("name")))
                .getContent();
        model.addAttribute("categories", categoriesDTO);
        model.addAttribute("powerTypes", PowerType.values());
        model.addAttribute("manufacturers", manufacturerDTO);
        model.addAttribute("appliance", new Appliance());
        return "appliance/newAppliance";
    }


    @PostMapping("/add-appliance")
    public String addAppliance(@ModelAttribute Appliance appliance,
                               @RequestParam("categoryId")   Long categoryId,
                               @RequestParam("manufacturerId") Long manufacturerId,
                               @RequestParam("imageFile") MultipartFile imageFile) throws IOException {

        categoryService.getById(categoryId).ifPresent(appliance::setCategory);
        manufacturerService.getById(manufacturerId).ifPresent(appliance::setManufacturer);

        Appliance saved = applianceService.createAppliance(appliance);

        if (!imageFile.isEmpty()) {
```

```
            String original = imageFile.getOriginalFilename();
            String ext = original.contains(".")
                    ? original.substring(original.lastIndexOf('.') + 1)
                    : "jpg";
            String filename = saved.getId() + "." + ext;
            Path uploadPath = Paths.get(uploadDir);
            if (Files.notExists(uploadPath)) {
               Files.createDirectories(uploadPath);
            }
            try (InputStream in = imageFile.getInputStream()) {
               Files.copy(in, uploadPath.resolve(filename),
StandardCopyOption.REPLACE_EXISTING);
            }
            saved.setImageUrl("/images/appliances/" + filename);
            applianceService.createAppliance(saved);
        }

        return "redirect:/appliances";
    }


    @GetMapping("/{id}")
    public String showApplianceDetail(@PathVariable Long id, Model model) {
        ApplianceDTO appliance = applianceService.getApplianeById(id);
        model.addAttribute("appliance", appliance);
        return "appliance/applianceDetail";
    }
package com.epam.rd.autocode.assessment.appliances.controller;

import com.epam.rd.autocode.assessment.appliances.dto.CategoryDTO;
import com.epam.rd.autocode.assessment.appliances.model.Category;
import com.epam.rd.autocode.assessment.appliances.service.CategoryService;
import java.io.IOException;
import jakarta.mail.Multipart;
import lombok.RequiredArgsConstructor;
```

```
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.util.StringUtils;
import org.springframework.web.bind.annotation.*;
import org.springframework.web.multipart.MultipartFile;

import java.io.File;
import java.io.InputStream;
import java.nio.file.Files;
import java.nio.file.Path;
import java.nio.file.Paths;
import java.nio.file.StandardCopyOption;


@Controller
@RequestMapping("/categories")
@RequiredArgsConstructor
public class CategoryController {

    private final CategoryService categoryService;
    private final String uploadDir = "./uploads/images/categories/";


    @GetMapping
    public String listCategories(
            Model model,
            @PageableDefault(size = 8, sort = {"id", "name"}, direction = Sort.Direction.ASC) Pageable
pageable){

        Page<CategoryDTO> categoryDTOPage;
        categoryDTOPage = categoryService.getAllCategories(pageable);
```

```
        model.addAttribute("categories", categoryDTOPage.getContent());
        model.addAttribute("categoryPage", categoryDTOPage);
        return "category/categories";
    }

    @GetMapping("/add")
    public String showAddCategoriesFrom(Model model) {
        model.addAttribute("category", new Category());
        return "category/newCategory";
    }

    @PostMapping("/add-category")
    public String createCategory(@ModelAttribute Category category,
                                 @RequestParam("imageFile") MultipartFile imageFile) throws IOException {
        Category saved = categoryService.createCategory(category);

        if (!imageFile.isEmpty()) {
            String original = imageFile.getOriginalFilename();
            String ext = original.contains(".")
                    ? original.substring(original.lastIndexOf('.') + 1)
                    : "jpg";
            String filename = saved.getId() + "." + ext;
            Path uploadPath = Paths.get(uploadDir);
            if (Files.notExists(uploadPath)) {
                Files.createDirectories(uploadPath);
            }
            try (InputStream in = imageFile.getInputStream()) {
                Files.copy(in, uploadPath.resolve(filename),
StandardCopyOption.REPLACE_EXISTING);
            }
            saved.setImageUrl("/images/categories/" + filename);
            categoryService.createCategory(saved);
        }
        return "redirect:/categories";
    }
```

```
package com.epam.rd.autocode.assessment.appliances.controller;

import com.epam.rd.autocode.assessment.appliances.dto.ClientDTO;
import com.epam.rd.autocode.assessment.appliances.model.Client;
import com.epam.rd.autocode.assessment.appliances.service.ClientService;
import lombok.RequiredArgsConstructor;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;


@Controller
@RequestMapping("/clients")
@RequiredArgsConstructor
public class ClientController {

    private final ClientService clientService;

    @GetMapping
    public String listClients(
            Model model,
            @RequestParam(value = "search", required = false) String search,
            @PageableDefault(size = 5, sort = {"id", "name", "email", "card"}, direction =
Sort.Direction.ASC) Pageable pageable) {
        Page<ClientDTO> clientsPage;
        if (search != null && !search.trim().isEmpty())
            clientsPage = clientService.searchClients(search, pageable);
        else
            clientsPage = clientService.getClients(pageable);
```

```
        model.addAttribute("clientsPage", clientsPage);
        model.addAttribute("search", search);
        return "client/clients";
    }

    @GetMapping("/add")
    public String showAddClientForm(Model model) {
        model.addAttribute("client", new Client());
        return "client/newClient";
    }

    @PostMapping("/add-client")
    public String addClient(Client client) {
        clientService.createClient(client);
        return "redirect:/clients";
    }

    @GetMapping("/{id}/delete")
    public String deleteClient(@PathVariable Long id) {
        clientService.deleteClient(id);
        return "redirect:/clients";
    }
}
package com.epam.rd.autocode.assessment.appliances.controller;

import com.epam.rd.autocode.assessment.appliances.dto.EmployeeDTO;
import com.epam.rd.autocode.assessment.appliances.model.Employee;
import com.epam.rd.autocode.assessment.appliances.service.EmployeeService;
import lombok.RequiredArgsConstructor;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
```

```
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;


@Controller
@RequestMapping("/employees")
@RequiredArgsConstructor
public class EmployeeController {

    private final EmployeeService employeeService;

    @GetMapping
    public String listEmployees(
            Model model,
            @RequestParam(value = "search", required = false) String search,
            @PageableDefault(size = 5, sort = {"id", "name", "email", "department"}, direction =
Sort.Direction.ASC) Pageable pageable) {
        Page<EmployeeDTO> employeePage;
        if (search != null && !search.trim().isEmpty())
            employeePage = employeeService.searchEmployees(search, pageable);
        else
            employeePage = employeeService.getEmployees(pageable);
        model.addAttribute("employeePage", employeePage);
        model.addAttribute("search", search);
        return "employee/employees";
    }

    @GetMapping("/add")
    public String showAddEmployeeForm(Model model) {
        model.addAttribute("employee", new Employee());
        return "employee/newEmployee";
    }

    @PostMapping("/add-employee")
```

```
    public String addEmployee(@ModelAttribute Employee employee) {
        employeeService.createEmployee(employee);
        return "redirect:/employees";
    }

    @GetMapping("/{id}/delete")
    public String deleteEmployee(@PathVariable Long id) {
        employeeService.deleteEmployee(id);
        return "redirect:/employees";
    }
}
package com.epam.rd.autocode.assessment.appliances.controller;

import com.epam.rd.autocode.assessment.appliances.aspect.Loggable;
import com.epam.rd.autocode.assessment.appliances.config.SecurityConfig;
import com.epam.rd.autocode.assessment.appliances.dto.ClientDTO;
import com.epam.rd.autocode.assessment.appliances.model.Client;
import com.epam.rd.autocode.assessment.appliances.service.ClientService;
import jakarta.servlet.http.Cookie;
import jakarta.servlet.http.HttpServletRequest;
import jakarta.servlet.http.HttpServletResponse;
import jakarta.validation.Valid;
import lombok.RequiredArgsConstructor;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
import org.springframework.http.HttpHeaders;
import org.springframework.http.HttpStatus;
import org.springframework.http.ResponseEntity;
import org.springframework.security.authentication.AuthenticationManager;
import org.springframework.security.authentication.UsernamePasswordAuthenticationToken;
import org.springframework.security.core.Authentication;
import org.springframework.security.core.AuthenticationException;
import org.springframework.security.core.context.SecurityContextHolder;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
```

```
import org.springframework.validation.BindingResult;
import org.springframework.web.bind.annotation.*;
import org.springframework.context.i18n.LocaleContextHolder;
import org.springframework.web.client.RestTemplate;

import java.util.Locale;

@Controller
@RequestMapping("/")
@RequiredArgsConstructor
public class IndexController {

    private final ClientService clientService;
    private final AuthenticationManager authenticationManager;



    @Loggable
    @GetMapping
    public String index(Model model) {
        String currentLanguage = LocaleContextHolder.getLocale().getLanguage();
        model.addAttribute("currentLanguage", currentLanguage);
        return "index";
    }

    @GetMapping("/change-lang")
    public String changeLanguage(@RequestParam("lang") String lang, HttpServletRequest request)
{
        Locale locale = new Locale(lang);
        LocaleContextHolder.setLocale(locale);
        return "redirect:/";
    }

    @GetMapping("/login")
    public String login(@RequestParam(value = "error", required = false) String error,
                        Model model) {
```

```
    if (error != null) {
        model.addAttribute("error", "Пожалуйста, введите корректные данные");
    }
    return "login";
}

@PostMapping("/login")
public String loginClient(@RequestParam("username") String username,
                          @RequestParam("password") String password,
                          Model model) {
    try {
        UsernamePasswordAuthenticationToken authRequest =
                new UsernamePasswordAuthenticationToken(username, password);
        Authentication authentication = authenticationManager.authenticate(authRequest);
        SecurityContextHolder.getContext().setAuthentication(authentication);
        return "redirect:/";
    } catch (Exception e) {
        model.addAttribute("error", "Неверное имя пользователя или пароль");
        return "login";
    }
}


@GetMapping("/logout")
public String logout() {
    return "logout";
}

@GetMapping("/register")
public String register(Model model) {
    model.addAttribute("client", new Client());
    return "register";
}

@PostMapping("/register")
```

```
    public String registerAdd(@Valid @ModelAttribute("client") ClientDTO clientDTO,
                              BindingResult bindingResult,  @RequestParam("confirmPassword") String
confirmPassword,
                              Model model) {
        if (!clientDTO.password().equals(confirmPassword))
            bindingResult.rejectValue("password", "error.client", "Пароли не совпадают");
        if (bindingResult.hasErrors())
            return "register";

        Client client = new Client();
        client.setName(clientDTO.name());
        client.setEmail(clientDTO.email());
        client.setPassword(clientDTO.password());
        client.setCard(clientDTO.card());
        clientService.createClient(client);
        return "redirect:/login";
    }
}
package com.epam.rd.autocode.assessment.appliances.controller;

import com.epam.rd.autocode.assessment.appliances.dto.ManufacturerDTO;
import com.epam.rd.autocode.assessment.appliances.model.Manufacturer;
import com.epam.rd.autocode.assessment.appliances.service.ManufacturerService;
import lombok.RequiredArgsConstructor;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.web.PageableDefault;
import org.springframework.stereotype.Controller;
import org.springframework.ui.Model;
import org.springframework.web.bind.annotation.*;
import org.slf4j.Logger;
import org.slf4j.LoggerFactory;
```

```
@Controller
@RequestMapping("/manufacturers")
@RequiredArgsConstructor
public class ManufacturerController {

    private final ManufacturerService manufacturerService;

    @GetMapping
    public String listManufacturer(
            Model model,
            @RequestParam(value = "search", required = false) String search,
            @PageableDefault(size = 5, sort = {"id", "name"}, direction = Sort.Direction.ASC) Pageable
pageable) {

        Page<ManufacturerDTO> manufacturerPage;
        if (search != null && !search.trim().isEmpty()) {
            manufacturerPage = manufacturerService.searchManufacturers(search, pageable);
        } else {
            manufacturerPage = manufacturerService.getAllManufacturers(pageable);
        }
        model.addAttribute("manufacturerPage", manufacturerPage);
        model.addAttribute("search", search);
        return "manufacture/manufacturers";
    }

    @GetMapping("/add")
    public String showAddManufacturerForm(Model model) {
        model.addAttribute("manufacturer", new Manufacturer());
        return "manufacture/newManufacturer";
    }

    @PostMapping("/add-manufacturer")
    public String createManufacturer(@ModelAttribute(name = "manufacturer") Manufacturer
manufacturer){
```

```
        manufacturerService.createManufacturer(manufacturer);
        return "redirect:/manufacturers";
    }

}
```

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ)

ГРАФІЧНА ЧАСТИНА

РОЗРОБКА WEB-ДОДАТКУ «ІНТЕРНЕТ МАГАЗИН ПОБУТОВИХ ТОВАРІВ»

Виконала: студентка 4 курсу, групи 4КН-21б
спеціальності 122 – Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Нілова М.Г
(прізвище та ініціали)

Керівник: к.т.н., доцент каф. КН
Сілагін О.В
(прізвище та ініціали)
«3» червня 2025 р.

Вінниця ВНТУ - 2025 рік

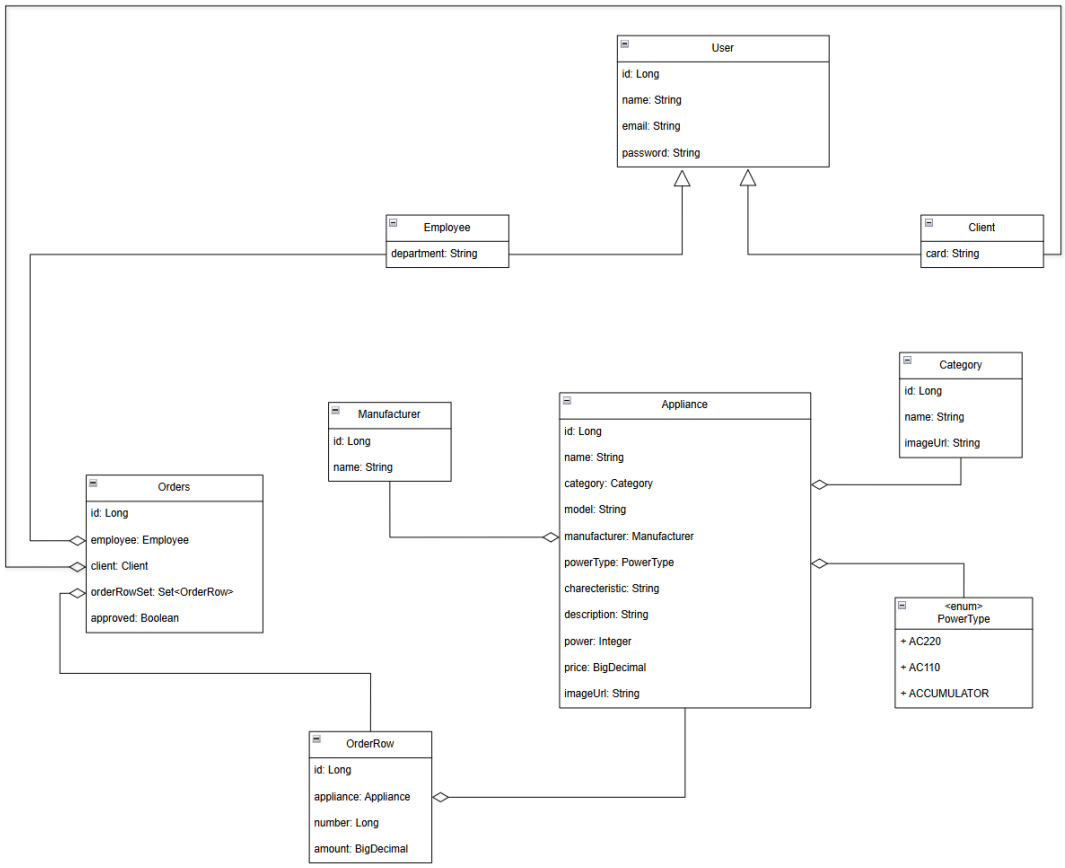


Рисунок В.1 – UML-діаграма класів побутового магазину

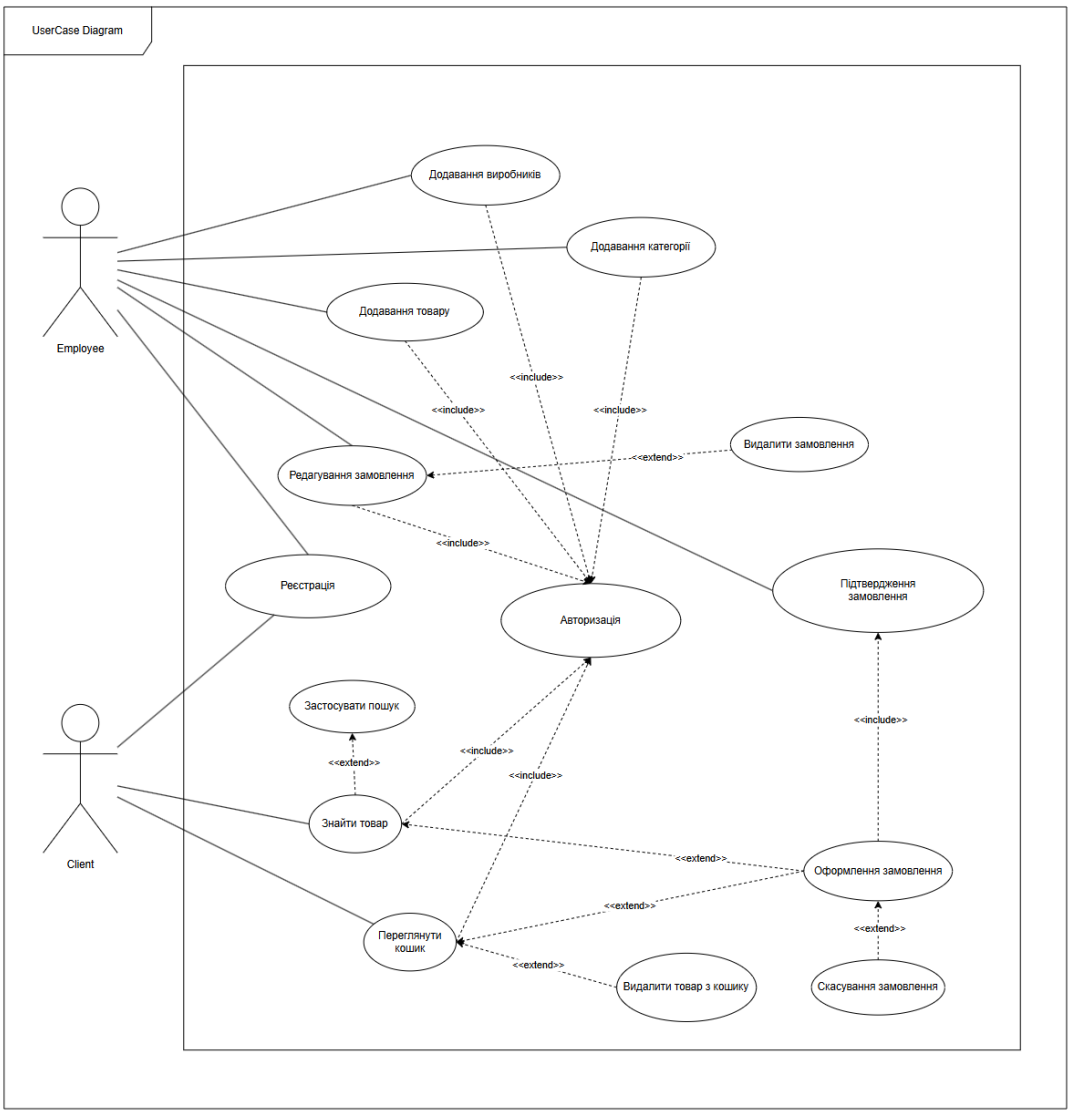


Рисунок В.2 – UML-діаграма прецедентів побутового магазину

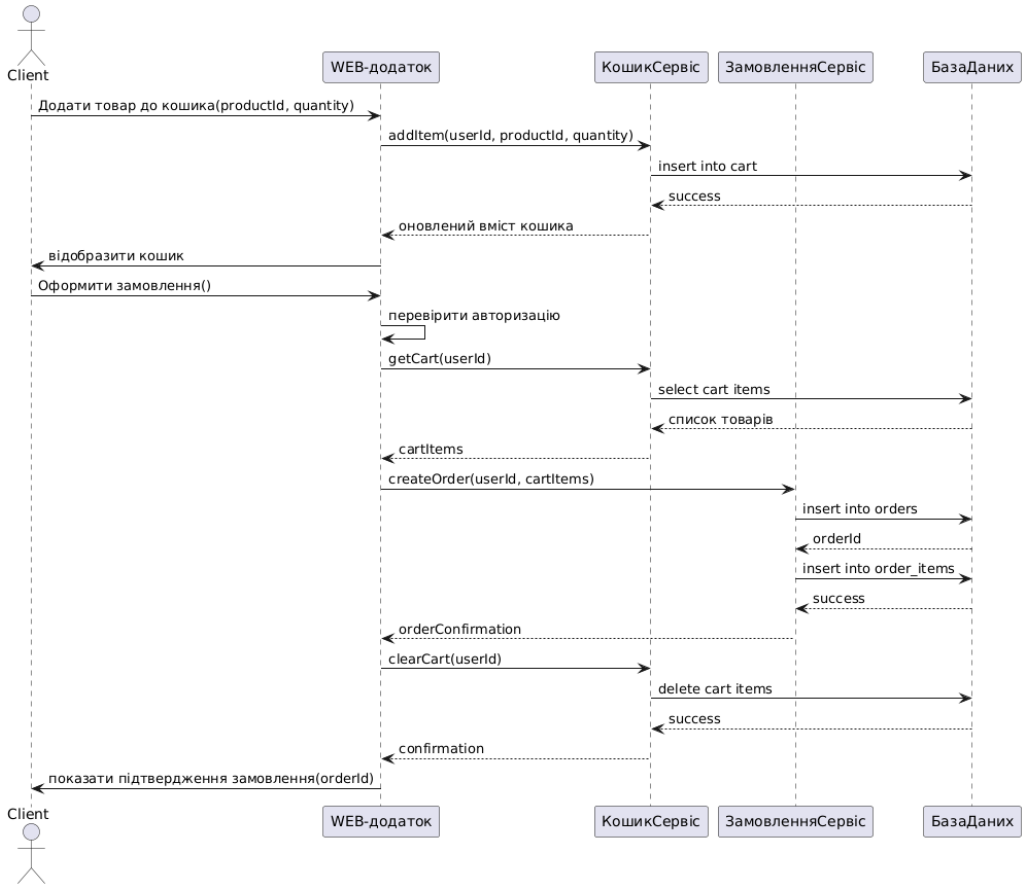


Рисунок В.3 – UML-діаграма послідовності побутового магазину

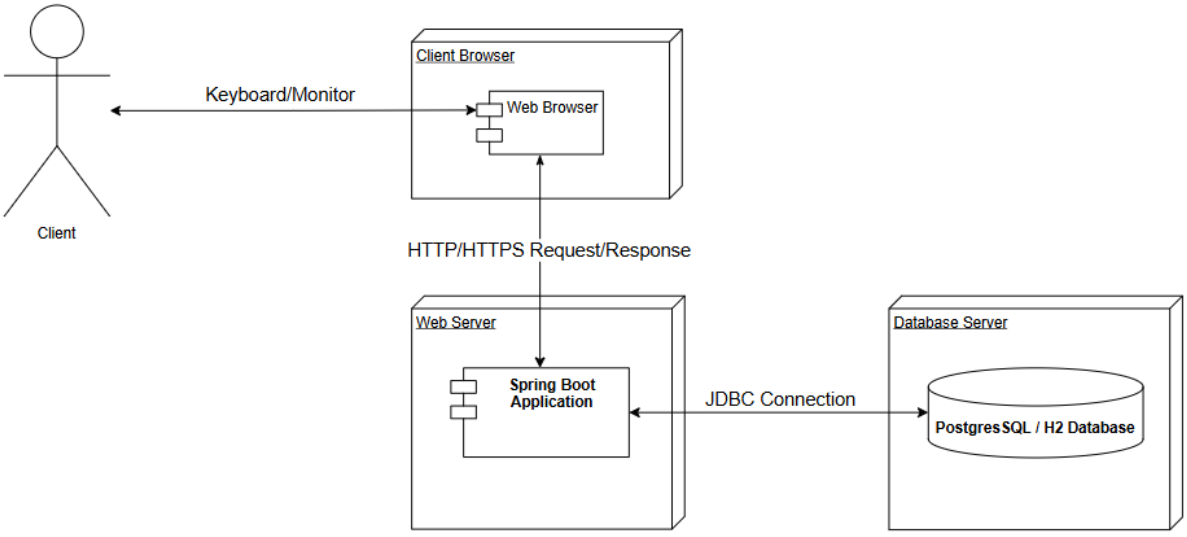


Рисунок В.4 – UML-діаграма розгортання для WEB-додатку «Інтернет-магазин побутових товарів»

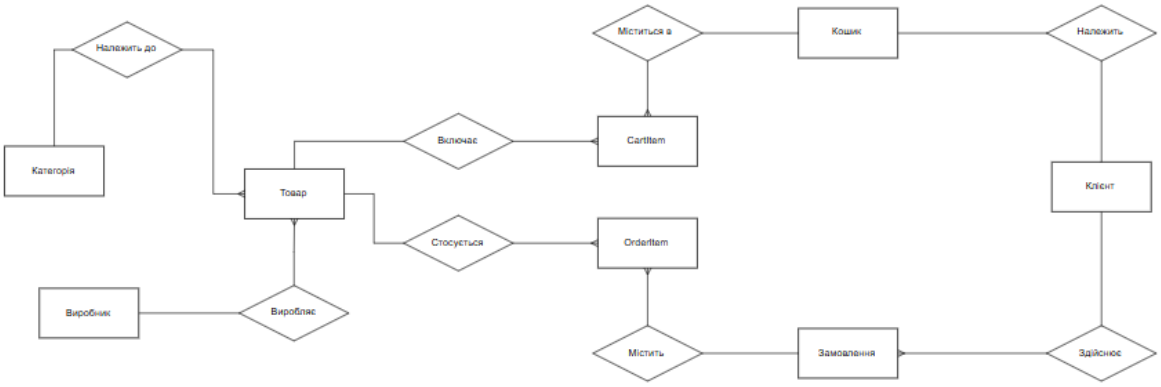


Рисунок В.5 – ER-моделі предметної області WEB-додатку інтернет-магазин побутових товарів

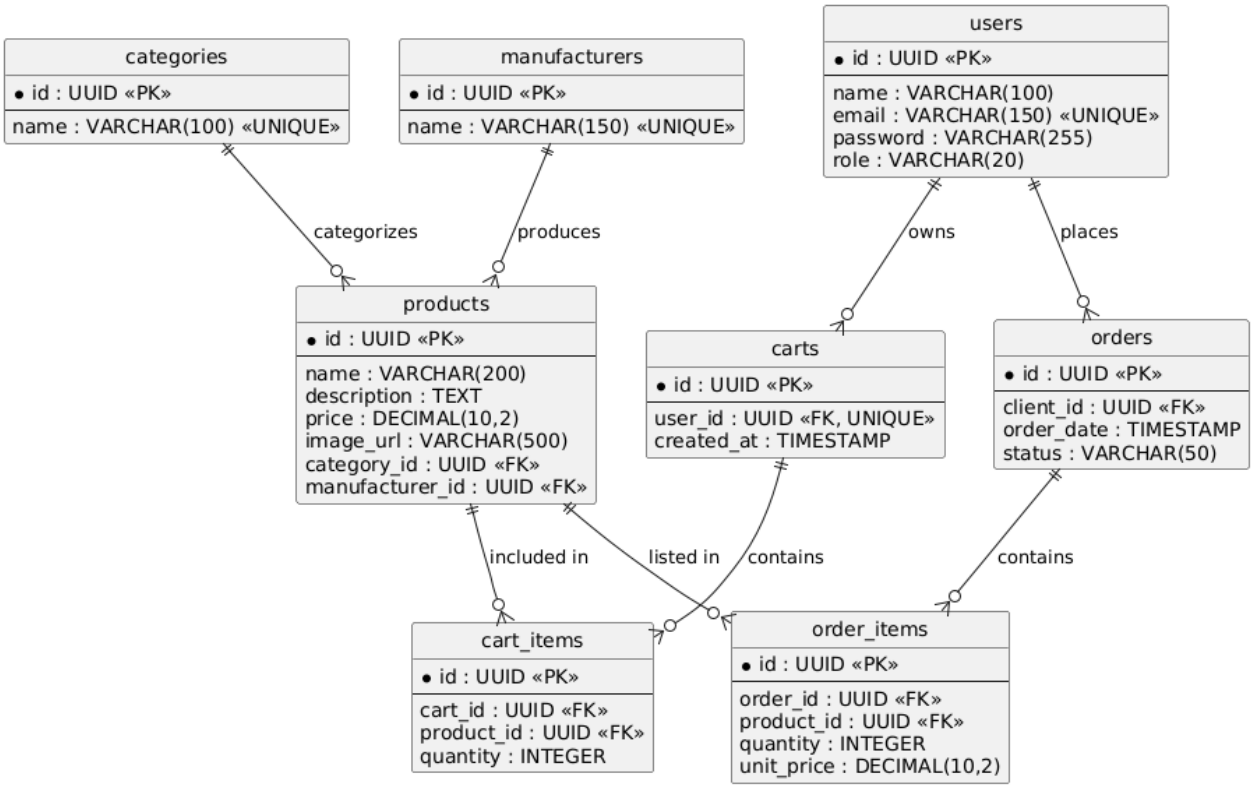


Рисунок В.6 – Схема бази даних WEB-додатку інтернет-магазин побутових товарів

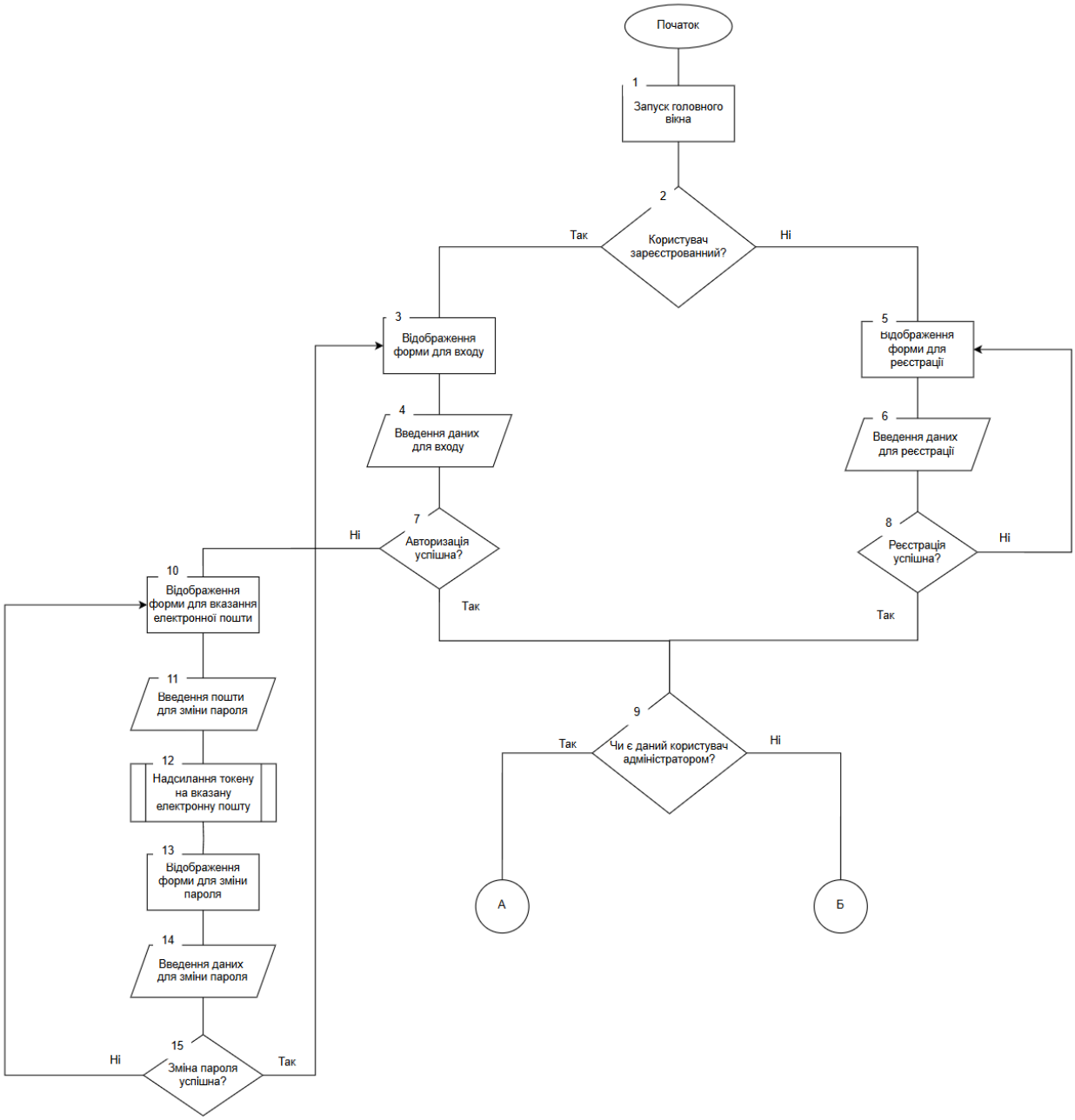


Рисунок В.7 – Схема алгоритму роботи WEB-додатку інтернет магазину побутових товарів

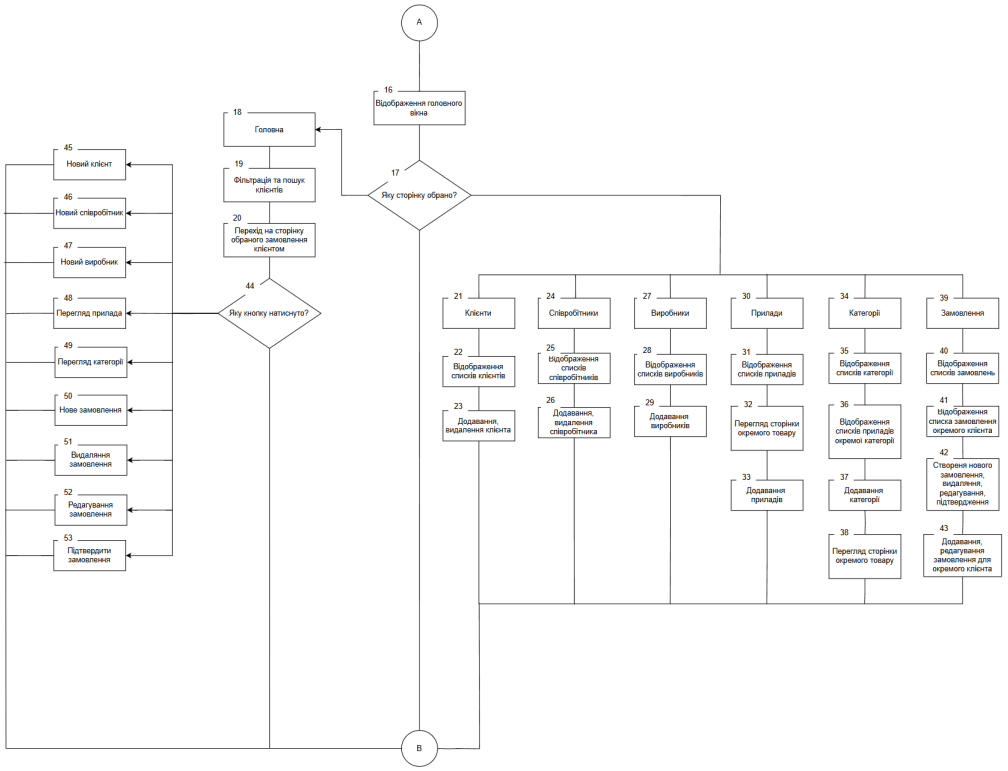


Рисунок В.8 – Продовження

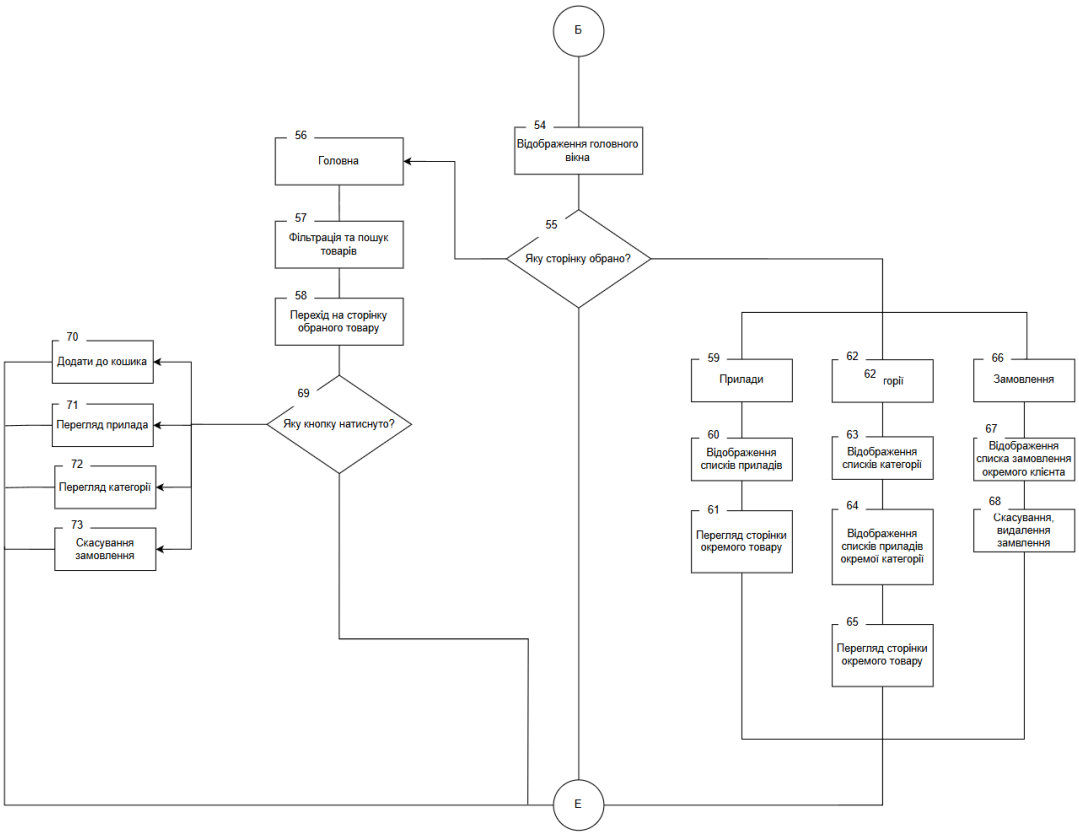


Рисунок В.9 – Продовження

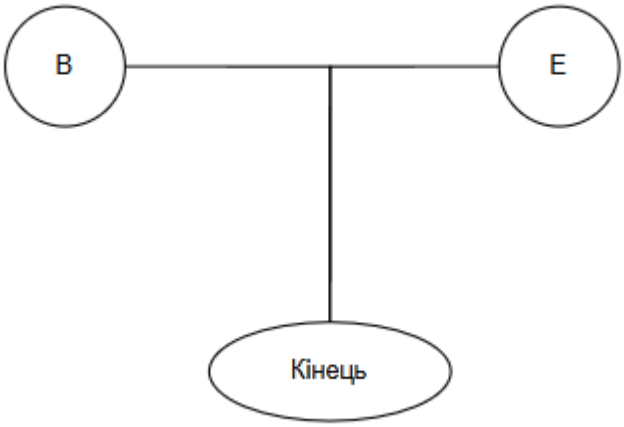


Рисунок В.10 – Продовження

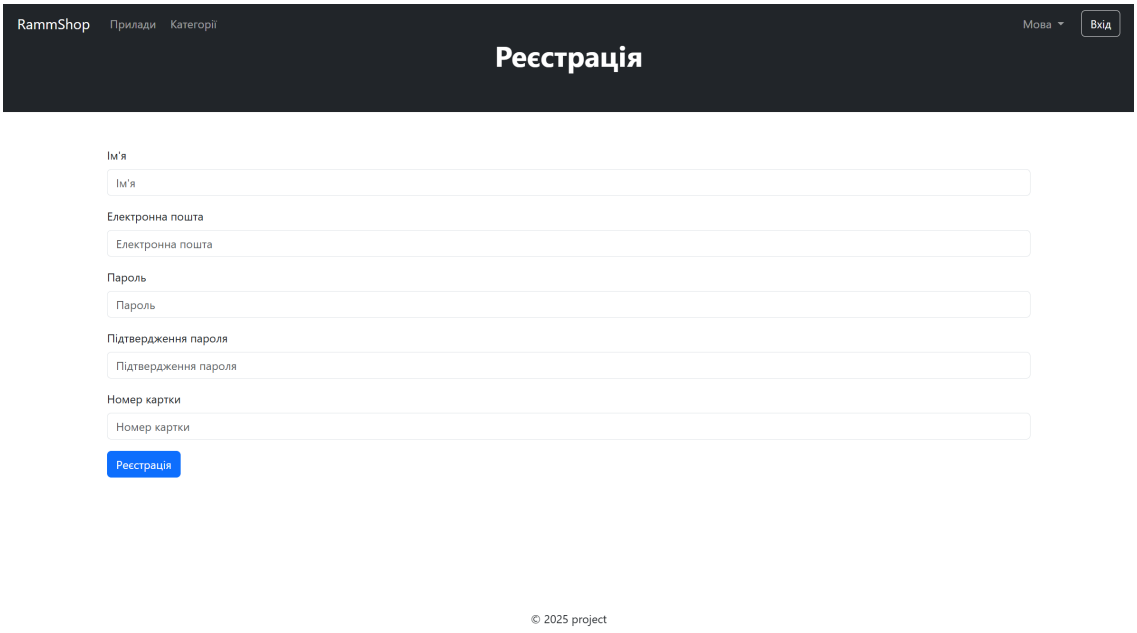


Рисунок В.11 – Сторінка для реєстрації нового користувача

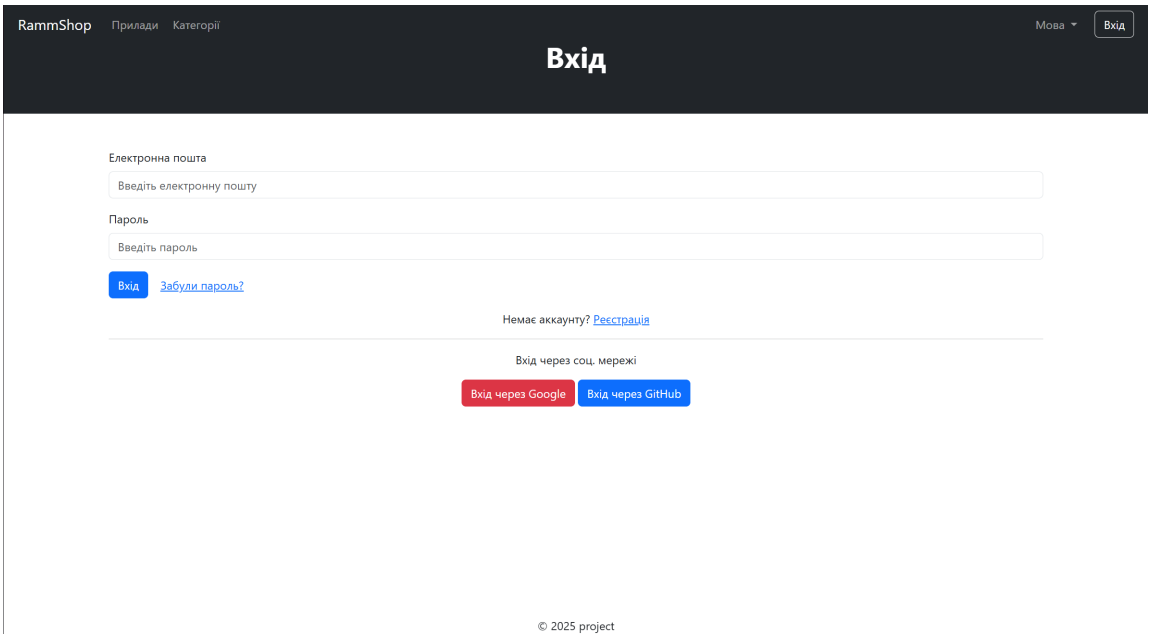


Рисунок В.12 – Сторінка для авторизації

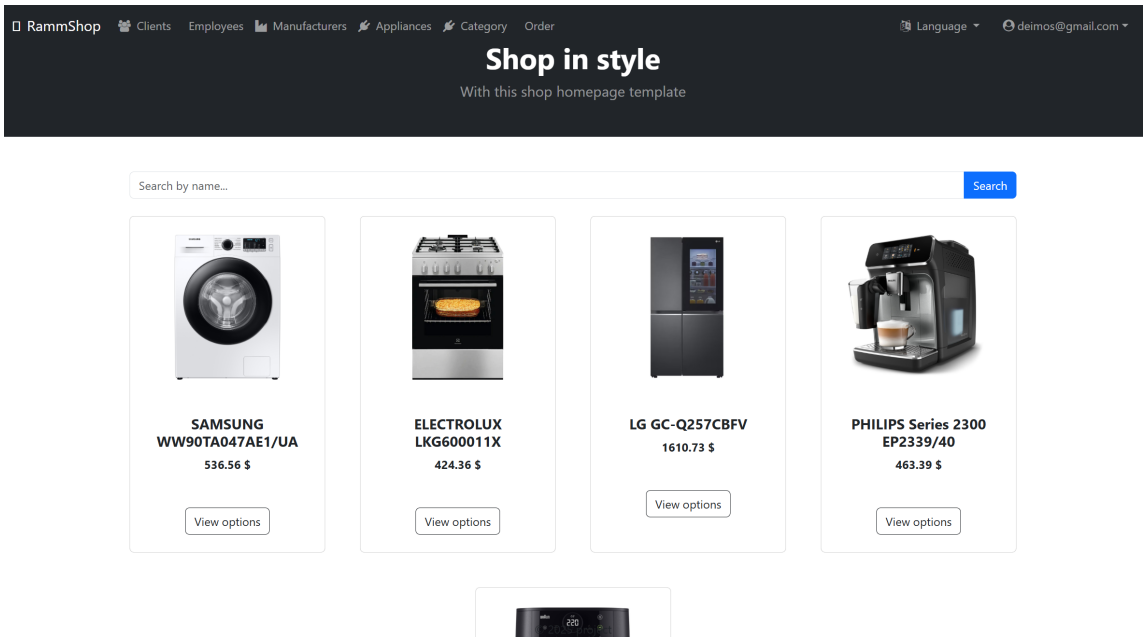


Рисунок В.13 – Сторінка «Прилади»

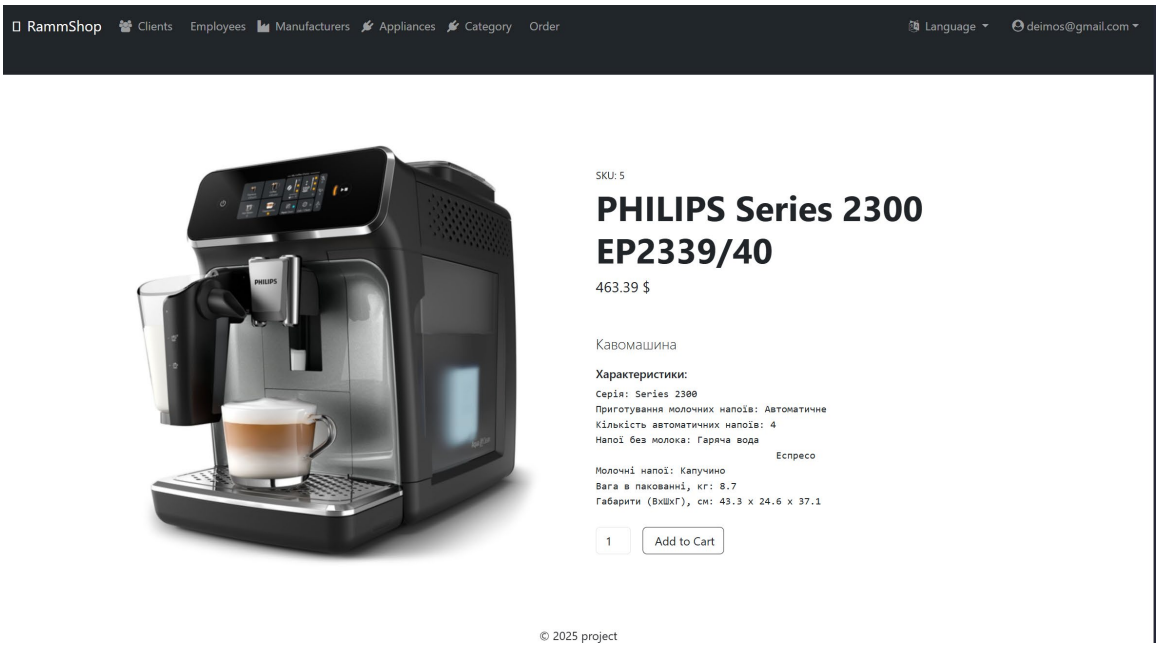


Рисунок В.14 – Сторінка окремого товару

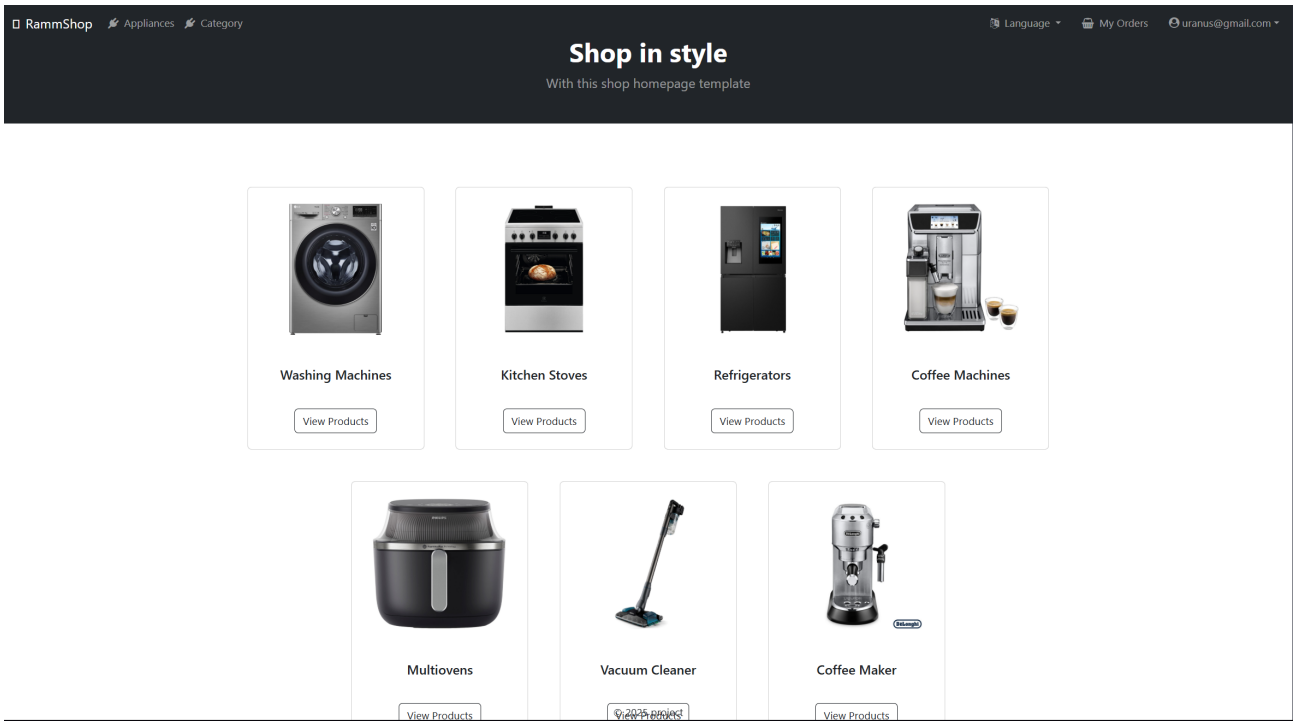


Рисунок В.15 – Сторінка «Категорії»

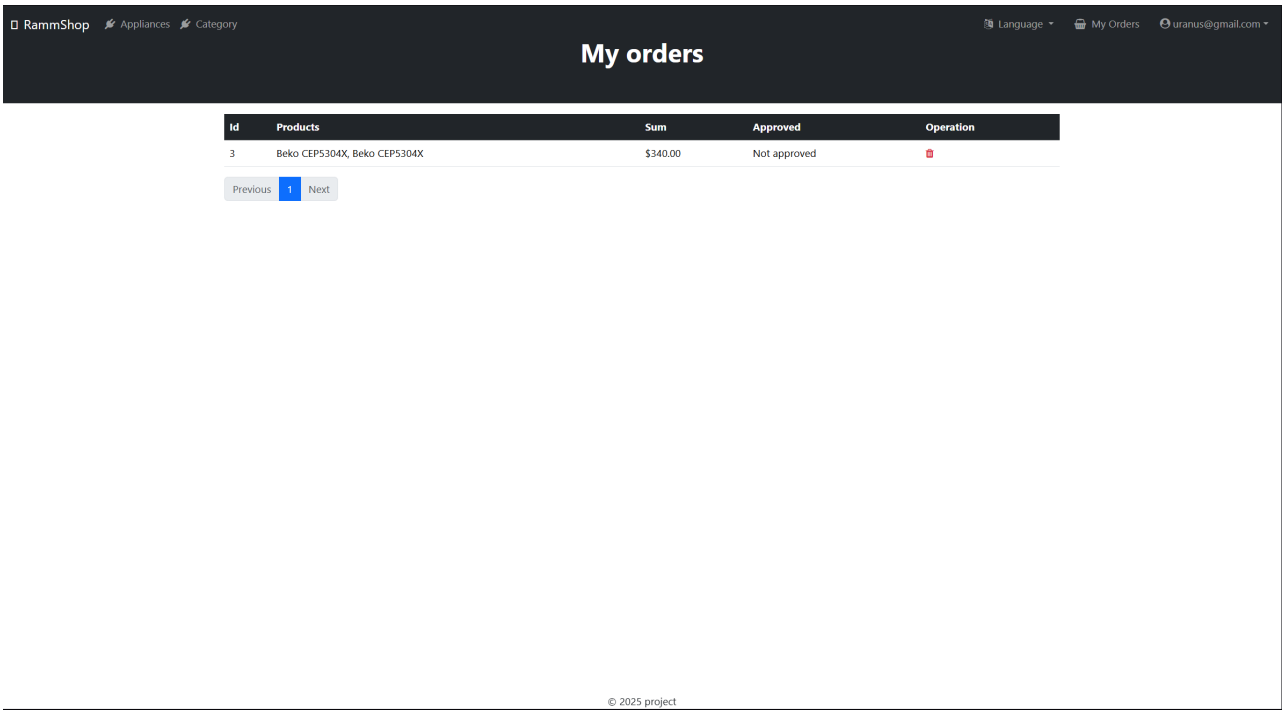


Рисунок В.16 – Сторінка «Мої Замовлення»

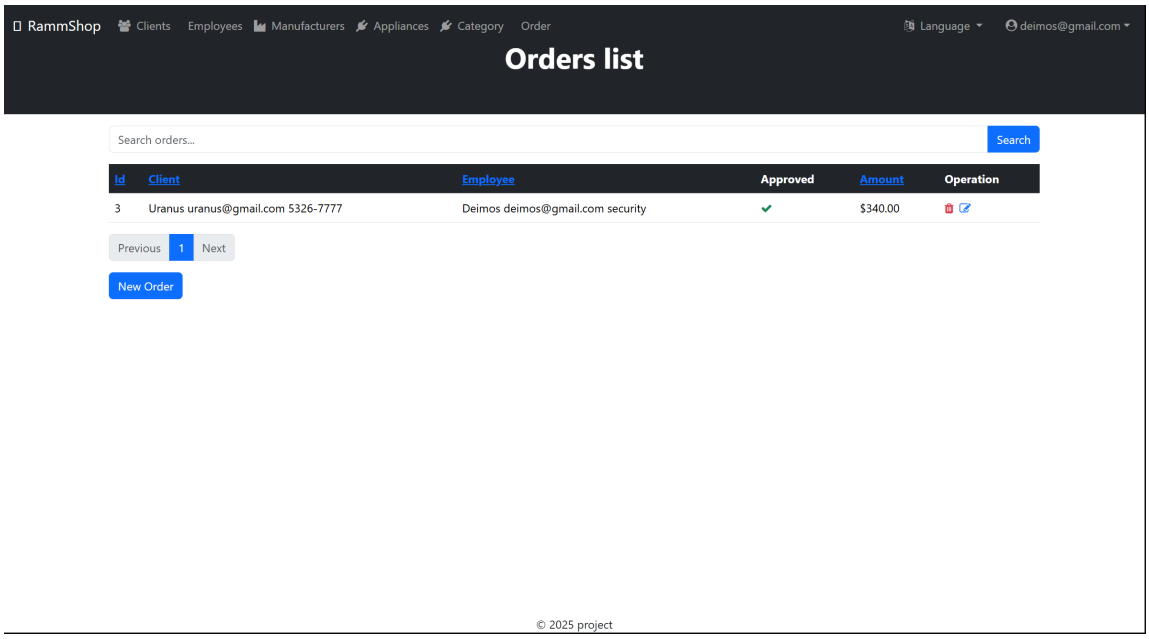


Рисунок В.17 – Сторінка менеджменту замовлення для адміністратора

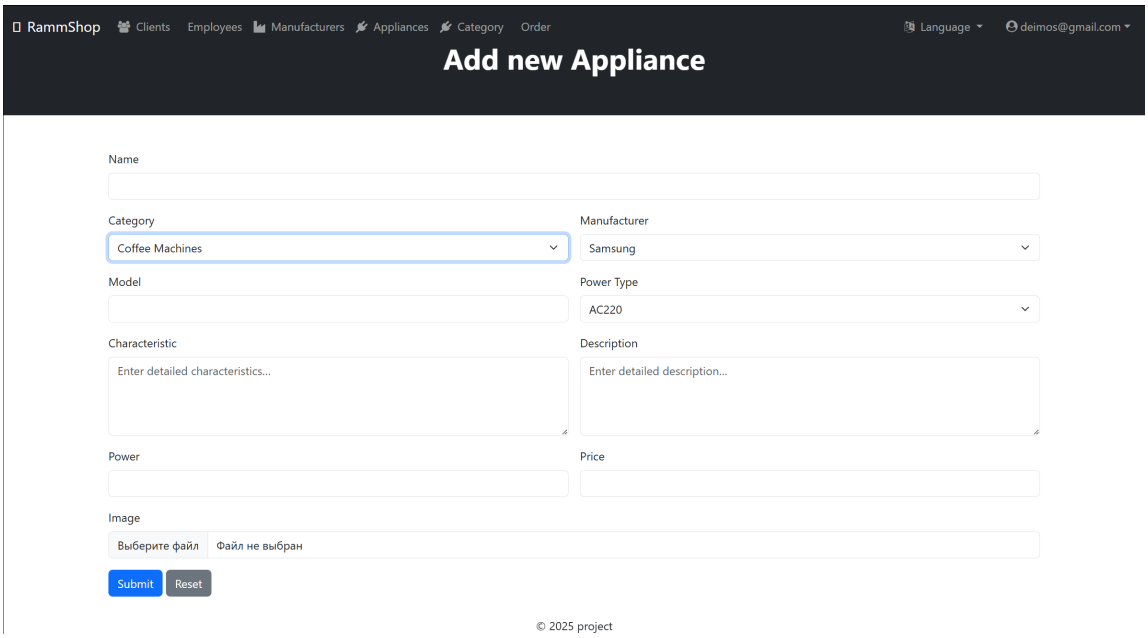


Рисунок В.18 – Приклад додання товару

ДОДАТОК Г (ДОВІДНИКОВИЙ)
ІНСТРУКЦІЯ КОРИСТУВАЧА

Для реєстрації користувача потрібно вказати ім’я, пошту, пароль, підтвердження паролю та прив’язати номер карти. Також для вільного перегляду товарів або категорії необов’язково реєструватися або авторизуватися на сайті, це потрібно тільки для того, щоб здійснити покупку зі сторони клієнта, або для адміністрування зі сторони адміністратора. Сторінка для реєстрації користувача зображено на рисунку Г.1, а для входу на рисунку Г.2.

Рисунок Г.1 – Сторінка для реєстрації нового користувача

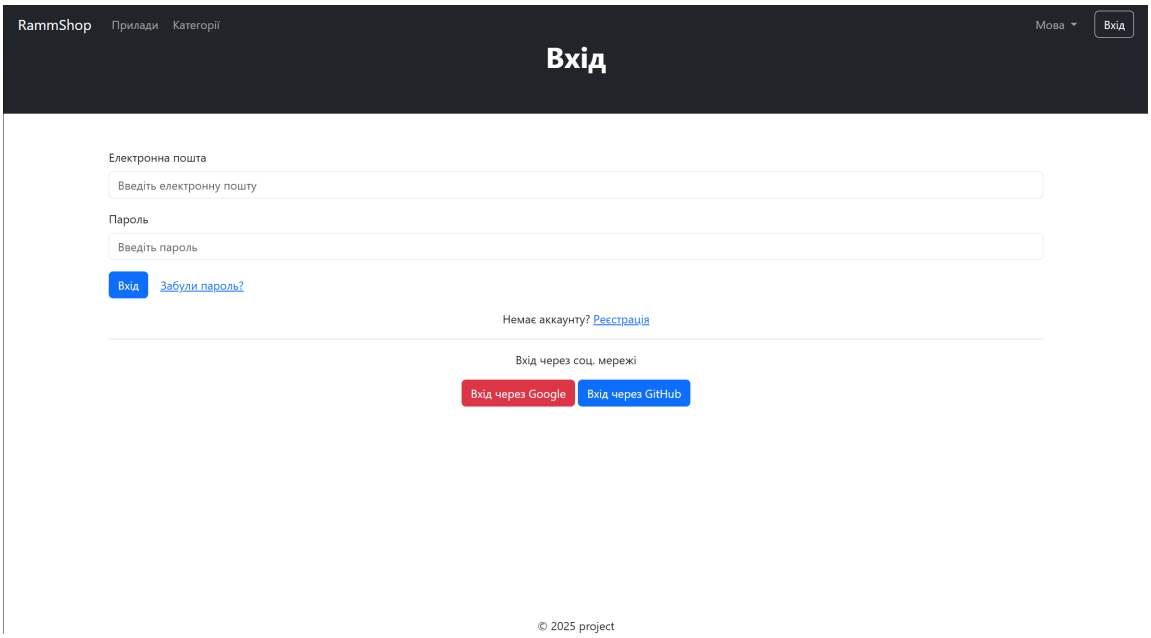


Рисунок Г.2 – Сторінка для авторизації

Після успішної авторизації відображається головна сторінка WEB-додатку, де користувач, в ролі клієнта може переходити на сторінку «Прилади», «Категорії» та «Мої Замовлення». Клієнт може продивлятися товар, шукати товар за категорією, та за назвою, виконувати фільтрацію, здійснювати покупки. Після натискання кнопки «Додати до кошика», обраний товар та обрана кількість товару потрапляє до кошика клієнта, де клієнт може подивитися своє замовлення та видалити його за потреби, та перевірити статус свого замовлення. Сторінки «Прилади», сторінка окремого товару, «Категорії» та «Мої Замовлення» зображенні на рисунках Г.3, Г.4, Г.5 та Г.6.

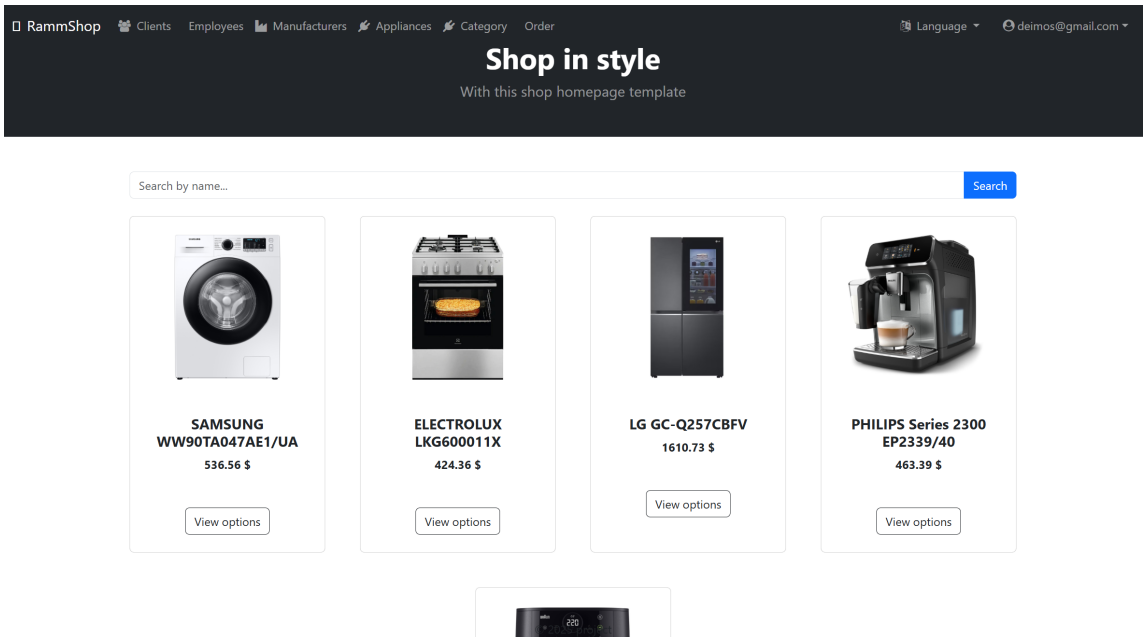


Рисунок Г.3 – Сторінка «Прилади»

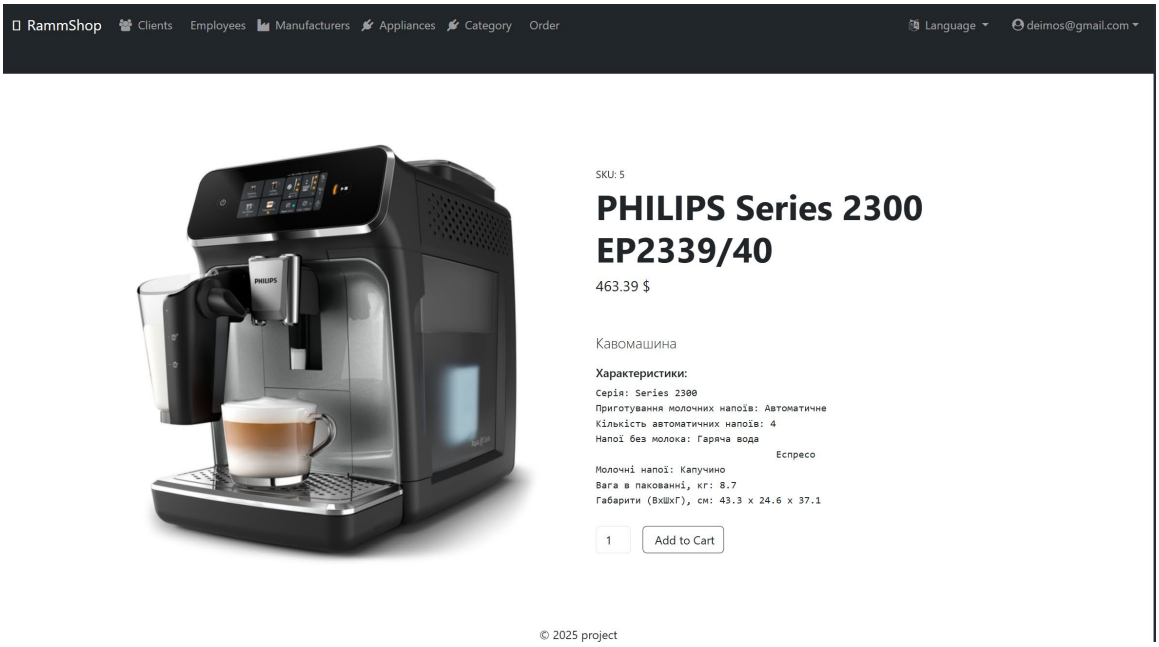


Рисунок Г.4 – Сторінка окремого товару

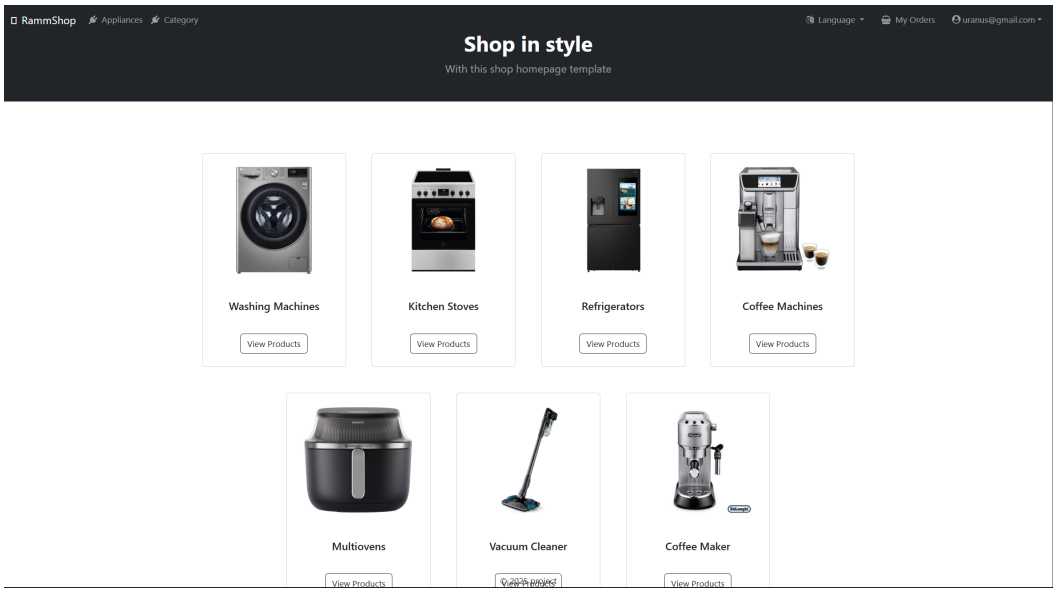


Рисунок Г.5 – Сторінка «Категорії»

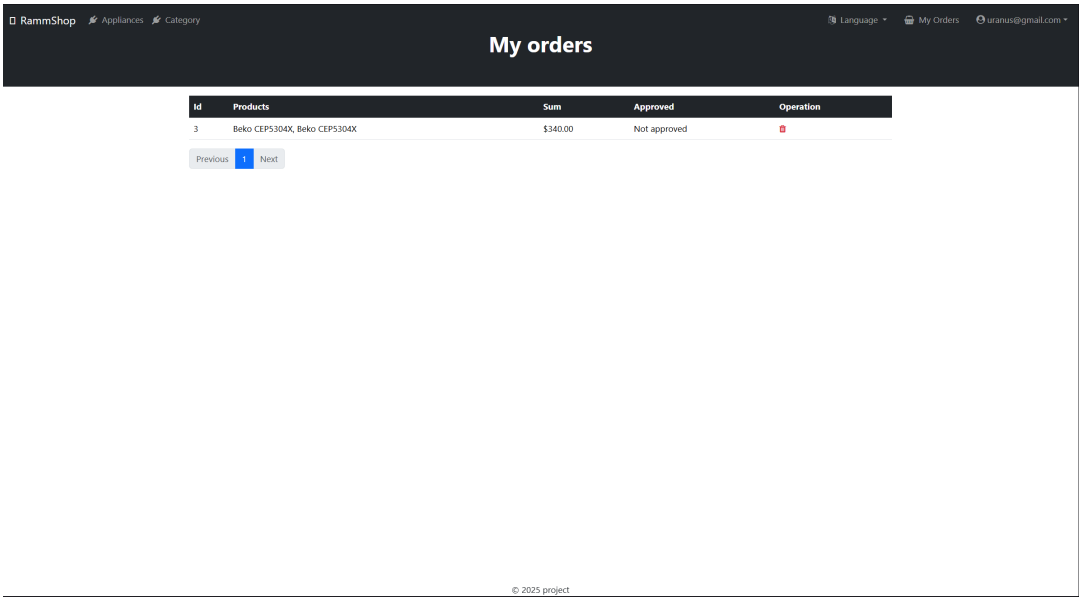


Рисунок Г.6 – Сторінка «Мої Замовлення»

Для адміністратора сайту, звісно доступно набагато більше функціоналу, ніж для клієнта. Адміністратор може повністю керувати замовленням клієнтів, скасовувати, редагувати та підтверджувати. Також найважливішою частиною адміністрування є додавання виробників, категорії, товарів. Адміністратор може додавати робочих(адміністраторів), призначати їм певний відділ, реєструвати нових клієнті за потреби, наприклад, якщо у клієнта немає акаунта і в магазині

йому запропонували зареєструватися на місці. На рисунку Г.7 зображено приклад менеджменту замовлення та Г.8 зображено приклад додання товару.

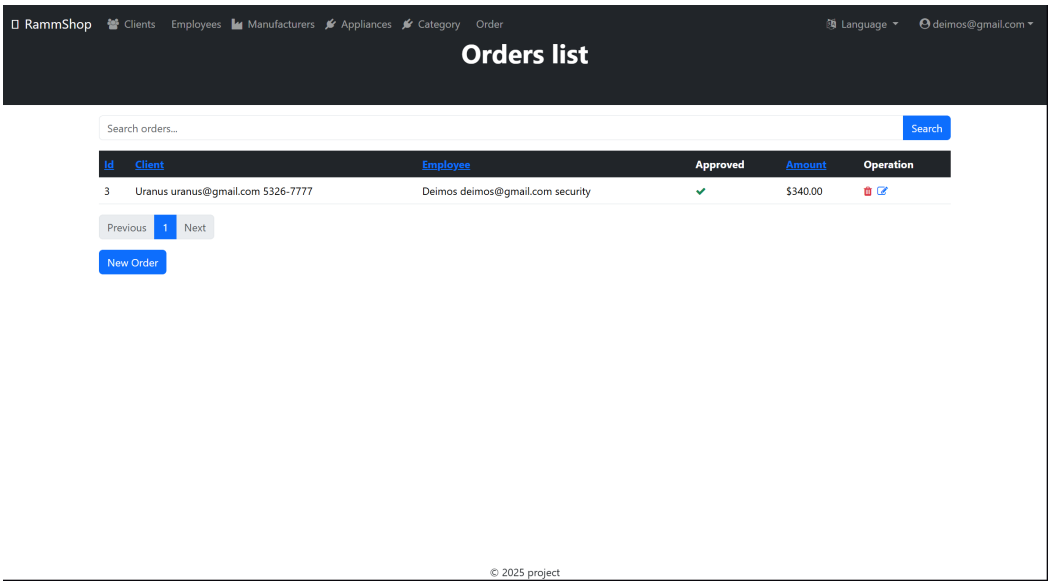


Рисунок Г.7 – Сторінка менеджменту замовлення для адміністратора

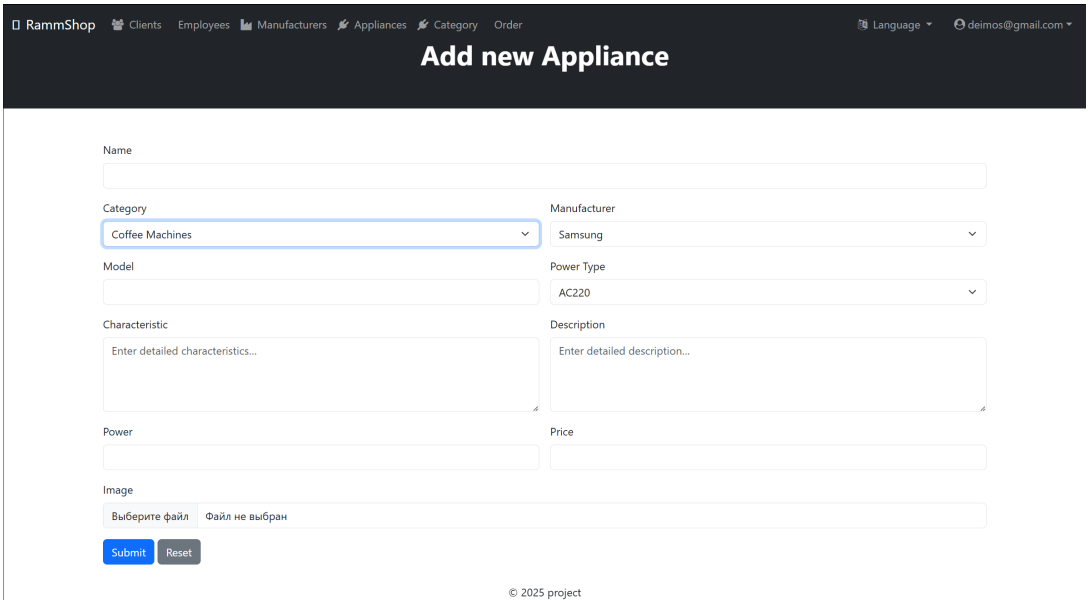
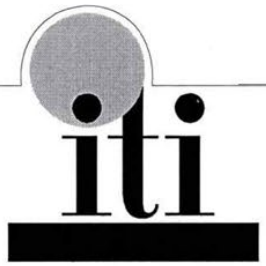


Рисунок Г.8 – Приклад додання товару

ДОДАТОК Д (ДОВІДНИКОВИЙ)
ДОВІДКА ПРО ВПРОВАДЖЕННЯ



МАЛЕ НАУКОВО-ВИРОБНИЧЕ ПІДПРИЄМСТВО "ТОВ "ІТІ"
Україна, 21021, м.Вінниця, вул. Келецька, 56

№ 40 від "6" 06 2025 р.

ДОВІДКА

Дана студентці групи 4КН-21б Ніловій Марії Геннадіївні в тому що результати, одержані нею в процесі виконання бакалаврської дипломної роботи «Розробка WEB-додатку «Інтернет магазин побутових товарів»», а саме: форми, комунікації і налаштування адміністратора, планується використати в розробках ТОВ «ІТІ».

Зам. Директора ТОВ «ІТІ» Бодяк В.Н.