МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки

«До захисту в ЕК»	**«До захисту допущено»**
Директор інституту(декан факультету)	Завідувач кафедри
_____ ___Андрій ФОРСЮК__	_____ _Сергій ГРИБКОВ__
(підпис) (ім'я та прізвище)	(підпис) (ім'я та прізвище)
«_2_» __червня___ 2025р.	«_2_» __червня___ 2025р.

КВАЛІФІКАЦІЙНА РОБОТА
НА ЗДОБУТТЯ ОСВІТНЬОГО СТУПЕНЯ БАКАЛАВРА

зі спеціальності 122 «Комп'ютерні науки»
(код та назва спеціальності)

освітньо-професійної програми __Інформаційні системи та штучний інтелект

на тему: Розроблення інтернет-магазину одягу бренду "Prilipko Design"

Виконав: здобувач _4_ курсу, групи __КН-4-4___

_____Прилипко Максим Сергійович_____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Керівник _____Андріюк Олена Петрівна_____ _____
(прізвище, ім'я, по батькові повністю) (підпис)

Консультанти_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

_____ _____
(ім'я та прізвище) (підпис)

Рецензент _____ _____
(ім'я та прізвище) (підпис)

Я як здобувач(ка) Національного університету харчових технологій розумію і підтримую політику університету з академічної доброчесності. Я не надавав(-ла) і не одержував(-ла) недозволеної допомоги під час підготовки цієї роботи. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач_____
(підпис)

Київ - 2025р.

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ХАРЧОВИХ ТЕХНОЛОГІЙ

Інститут (факультет) Автоматизації і комп'ютерних систем
Кафедра Інформаційних технологій, штучного інтелекту і кібербезпеки
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма Інформаційні системи та штучний інтелект

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інформаційних технологій, штучного інтелекту і кібербезпеки
Сергій ГРИБКОВ

" 28 " квітня 2025 року

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧА
Прилипко Максима Сергійовича
(прізвище, ім'я, по батькові)

1. Тема роботи Розроблення інтернет-магазину одягу бренду "Prilipko Design"
керівник роботи Андріюк Олена Петрівна, доцент кафедри ІТШІК
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від 28_квітня_ 2025 року № _254-кс
2. Строк подання здобувачем роботи 30.05. 2025 р.
3. Вихідні дані до роботи
1. Нормативні документи НУХТ
2. Правила прийому до НУХТ
3. Інструкції щодо роботи приймальної комісії
4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)
   1. Аналіз діяльності компанії *Prilipko Design* та виявлення потреб щодо цифровізації бізнес-процесів.
   2. Формулювання функціональних та нефункціональних вимог до веб-сайту з урахуванням особливостей бренду.
   3. Проєктування архітектури інформаційної системи та обґрунтування вибору технологічного стеку (React, Node.js, MongoDB).
   4. Розробка структури сайту, створення схем навігації та макетів основних сторінок.
   5. Реалізація клієнтської та серверної частин веб-застосунку: каталог товарів, реєстрація, авторизація, кошик, замовлення, профіль користувача, адмінпанель.
   6. Інтеграція з базою даних, реалізація захищених API та налаштування середовища розробки й деплою.
   7. Створення супровідної документації для користувачів і адміністраторів системи.

5. Перелік графічного матеріалу

Функціональні моделі, бази даних, інтерфейс веб-сайту, фрагменти коду програми

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1	Андріюк Олена Петрівна		
2	Андріюк Олена Петрівна		
3	Андріюк Олена Петрівна		

7. Дата видачі завдання_____28 квітня 2025 року_____

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів виконання кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
	Аналіз предметної області	28.04.2025	
	Формулювання технічного завдання	01.05.2025	
	Проектування архітектури та бази даних	07.05.2025	
	Реалізація основного функціоналу	15.05.2025	
	Тестування, відлагодження	18.05.2025	
	Підготовка пояснювальної записки	21.05.2025	
	Попередній захист	25.05.2025	
	Остаточне оформлення, подання роботи	30.05.2025	

Здобувач _____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Кваліфікаційна робота на тему «Розробка веб-сайту для бренду одягу «Prilipko Design» складається з 3 розділів загальним обсягом 71 сторінки, містить 33 рисунків, 18 таблиці, 2 додатків та 37 посилань на використані джерела.

Загалом, праця зосереджена саме на аналізі магазинів одягу в Україні та закордоном, взявши за приклад діяльність компанії Prilipko Design, що займається виробництвом і продажем одягу. Основною метою є розробка веб-сайту для автоматизації замовлень, управління асортиментом продукції та покращення взаємодії з клієнтами. У процесі дослідження проаналізовано рішення, які використовуються компанією зараз, виявлено проблеми в обслуговуванні клієнтів через відсутність онлайн-системи, а також доведено необхідність створення власного інтернет-ресурсу.

У підсумку було розроблено сучасний, адаптивний веб-сайт, з використанням технологій React, Redux, Node.js та MongoDB. Система включає в себе функціонал реєстрації та авторизації користувачів, можливість перегляду товарів, оформлення замовлень, перегляду замовлен у кабінеті користувача, а також адміністративну панель для керування даними клієнтів, товарів, замовлень та відгуків. Також проведено техніко-економічне обґрунтування проекту, яке підтвердило доцільність його реалізації.

Ключові слова: FRONTEND**,** REACT**,** ІНТЕРФЕЙС КОРИСТУВАЧА**,** СИСТЕМА ОФОРМЛЕННЯ ЗАМОВЛЕНЬ**,** АДМІНІСТРАТИВНА ПАНЕЛЬ**,** ВЕБ-РОЗРОБКА**,** ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ**.**

ABSTRACT

The qualification work on “Development of a website for the clothing brand ”Prilipko Design" consists of 3 chapters with a total volume of 71 pages, contains 33 figures, 18 tables, 2 extras and 37 references to the sources used.

In general, the work focuses on the analysis of clothing stores in Ukraine and abroad, taking as an example the activities of Prilipko Design, a company engaged in the production and sale of clothing. The main goal is to develop a website to automate orders, manage product assortment, and improve customer interaction. The study analyzed the solutions currently used by the company, identified problems in customer service due to the lack of an online system, and proved the need to create its own online resource.

As a result, a modern, responsive website was developed using React, Redux, Node.js, and MongoDB technologies. The system includes the functionality of user registration and authorization, the ability to view products, place orders, view orders in the user's account, as well as an administrative panel for managing customer data, products, orders, and reviews. A feasibility study of the project was also conducted, which confirmed the feasibility of its implementation.

Keywords: FRONTEND, REACT, USER INTERFACE, CHECKOUT SYSTEM, ADMINISTRATIVE PANEL, WEB DEVELOPMENT, INFORMATION TECHNOLOGY.

ЗМІСТ

ВСТУП

У сучасних умовах цифровізації все більшої важливості набуває розробка зручних та функціональних веб-рішень, які сприяють покращенню взаємодії між компаніями та їхніми клієнтами. Веб-сайти стають ключовим інструментом представлення бренду, надання сервісів і підтримки користувачів у режимі онлайн.

Для компаній, що прагнуть зміцнити свою присутність у цифровому просторі, важливим етапом розвитку є впровадження сучасних веб-рішень. Створення повноцінного сайту для бренду одягу Prilipko Design дозволяє оптимізувати внутрішні процеси, автоматизувати обробку замовлень, спростити управління асортиментом і забезпечити ефективну взаємодію з клієнтською базою через зручний онлайн-інструмент.

Метою кваліфікаційної роботи є створення повноцінного веб-застосунку для Prilipko Design, який охоплює ключові функціональні модулі: реєстрацію та авторизацію користувачів, каталог продукції з фільтрацією, динамічний кошик, механізм оформлення замовлень, відображення історії покупок, а також адміністративну панель для управління контентом. Для клієнтської частини використано бібліотеку React у поєднанні з Redux, а для серверної — Node.js з базою даних MongoDB.

У межах роботи проведено аналіз предметної області, визначено функціональні та нефункціональні вимоги до системи, розроблено технічне завдання, спроєктовано архітектуру веб-застосунку, реалізовано основний функціонал і виконано перевірку працездатності системи.

Результатом реалізації проєкту є веб-сайт, адаптований до потреб сучасного користувача, що підвищує ефективність обслуговування клієнтів та сприяє розширенню онлайн-присутності бренду.

РОЗДІЛ 1. ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Загальна характеристика Prilipko design

Prilipko Design — український авторський бренд, що спеціалізується на пошитті одягу за індивідуальними замовленнями. Основним принципом компанії є орієнтація та розуміння кожного клієнта, враховуючи його побажання щодо дизайну, розміру, матеріалів і стилю майбутнього неповторного виробу. Такий персоналізований та винятковий підхід дозволяє створювати унікальні речі, які точно відповідають очікуванням замовника.

Окрім роботи під індивідуальні запити, бренд також пропонує колекцію готових виробів (асортимент майбутнього сайту), розроблених у фірмовому стилі. Цей асортимент постійно доповнюється новими моделями, деталями та стилями, що поєднують сучасні тенденції та авторське індивідуальне бачення дизайну.

Усі етапи створення одягу від розробки концепції до фінального пошиття здійснюються в робочому просторі компанії, а саме в офісі. За творчий та виробничий процес відповідає засновниця бренду, дизайнерка Валентина Прилипко, яка особисто працює з клієнтами, розробляє викрійки, підбирає матеріали та втілює задуми в готові вироби. Такий формат дозволяє забезпечити високу якість кожного елементу та повний контроль на всіх етапах створення одягу.

В організаційній структурі компанії задіяні: менеджер, що відповідає за комунікацію з клієнтами, прийом замовлень, організацію записів і супровід процесу замовлення; бухгалтер, що здійснює ведення обліку фінансової діяльності.

Із розширенням кола клієнтів Prilipko Design зростає і потреба в новому рівні взаємодії з клієнтами - швидкому, зручному, але водночас особистому та унікальному підході. Звичайна система комунікації вже не встигає за темпом сучасного світу, а ручне опрацювання замовлень починає сповільнювати

розвиток бренду. Саме тому було схвалено рішення створити веб-сайт, який стане не просто технічним інструментом, а продовженням стилю та ідеї бренду у цифровому просторі інтернет.

1.2 Організаційна структура Prilipko design, роль і взаємодія підрозділів

1.2.1 Загальна схема організаційної структури

Організаційна структура Prilipko Design вирізняється простотою та компактністю, що пояснюється камерним характером діяльності та особистим наглядом з боку засновниці. Компанія працює як невеличкий креативний осередок, де кожен працівник виконує чітко окреслену функцію, а взаємодія між відділами відбувається безпосередньо, без зайвих бюрократичних перепон.

Центральну роль в організаційній структурі відіграє засновниця та головний дизайнер – Валентина Прилипко. Вона відповідає за весь виробничий цикл: від створення ескізів та зняття мірок до підбору тканин та пошиття виробів. Також вона ухвалює ключові рішення щодо стратегії бренду, взаємодії з клієнтами та загального візуального стилю компанії. Допомагає в організації процесів менеджер, який відповідає за прийом замовлень, ведення клієнтських даних, комунікацію через соціальні мережі або месенджери, а також – за логістику: передачу готових замовлень та контроль термінів виконання.

Менеджер є посередником між клієнтом та дизайнером, забезпечуючи чіткість та злагодженість у роботі. Ще одним важливим елементом структури є бухгалтер, який веде фінансову звітність, контролює облік витрат і доходів, здійснює підготовку податкової звітності та забезпечує фінансову стабільність діяльності компанії. Комунікація між усіма членами команди здійснюється напряму, що дає змогу оперативно реаугувати на зміни, швидко приймати рішення та підтримувати гнучкість у роботі.

Така структура є ефективною для малого бізнесу, який орієнтується на якість, швидкість та індивідуальний підхід до клієнта.

1.2.2 Структура підсистеми замовлень та клієнтського обслуговування

У межах компанії Prilipko Design ключовим аспектом, котрий потребує автоматизації, виступає процес прийняття та опрацювання персоналізованих замовлень, а також взаємодія з клієнтурою. З огляду на це, головний напрямок автоматизації — підсистема роботи з замовленнями та клієнтським сервісом, котра в теперішній структурі компанії реалізується менеджером у взаємодії з дизайнером та бухгалтером.

Даний підрозділ виконує функцію мосту між клієнтами та виробничою складовою бренду. Її ключові функції охоплюють отримання запитів на пошиття чи замовлення готових виробів, узгодження деталей, формування замовлення, його передача у виробництво, контроль термінів та підтримка зворотного зв’язку.

На теперішньому етапі менеджер виконує більшість операцій власноруч: спілкується з клієнтами через соцмережі, приймає замовлення у чатах, записує інформацію у нотатниках чи електронних таблицях. Це створює значне навантаження, підвищує ймовірність втрати даних та ускладнює аналіз.

Створення веб-сайту має на меті централізувати та автоматизувати всі ці процеси, покращивши системність та ефективність роботи.

Основними функціями підсистеми веб-сайту є обробка вхідних замовлень, що надходять через інтерфейс сайту, а також збір контактних даних клієнта для подальшого оформлення та обробки замовлень. Після оформлення система забезпечує інформування користувача про поточний статус його замовлення. Крім того, реалізовано облік історії замовлень для зареєстрованих користувачів, що дозволяє переглядати попередні покупки. Додатково передбачено можливість залишення клієнтами відгуків про товари, які після модерації можуть бути опубліковані на сайті.

Знизу наведена таблиця щодо того, яку інформацію вказаний підрозділ отримує, а яку передає іншим.

Таблиця 1.1. Обмін інформацією між підрозділами

Отримує	Передає
Від клієнта: замовлення, контактні дані, побажання, відгуки	Дизайнеру: технічне завдання, мірки, тип виробу
Від дизайнера: підтвердження, вартість, готовність	Клієнту: статус замовлення, підтвердження
Від бухгалтера: підтвердження оплати	Бухгалтеру: сума замовлення, ID клієнта

1.3 Аналіз нинішнього стану комп'ютеризації Prilipko design

На сьогоднішній день у компанії Prilipko Design не впроваджено єдиної інформаційної системи, яка б автоматично забезпечувала підтримку основних бізнес-процесів. Уся діяльність, пов'язана з клієнтською взаємодією, оформленням замовлень, веденням обліку товарів і внутрішнім спілкуванням, здійснюється вручну — переважно через особисті повідомлення в соціальних мережах, месенджерах або шляхом телефонних дзвінків.

Менеджер виконує всі етапи замовлення вручну: записує заявки, передає їх дизайнеру, відстежує виконання та тримає зв'язок із клієнтами. Водночас дані про замовлення, клієнтів і процеси компанії не фіксуються у структурованому вигляді, що призводить до ускладнення пошуку інформації та підвищує ризики помилок.

Жодне спеціалізоване програмне забезпечення для управління обліком чи замовленнями наразі не використовується. Робочі процеси ведуться фрагментарно — у вигляді заміток, таблиць або листування в чатах. Через відсутність централізованої системи обліку неможливо ефективно контролювати наявність продукції, вести історію клієнтів, швидко формувати повторні замовлення чи отримувати аналітичні дані для прийняття рішень.

Таким чином, Prilipko Design наразі не має жодної інтегрованої IT-системи, яка дозволила б поєднати ключові компоненти діяльності —

замовлення, клієнтів, асортимент і звітність — у єдиному цифровому середовищі.

У підсумку можна сказати, що такий рівень комп'ютеризації є недостатнім для сучасного бізнесу, що прагне розвитку та збільшення обсягів продажів. Відсутність автоматизованої системи знижує ефективність роботи, ускладнює управління інформацією, підвищує ризики людських помилок та уповільнює обробку замовлень. У зв'язку з цим виникла потреба у впровадженні веб-сайту з вбудованою інформаційною системою, яка дозволить централізувати всі ключові бізнес-процеси, покращити якість обслуговування клієнтів та забезпечити подальший розвиток бренду.

1.4 Функціональне моделювання та аналіз існуючих бізнес-процесів

1.4.1 Функціональна модель діяльності Prilipko design

Наразі, бізнес-процеси в Prilipko Design здійснюються вручну, без застосування уніфікованої інформаційної системи. Це в першу чергу стосується підсистеми обробки замовлень та взаємодії з клієнтами. Функціональну модель поточного процесу (AS IS) можна представити як просту послідовність дій між клієнтом, менеджером, дизайнером та бухгалтером (наведено на таблиці 2).

Таблиця 1.2. Функціональна модель бізнес-процесу (AS IS)

Етап	**Учасник**	**Дія / Опис процесу**	**Формат виконання**
1	2	3	4
1	Клієнт	Звертається через соцмережі або месенджер	Instagram, Telegram, Viber
2	Менеджер	Отримує запит, уточнює деталі (модель, розмір, терміни, контакти)	Листування, усна розмова
3	Менеджер	Записує дані вручну	Блокнот, Google Таблиця, нотатки

1	2	3	4
4	Менеджер -> Дизайнер	Передає замовлення дизайнеру	Повідомлення або усно
5	Дизайнер	Оцінює замовлення, погоджує, виконує пошив	У майстерні вручну
6	Менеджер	Повідомляє клієнта про готовність виробу	Месенджер, телефон
7	Бухгалтер	Фіксує оплату, готує звітність	Excel, ручні записи
8	Клієнт	Отримує виріб, залишає відгук (не завжди)	Instagram Stories, повідомлення

1.4.2 Виявлені проблеми

Аналіз, який було проведено в розділі 1.4.1 стосовно поточної моделі бізнес-процесів у Prilipko Design показав низку суттєвих недоліків, які впливають на ефективність роботи компанії та ускладнюють її подальший розвиток:

1. Відсутність централізованого зберігання даних.

Вся інформація, що стосується замовлень, клієнтів та деталей пошиву, існує як фрагментовані записи, нотатки чи таблиці. Це робить неможливим швидкий доступ до історії замовлень, аналіз частоти покупок або вподобань клієнтів.

2. Ручне оброблення замовлень.

Менеджер виконує весь процес замовлення власноруч: приймає заявки, заносить дані, передає дизайнеру, контролює стан. Це забирає багато часу, збільшує навантаження та призводить до помилок через людський фактор.

3. Низький рівень автоматизації процесів.

Всі дії виконуються без застосування спеціального програмного забезпечення, що не дозволяє автоматично генерувати звіти, підтвердження, сповіщення або вести облік в режимі реального часу.

4. Відсутність єдиної клієнтської бази.

Дані про клієнтів не структуровані і не зберігаються у формалізованому вигляді, що ускладнює повторну роботу з ними, відстеження замовлень або надання персоналізованого сервісу.

5. Ускладнення при масштабуванні бізнесу.

Існуюча модель ефективна лише за невеликої кількості замовлень. З ростом клієнтів та навантаження система починає працювати неефективно й нестабільно.

6. Обмежені можливості комунікації з клієнтами.

Відсутність веб-платформи ускладнює інформування клієнтів про нові товари, акції або стан замовлення. Уся комунікація залежить від ручної ініціативи менеджера.

7. Відсутність механізму зворотного зв'язку.

Відгуки клієнтів не збираються систематично й не відображаються на загальнодоступних платформах, що знижує рівень довіри до бренду серед потенційних клієнтів.

1.4.3 Пропозиції щодо усунення наявних проблем

Для виправлення виявлених вад у наявній моделі (AS IS) передбачається розробка сучасного веб-сайту. Цей ресурс покликаний не тільки репрезентувати бренд Prilipko Design, але й стати практичним інструментом для автоматизації внутрішніх процесів.

Нова організаційна модель роботи (TO BE), базується на застосуванні інформаційної системи, яка централізує основні бізнес-процеси та забезпечує комфортну взаємодію з клієнтами та між працівниками.

Таблиця 1.3. Опис моделі TO BE (послідовність дій)

Етап	Учасник	Дія / Опис процесу	Формат виконання
1	Клієнт	Заходить на сайт, вибирає товар, оформлює замовлення	Веб-сайт (React)
2	Система	Записує дані в базу, автоматично присвоює замовленню статус	MongoDB
3	Менеджер	Отримує замовлення через адмін-панель, перевіряє деталі	Admin Dashboard
4	Менеджер -> Дизайнер	Передає технічну інформацію, погоджує терміни	Особисто через мессенджери/офлайн
5	Дизайнер	Виконує пошив, змінює статус замовлення на "Готово"	Особисто в офісі
6	Клієнт	Отримує повідомлення про готовність, залишає відгук	Сайт / email / SMS

1.5 Огляд існуючих рішень для розв’язання виявлених проблем

Під час роботи було проаналізувано наявні на українському ринку програмні продукти. Їх було обрано з урахуванням можливості частково або повністю задовольнити потреби компанії Prilipko Design. Тут йдеться про автоматизацію обробки замовлень, управління клієнтською базою та облік товарів. Головною метою цього аналізу стала оцінка: чи варто використовувати готові рішення, або є сенс у розробці власного веб-застосунку.

Нижче наведено перелік наявних програмних продуктів на українському рикну:

1. Prom.ua

Prom.ua — одна з найвідоміших українських платформ для підприємців малого бізнесу, яка дає можливість відкрити інтернет-магазин за допомогою шаблонів. Я роздумував над її застосуванням, оскільки вона надає основні функції для онлайн-торгівлі.

Під час огляду було зафіксовано, що платформа забезпечує облік продукції, процес замовлень та інтеграцію з доставкою. Водночас вона має значні обмеження щодо гнучкості, що не дає змоги втілити у життя специфічні рішення — скажімо, гостьовий чекаут або власну систему управління.

2. Tilda + Ecwid

Протестувавши можливість створення сайту на базі Tilda, інтегруючи її з модулем електронної комерції Ecwid. Це рішення дозволяє швидко зібрати візуально привабливу вітрину з базовим кошиком та формою замовлення.

Проте в процесі аналізу з’ясувалося, що функціональність обмежена, а масштабування і створення повноцінної системи керування користувачами, ролями та товарами потребує складних обхідних рішень або додаткових сервісів.

3. 1С:Підприємство \ BAS

Також було приділено увагу на комплексне рішення BAS Управління торгівлею, котре широко застосовується для автоматизації бізнесу середніх та великих розмірів. Незважаючи на значний функціонал та підтримку всіх необхідних операцій (облік, звітність, аналіз), система виявилася надмірно складною для невеликої творчої майстерні.

Потрібне тривале впровадження, навчання співробітників, а сам інтерфейс виглядає доволі застарілим та незручним для роботи з креативним продуктом.

4. Shopify

Shopify — сучасна платформа для створення онлайн-магазинів, яка має широкий спектр функцій. Було досліджено можливості цієї системи й дійшов висновку, що вона добре підходить для типового e-commerce бізнесу.

Проте високі щомісячні витрати, складність кастомізації, а також відсутність локалізації українською мовою стали ключовими факторами, через які я відмовився від цієї платформи.

Також разом із цим наведно таблицю стосовно порівнняння даних систем.

Таблиця 1.4. Порівняльна таблиця систем

Система	**Вартість**	**Функціональність**	**Гнучкість**	**Доцільність впровадження**
Prom.ua	Середня	Базова для онлайн-магазину	Обмежена	Частково підходить
Tilda + Ecwid	Середня	Вітрина + кошик	Обмежена	Частково підходить
BAS Управління торгівлею	Висока	Повний облік, ERP	Надлишкова	Недоцільно
Shopify	Висока	Інтернет-магазин, розширення	Обмежена	Частково підходить
Власна розробка (React + Node.js)	Висока (разово)	Повна відповідність бізнес-потребам	**Максимальна**	Найкраще рішення

1.6 Розрахунок техніко-економічного обґрунтування впровадження створюваного програмного забезпечення

У межах кваліфікаційної роботи було здійснено аналіз актуальних затрат часу й ресурсів на ручне опрацювання замовлень у компанії Prilipko Design. Запропоновано впровадження веб-сайту з інформаційною системою для автоматизації процесів. Цей крок дає можливість не тільки покращити

результативність праці, а й оптимізувати видатки на комунікацію, мінімізувати кількість помилок, плюс прискорити обробку кожного замовлення.

Проект передбачає одноразові інвестиції у розробку та запуск веб-сайту, а також мінімальні супутні видатки на його підтримку (наведено в таблиці 1.5)

Таблиця 1.5. Актуальні ціни

№	Витрати	Вартість (грн)	Частка в загальній структурі (%)
1	Аналіз і проєктування (ТЗ, прототипи)	3 000	10%
2	Розробка фронтенду (React, верстка)	7 000	23.3%
3	Розробка бекенду (Node.js, MongoDB)	7 000	23.3%
4	Реалізація системи авторизації, безпека	3 000	10%
5	Тестування та усунення помилок	2 000	6.7%
6	Дизайн інтерфейсу, адаптивність	3 000	10%
7	Розміщення на хостингу (1 рік)	1 000	3.3%
8	Домени, SSL-сертифікат	1 000	3.3%
9	Резерв на супровід та техпідтримку	3 000	10%
	Загальна сума	**30 000**	**100%**

Проведений аналіз доводить економічну вигідність розробки власного веб-сайту для Prilipko Design. Капіталовкладення у створення онлайн-ресурсу є разовими, а поточні витрати на його підтримку значно менші, ніж щомісячні внески за використання сторонніх платформ.

Створення власного сайту сприяє оптимізації основних бізнес-операцій, полегшує роботу персоналу, збільшує швидкість і якість взаємодії з клієнтами, формує структуровану клієнтську базу, а також створює передумови для розширення бізнесу та збільшення прибутків у майбутньому.

1.7 Обґрунтування доцільності проєктування й розроблення Prilipko design

На основі проведеного аналізу поточного стану роботи компанії Prilipko Design було виявлено низку суттєвих проблем, що пов’язані з відсутністю єдиної інформаційної системи, яка б забезпечувала централізоване ведення обліку замовлень, клієнтів, продукції та внутрішньої взаємодії між працівниками. Уся діяльність на даний момент виконується вручну — через особисті повідомлення в месенджерах, хаотичні записи, електронні таблиці або усне спілкування, що значно ускладнює роботу та уповільнює процес обслуговування клієнтів

У ході побудови функціональної моделі існуючих процесів (AS IS) я визначив основні проблеми: ручне опрацювання замовлень, відсутність централізованої клієнтської бази, неможливість збору й аналізу відгуків, а також труднощі з контролем термінів виконання.

Провівши порівняльний огляд актуальних рішень на ринку, я було дійдено до висновку, що жодна з готових платформ не відповідає повною мірою потребам компанії. Деякі системи мають занадто спрощений функціонал, інші не дозволяють адаптуватися під специфіку невеликого індивідуального бізнесу з креативним підходом до клієнта

Зважаючи на це, було прийнято рішення про необхідність створення власного веб-сайту з інтегрованою інформаційною системою, яка максимально враховуватиме особливості роботи Prilipko Design. Передбачена система дозволить автоматизувати ключові процеси — від оформлення та обробки замовлень до зберігання інформації про клієнтів і товарний асортимент. Вона включатиме зручний інтерфейс для адміністратора й працівників, а також ефективний механізм комунікації з користувачами сайту

Впровадження такої розробки на базі сучасних вебтехнологій створить надійну основу для розвитку компанії, підвищення продуктивності, розширення клієнтської аудиторії та посилення онлайн-присутності бренду. У результаті проєктування веб-сайту є обґрунтованим і стратегічно важливим кроком, який

дозволить Prilipko Design перейти на якісно новий рівень роботи, впровадити ефективну цифрову інфраструктуру та розширити потенціал бізнесу.

РОЗДІЛ 2. ТЕХНІЧНЕ ЗАВДАННЯ НА ПРОЄКТУВАННЯ

2.1 Загальні положення

2.1.1. Найменування системи: Розроблення веб-сайту з продажу одягу бренду Prilipko Design.

2.1.2 Результати робіт зі створення веб-сайту оформлюються відповідно до вимог ДСТУ на відповідні етапи життєвого циклу проекту. Комплексна документація включає текстові матеріали дипломної роботи, супровідну проєктну документацію, а також програмну реалізацію веб-системи згідно з технічним завданням.

2.1.3 За потреби в процесі подальшої розробки окремі пункти цього технічного завдання можуть бути доповнені або уточнені відповідно до нових вимог чи змін у проєкті.

2.2 Призначення і цілі створення системи

2.2.1 Призначення веб-сайту.

Система призначена для автоматизації роботи компанії Prilipko Design шляхом створення повно-функціонального веб-сайту. Розроблений сайт має виконувати роль інформаційної системи, яка охоплює процеси прийому та обробки замовлень, ведення клієнтської бази, управління асортиментом товарів і збору відгуків. Веб-застосунок забезпечує зручну взаємодію між клієнтом, менеджером і дизайнером, а також містить гнучкий інтерфейс адміністрування, що дозволяє швидко оновлювати/додавати/редагувати дані, керувати переліком замовлень, користувачами, обліком товарів та відгуками клієнтів.

2.2.2 Цілі створення системи

Головною метою створення веб-сайту є впровадження сучасного цифрового інструменту, який дозволить налагодити ефективну, зручну та структуровану взаємодію між компанією Prilipko Design і її клієнтами. Реалізація даної системи забезпечить оперативний доступ до інформації про замовлення, клієнтів, товарний асортимент та відгуки клієнтів, а також дозволить автоматизувати процеси обліку та внутрішньої організації роботи.

Впровадження сайту сприятиме підвищенню точності обробки даних, покращенню зручності для клієнтів і створить основу для масштабування та подальшого розвитку компанії.

2.3 Характеристика об’єкта автоматизації.

2.3.1 Короткі відомості про об’єкт автоматизації.

Об’єктом автоматизації є діяльність компанії **Prilipko Design**, що спеціалізується на індивідуальному пошитті одягу.

2.4 Вимоги до системи

2.4.1 Вимоги до системи в цілому.

4.1.1 Вимоги до структури і функціонування системи.

4.1.1.1 Розроблювана система повинна працювати за принципом клієнт-серверної архітектури, де користувач взаємодіє з сайтом через звичайний веб-браузер. Сайт обмінюється даними з сервером за допомогою спеціальних запитів, а сервер, у свою чергу, працює з базою даних. Уся інформація про замовлення, клієнтів, товари та відгуки зберігається в єдиній базі даних MongoDB. Система складається з кількох основних компонентів: веб-інтерфейсу на React, серверної частини на Node.js з Express, бази даних MongoDB для збереження інформації про товари, клієнтів і замовлення, а також адміністративної панелі для керування контентом і обробки замовлень.

4.1.1.2 Сайт повинен повідомляти користувача про помилки під час роботи. Наприклад, якщо щось пішло не так при відправленні форми, втрачено з’єднання з сервером або введені дані були некоректні. Повідомлення мають бути зрозумілими та відображатися у відповідних місцях інтерфейсу залежно від того, хто зараз користується сайтом — клієнт чи працівник.

Усі частини сайту (реєстрація, замовлення, перегляд товарів, залишення відгуків тощо) повинні бути пов’язані між собою через спільну базу даних. Це дає змогу зберігати інформацію послідовно, без помилок і дублювань. Дані обробляються на сервері, а сайт показує оновлення через запити, які виконує автоматично.

4.1.1.3 Програмна частина сайту має бути такою, щоб у майбутньому її можна було доповнювати та вдосконалювати. Наприклад, можна буде додати нові функції — онлайн-оплату, підтримку інших мов або підключення до зовнішніх сервісів. Структура системи повинна дозволяти збільшення кількості товарів, клієнтів і замовлень без зниження швидкості роботи. Зміни в процесах компанії не повинні вимагати повної переробки сайту — тільки точкових оновлень.

4.1.1.4 Усі зміни (наприклад, нове замовлення чи оновлення статусу) мають з’являтися максимально швидко, без затримок.

4.1.2 Вимоги до чисельності і кваліфікації персоналу.

4.1.2.1 Перед початком роботи з системою персонал має бути ознайомлений з інструкцією користувача, правилами введення даних, збереження інформації та створення резервних копій. Необхідно дотримуватись загальних вимог техніки безпеки при роботі з ПК.

4.1.2.2 Система передбачає декілька рівнів доступу: адміністратор, клієнт. Вхід здійснюється за логіном і паролем. Адміністратор має повні права доступу до бази даних, клієнт — вхід в затосунок, змогу залишити відгук, перегляд товарів і оформлення замовлення.

4.1.3 Показники призначення.

4.1.3.1 Сайт повинен забезпечити надійне функціонування основних бізнес-процесів: прийом замовлень, обробка клієнтських даних, управління товарами та комунікація з клієнтами. Ступінь автоматизації має дозволити зменшити ручну роботу і підвищити точність ведення обліку.

4.1.3.2 Система має підтримувати налаштування під потреби компанії: розширення каталогу товарів, зміну форм замовлення, адаптацію до оновлення технічного середовища або інтерфейсу.

4.1.4 Вимоги до надійності.

4.1.4.1 Сайт є багатофункціональним та призначений для постійного використання протягом усього робочого дня. Всі дії користувачів виконуються окремо, послідовно — як наприклад, оформлення замовлення, реєстрація або

оновлення товару. Надійність роботи залежить від стабільності вебсерверу, коректної роботи бази даних MongoDB та якості хостинг-платформи, на якій буде розміщено сайт.

4.1.4.2 Програмна частина сайту повинна забезпечувати:

- можливість виконання всіх функцій (перегляд, замовлення, адміністрування) з будь-якого пристрою через інтернет
- стабільну роботу на різних браузерах і пристроях
- використання модульного, структурованого та об'єктно-орієнтованого підходу при розробці
- перевірку введених даних із повідомленням про помилки (наприклад, неправильний email чи порожнє поле)
- наявність механізмів для виявлення та виправлення можливих помилок у базі даних
- захист від несанкціонованого доступу, збереження даних користувачів, обмеження доступу за ролями
- регулярне створення резервних копій бази даних

4.1.5 Вимоги до безпеки.

Щоб забезпечити безпечне використання системи, необхідно дотримуватись чинних державних стандартів щодо охорони праці, захисту інформації та пожежної безпеки, зокрема вимог ДСТУ 2293-99, ДСТУ ISO 6309:2007, ДСТУ 12.0.230:2008, ДСТУ 7237:2011, ДСТУ 7238:2011, ДСТУ 7239:2011

4.1.6 Вимоги з ергономіки та технічної естетики.

Загальні ергономічні і естетичні вимоги до системи повинні відповідати держстандартам ДСТУ 8604:2015, ДСТУ 7298:2013.

4.1.7 Вимоги по експлуатації, технічного обслуговування, ремонту і зберігання компонентів системи.

4.1.7.1 Обслуговування сайту має проводитися згідно з чинними стандартами щодо надійності й підтримки технічних систем (зокрема, ДСТУ EN 13306:2019 та ДСТУ 3576-97). Сюди входить регулярне оновлення програмного забезпечення, резервне копіювання даних, перевірка безпеки й працездатності сайту.

4.1.7.2 Фізичне розміщення серверів або використання хостинг-платформи повинно відповідати загальним технічним нормам. У разі використання власного обладнання — умови експлуатації мають відповідати технічній документації та вимогам щодо безпечної подачі електроенергії. При використанні хмарного або VPS-хостингу відповідальність за інфраструктуру несе провайдер.

4.1.7.3 Обслуговуванням сайту повинен займатись кваліфікований спеціаліст або розробник, який має доступ до серверної частини, бази даних і панелі керування хостингом. Частота перевірок, резервного копіювання та оновлень визначається графіком технічної підтримки.

4.1.7.4 Всі компоненти системи (вихідний код, база даних, резервні копії, інструкції) повинні зберігатись у безпечному місці — на захищених хмарних сховищах або локальних серверах із контрольованим доступом.

4.1.7.5 Система повинна мати чіткий регламент підтримки. У разі виникнення збоїв має бути можливість відновлення роботи з резервної копії, а в разі тимчасових технічних проблем — запуск сайту в обмеженому режимі з базовим функціоналом (наприклад, перегляд товарів без можливості замовлення).

4.1.8 Вимоги до захисту інформації від несанкціонованого доступу.

Для безпечного збереження даних та захисту від несанкціонованого доступу в системі повинні бути реалізовані такі заходи:

1. Авторизація через пароль — кожен користувач входить до системи за допомогою логіна і пароля. Адміністратор має повний доступ до функцій, клієнт — лише до перегляду товарів та оформлення замовлень.

2. Розмежування доступу — система повинна підтримувати поділ прав доступу за ролями. Користувачі не повинні мати можливості змінювати або видаляти інформацію, яка їм не належить.

3. Захист даних на сервері — використання перевірених засобів безпеки на сервері (наприклад, HTTPS-протокол, обмеження доступу до бази, автоматичне завершення сесій). Сервер має бути налаштований відповідно до рекомендацій щодо безпеки (наприклад, Node.js + Firewall + TLS).

4. Захист бази даних — налаштування ролей у MongoDB, обмеження доступу до колекцій, захист чутливої інформації (наприклад, хешування паролів), створення резервних копій.

5. Регулярне оновлення паролів — можливість змінити пароль користувача вручну або через запит «Забули пароль», із підтвердженням через електронну пошту.

6. Заборона на зміну критичних даних — окремі таблиці (наприклад, історія замовлень або список клієнтів) мають бути захищені від видалення або зміни зі сторони клієнта.

4.1.9 Вимоги щодо збереження інформації при аваріях.

4.1.9.1 Система повинна забезпечувати регулярне створення резервних копій бази даних після змін, з можливістю відновлення даних у разі втрати чи пошкодження.

4.1.9.2 Резервні копії слід зберігати на окремих носіях або в іншому хмарному сховищі, відокремленому від основної бази.

4.1.10 Вимоги по захисту від впливу зовнішніх діянь.

4.1.10.1 У разі використання власного обладнання потрібно забезпечити захист техніки від впливу електромагнітних завад, перегріву та нестабільної напруги, встановивши відповідні фільтри, екрани та стабілізатори.

4.1.10.2 Робоче середовище або серверне приміщення повинні відповідати чинним будівельним і технічним нормам (ДБН, ДСТУ), щоб забезпечити безперебійну роботу обладнання.

4.1.11 Вимоги до патентної чистоти.

При створенні цієї системи патентні дослідження не проводяться.

4.1.12 Вимоги по стандартизації і уніфікації.

При зберіганні та обробці даних у системі слід використовувати загальноприйняті формати запису, а також єдину систему найменувань, кодів і класифікаторів. Формати дати, часу, електронної пошти, номерів телефонів, цін тощо повинні відповідати стандартам та бути однаковими на всіх сторінках сайту.

2.4.2 Вимоги до функцій.

Система повинна реалізовувати всі основні функції, необхідні для ефективної роботи з клієнтами та замовленнями. Особливу увагу приділено функціям адміністрування: керуванню товарним каталогом, базою клієнтів та обробкою замовлень, що забезпечує повноцінну підтримку бізнес-процесів компанії Prilipko Design. (наведено в таблиці 2.4.2.1)

Таблиця 2.4.2.1. Перелік функцій, вхідної та вихідної інформації

№	Найменування функції	Вхідна інформація	Вихідна інформація
1	2	3	4
1	Реєстрація клієнта	Ім’я, прізвище, email, номер телефону, пароль	Дані збережено, створено обліковий
2	Авторизація користувача	Email, пароль	Доступ до системи, токен
3	Оформлення замовлення	Дані замовлення, контактна інформація	Замовлення збережено, підтвердження
4	Перегляд асортименту товарів	Категорії, запит	Список товарів відповідно до запиту

1	2	3	4
5	Керування товарами (адміністратор)	Інформація про товар (назва, опис, фото,	Оновлений каталог товарів
6	Керування замовленнями (адміністратор)	Інформація про замовлення	Список замовлень, оновлений статус, підтвердження змін
7	Керування клієнтською базою (адміністратор)	Ім’я, прізвище, email, телефон клієнта	Список клієнтів, редаговані дані, підтвердження змін
8	Перегляд і залишення відгуків	Текст відгуку, рейтинг	Відгук збережено або відображено

2.4.1 Вимоги до видів забезпечення.

4.3.1 Система не потребує спеціального математичного забезпечення. Усі покладені на неї функції реалізуються за допомогою стандартних можливостей бази даних та інструментів веброзробки. Додаткові математичні обчислення не передбачаються.

4.3.2 Вимоги до інформаційного забезпечення (ІЗ)

4.3.2.1 Інформаційне забезпечення системи має включати всі необхідні дані для виконання функцій сайту — облік замовлень, управління товарами, зберігання клієнтської інформації та відгуків. Вся інформація зберігається в базі даних у структурованому вигляді, що забезпечує швидкий доступ до неї та зручне оновлення.

4.3.2.2 Необхідно передбачити захист усіх даних від втрати або пошкодження у разі збою живлення чи інших технічних проблем шляхом регулярного створення резервних копій бази даних.

4.3.3 Вимоги до лінгвістичного забезпечення

4.3.3.1 Для розробки вебсайту використовуються мови програмування високого рівня — JavaScript (React для клієнтської частини та Node.js для серверної), а також мова запитів до бази даних MongoDB. Ці інструменти

дозволяють створювати структурований код і забезпечують доступ до інформації, що зберігається в базі.

4.3.3.2 Інтерфейс сайту будується на основі зрозумілих меню, кнопок, форм і підказок. Взаємодія з користувачем повинна бути максимально простою та зрозумілою, орієнтованою на логіку виконання дій. Тексти інтерфейсу оформлюються англійською мовою з можливістю додавання інших мов у майбутньому.

4.3.4 Вимоги до програмного забезпечення (ПЗ).

4.3.4.1 Загальносистемне ПЗ має забезпечувати стабільну роботу вебсайту та обробку всіх функцій. До складу використовуваного програмного забезпечення входить:

- операційна система сервера: Windows
- система керування базою даних: MongoDB

4.3.4.2 Системне ПЗ повинно:

- мати невисокі вимоги до ресурсів
- забезпечувати швидку обробку даних
- повністю відповідати потребам системи

4.3.4.3 Вимоги до ОС

Операційна система має стабільно працювати з вебсервером, базою даних і зовнішніми сервісами. Вона повинна забезпечувати високу швидкість, мінімальне використання ресурсів, захист від збоїв і підтримку сучасного програмного забезпечення. На клієнтському боці сайт має відкриватися в сучасних браузерах (Chrome, Firefox, Safari тощо), незалежно від ОС.

4.3.4.4 Вимоги до СУБД

MongoDB повинна забезпечувати надійне зберігання, пошук і зміну даних, високу швидкодію при запитах, масштабованість, захист доступу, а також роботу з великим обсягом інформації без втрати продуктивності.

4.3.4.5 Засоби введення/виведення

Користувацький інтерфейс повинен:

- надавати адміністратору можливість швидко редагувати товари, бачити список замовлень, переглядати клієнтські дані й завантажувати потрібну інформацію у вигляді зручних таблиць або форм
- показувати повідомлення у випадках помилок (наприклад, при неправильному заповненні форми чи відсутності обов'язкових даних), з можливістю виправлення без перезавантаження сторінки
- реагувати на взаємодію користувача в режимі реального часу: кнопки, форми, підтвердження

4.3.4.6 Вимоги до спеціального ПЗ

Програмне забезпечення розробляється за допомогою сучасних засобів — React (клієнтська частина) та Node.js (сервер). Код має бути модульним, розширюваним, сумісним з іншими частинами системи та легко підтримуваним. Інтерфейс повинен бути зручним і зрозумілим, з можливістю додавання нових функцій без повної переробки системи. Робота з сайтом має здійснюватися через мишу та клавіатуру з доступом до підказок.

4.3.5 Вимоги до технічного забезпечення.

Для роботи системи необхідний інтернет-доступ, сервер з підтримкою Node.js і MongoDB (локальний або хмарний), а також клієнтські пристрої (ПК, ноутбуки, смартфони) з сучасним веббраузером. Уся інфраструктура має забезпечувати стабільне з'єднання, безперервну роботу сайту та доступ до бази даних. Спеціалізованого обладнання не потрібно. Вимоги до метрологічного забезпечення.

4.3.6 Вимоги до метрологічного забезпечення.

Оскільки система не використовує жодних вимірювальних пристроїв чи каналів, вимоги до вимірювального забезпечення не передбачаються.

4.3.7 Вимоги до організаційного забезпечення

4.3.7.1 Організаційне забезпечення сайту реалізується відповідно до вимог чинних стандартів щодо впровадження інформаційних систем на підприємствах.

4.3.7.2 Запровадження сайту Prilipko Design не потребує збільшення штату працівників. Адміністрування здійснюється існуючим персоналом. Місце доступу до адмінпанелі визначається підприємством.

4.3.7.3 Для організації роботи системи необхідно:

- визначити відповідальних осіб, які матимуть доступ до адміністративної частини сайту
- призначити відповідального за підтримку і реагування у випадку технічних збоїв

2.5 Склад і зміст робіт по створенню системи

2.5.1 Стадії створення системи і терміни виконання робіт

Стадії створення системи і терміни виконання робіт наведені у таблиці 2.1.

Таблиця 2.1. Перелік функцій, вхідної та вихідної інформації

№	Найменування робіт	Строки виконання
1	Підготовка користувацької та технічної документації	28.04.2025 – 02.05.2025
2	Тестування функціоналу та виправлення помилок	29.04.2025 – 01.05.2025
3	Реалізація особистого кабінету та адмін-панелі	01.05.2025 – 04.05.2025
4	Розробка клієнтської частини (React)	02.05.2025 – 06.05.2025
5	Розробка серверної частини (Node.js)	04.05.2025 – 06.05.2025
6	Підключення та налаштування MongoDB	06.05.2025 – 09.05.2025
7	Розробка дизайну інтерфейсу сайту	08.05.2025 – 12.05.2025
8	Проєктування структури сайту та бази даних	10.05.2025 – 13.05.2025
9	Розробка технічного завдання	12.05.2025 – 15.05.2025
10	Передпроєктний аналіз діяльності Prilipko Design	14.05.2025 – 17.05.2025

2.5.1 Далі на буде наведено діаграму Ганта (рисунок 2.1), яка була створена на основі термінів зазначених у таблиці 2.1

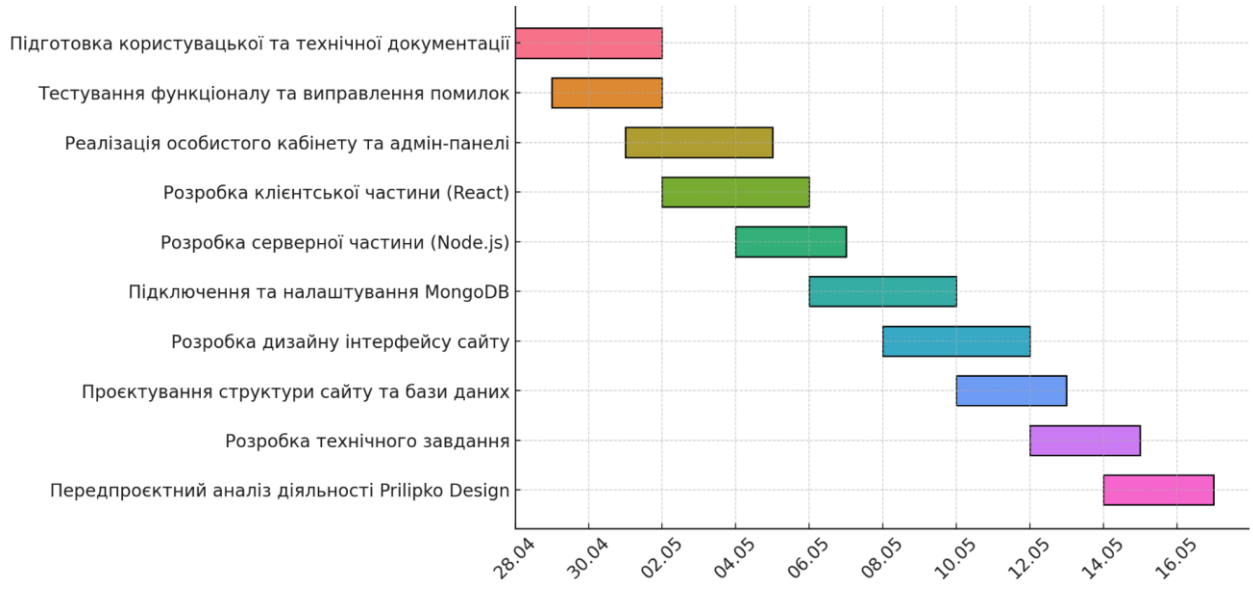


Рисунок 2.1 – Діаграма ганта

2.6 Порядок контролю і приймання системи

2.6.1 Розроблений вебсайт для компанії *Prilipko Design* вводиться в дію на основі узгодженого технічного завдання. Перед запуском сайт повинен пройти приймальні випробування відповідно до вимог ДСТУ 3974-2000.

2.6.2 Випробування проводяться розробником спільно із замовником. Вони включають перевірку працездатності основних функцій: реєстрації, оформлення замовлення, роботи адмін-панелі, безпеки даних. Програму перевірки готує розробник, а затверджує її представник компанії Prilipko Design.

2.6.3 Сайт передається в дослідну експлуатацію на підставі технічного завдання та інструкції користувача. У процесі тестового використання визначаються можливі зауваження, після чого складається перелік доопрацювань із зазначенням рекомендованих термінів їх виконання.

2.6.4 Після завершення дослідної експлуатації та усунення недоліків сайт вводиться в постійну роботу. Факт приймання системи оформлюється актом здачі-приймання між розробником і замовником.

2.7 Вимоги до складу і змісту робіт із підготовки до введення системи в дію

Для запуску вебсайту в роботу замовник виконує комплекс підготовчих дій, що включають:

- забезпечення наявності технічних засобів (комп’ютери, мобільні пристрої з доступом до інтернету) для роботи з сайтом
- організацію навчання персоналу, відповідального за адміністрування сайту, з ознайомленням із функціоналом та інструкцією користувача
- проведення дослідної експлуатації з метою перевірки працездатності основних функцій та збирання зворотного зв’язку від працівників
- після успішного тестування — прийняття рішення про повноцінне введення сайту в дію

2.8 Вимоги до документації

2.8.1 Для створеної інформаційної системи у вигляді вебсайту Prilipko Design розробляється комплекс проєктної документації, що включає технічне завдання та технічний проект

Уся документація оформлюється відповідно до вимог Державних стандартів серій:

- ДСТУ 19 – Єдина система програмної документації
- ДСТУ 24 – Єдина система стандартів автоматизованих систем управління

2.9 Джерела розробки

2.9.1 При розробленні технічного завдання на систему використано наступні документи:

- ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;
- ДСТУ 3973–2000 Система розроблення та поставлення продукції на виробництво;

- ДСТУ ISO/IEC/IEEE 29119-1:2015 Тестування програмного забезпечення. Частина 1. Поняття і визначення;
- ДСТУ ISO 9241-210:2019 Ергономіка взаємодії людина-система. Частина 210. Людиноцентричне проєктування інтерактивних систем.
- ДСТУ Б В.2.5–82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом;
- ДСТУ ISO/IEC 27001:2015 Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги;
- ДСТУ ISO/IEC TR 29110-5-1-2:2015 Інженерія програмного забезпечення. Профілі життєвого циклу для малих організацій;

РОЗДІЛ 3. РОЗРОБЛЕННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Опис та обґрунтування вибору програмно-технічних засобів розроблення програмного продукту

Для створення вебсайту компанії Prilipko Design було обрано сучасні програмно-технічні засоби, які забезпечують надійну, масштабовану та зручну в розробці клієнт-серверну архітектуру. Такий підхід дозволяє розділити логіку застосунку між клієнтською та серверною частинами, що спрощує обслуговування системи, її розширення та модернізацію в майбутньому.

3.1.1 Клієнтська частина

Клієнтська частина (frontend) реалізована з використанням фреймворку React у поєднанні з бібліотекою Redux для управління глобальним станом застосунку. Використання Redux дозволяє централізовано зберігати інформацію про авторизованого користувача, вміст кошика, список товарів, статус замовлень та інші ключові дані, які мають бути доступні в різних компонентах інтерфейсу. Це значно спрощує передавання даних між сторінками та підвищує стабільність роботи застосунку.

Трохи про React та Redux

React — це популярна JavaScript-бібліотека з відкритим кодом, призначена для створення інтерфейсів користувача. Її розробила компанія Meta (раніше Facebook) у 2013 році для спрощення процесу побудови складних вебзастосунків. React дозволяє розробникам створювати багаторазові компоненти, які ефективно оновлюються при зміні даних, забезпечуючи високу продуктивність і зручність у розробці. [1]

React широко використовується для створення односторінкових додатків (SPA), мобільних застосунків за допомогою React Native, а також для серверного рендерингу з використанням таких фреймворків, як Next.js.[2]

Redux — це популярна JavaScript-бібліотека з відкритим кодом, призначена для керування станом у вебзастосунках. Її створили Дан Абрамов і Ендрю Кларк у 2015 році, надихнувшись архітектурою Flux від Facebook та функціональними підходами мови Elm. Redux найчастіше використовується

разом із React для побудови інтерфейсів користувача, але також сумісний з іншими бібліотеками, такими як Angular або Vue.js. [3]

Для інтеграції Redux з React використовується офіційна бібліотека React-Redux, яка забезпечує зручне з'єднання між Redux store та React-компонентами. [4]

3.1.2 Серверна частина

Серверна частина (backend) створена на основі середовища Node.js з використанням фреймворку Express. Це дозволяє обробляти запити до бази даних, керувати логікою авторизації, обліком замовлень, створенням та редагуванням товарів. Node.js дає змогу писати як клієнтську, так і серверну частину однією мовою — JavaScript, що зменшує складність розробки.

Node.js

Node.js — це середовище виконання JavaScript, розроблене на базі високопродуктивного двигуна V8 від компанії Google. Його головною особливістю є можливість запуску JavaScript-коду на стороні сервера, що дозволяє будувати повноцінні серверні застосунки за допомогою єдиної мови програмування — JavaScript [5].

Node.js побудовано на асинхронній неблокуючій моделі обробки подій (event-driven, non-blocking I/O). Такий підхід дозволяє ефективно обробляти велику кількість одночасних запитів без створення окремих потоків для кожного з них. Завдяки цьому Node.js часто використовують для масштабованих вебзастосунків, мікросервісів, REST API, чатів у реальному часі тощо [5].

Express — фреймворк

Express — це мінімалістичний та гнучкий фреймворк для Node.js, який значно спрощує створення серверних застосунків і REST API. Він надає інструменти для організації маршрутизації, обробки запитів, підключення middleware-функцій і налаштування відповідей на HTTP-запити [6].

Express є де-факто стандартом при розробці застосунків на Node.js завдяки своїй простоті, масштабованості та широкій підтримці. Його основна

роль — надати розробнику зручну структуру для написання бекенд-логіки, при цьому не обмежуючи гнучкість налаштувань. [6]

JavaScript

JavaScript (JS) — це високорівнева, інтерпретована мова програмування, яка є однією з ключових технологій у веброзробці поряд з HTML і CSS. Первинно створений для додавання інтерактивності до вебсторінок, JavaScript на сьогодні є повноцінною мовою програмування з широким спектром застосування, зокрема у фронтенді, бекенді, мобільних застосунках, серверній розробці та навіть у машинному навчанні [7].

MongoDB

Для зберігання даних використовується MongoDB — документно-орієнтована база даних NoSQL. Її гнучка структура дозволяє зручно зберігати й обробляти інформацію про користувачів, замовлення, товари, відгуки. Вона добре масштабується та інтегрується з Node.js через бібліотеку Mongoose.

MongoDB — сучасна, документно-орієнтована NoSQL база даних, яка забезпечує гнучке, масштабоване та ефективне зберігання даних. На відміну від класичних реляційних баз (як-от MySQL чи PostgreSQL), MongoDB не використовує таблиці з жорстко визначеними схемами. Замість цього вона працює з документами у форматі BSON (Binary JSON), що дає змогу зберігати дані в динамічній, гнучкій формі [8].

Структура бази даних у проєкті

У рамках кваліфікаційної роботи дані про користувачів, замовлення, товари та відгуки зберігаються у чотирьох колекціях MongoDB:

- users — інформація про зареєстрованих клієнтів (ім’я, email, номер телефону, роль, хешований пароль);
- orders — замовлення, оформлені на сайті (вміст кошика, контактні дані, статус виконання);
- products — перелік товарів, доступних у магазині (назва, опис, ціна, категорія, фото);

- feedbacks — відгуки, залишені користувачами (ім’я, email, номер замовлення, повідомлення, дата створення).

3.1.3 Середовище розробки

Розробка була проведена у середовищі Visual Studio Code, яке забезпечує зручність написання коду, автодоповнення, перевірку синтаксису та інтеграцію зі системами контролю версій (Git).

Visual Studio Code

Visual Studio Code (VS Code) — один із найпопулярніших сучасних редакторів коду, розроблений компанією Microsoft. Він безкоштовний, кросплатформовий та має розширені можливості для роботи з багатьма мовами програмування, зокрема JavaScript, HTML, CSS, а також з Node.js, React та іншими вебтехнологіями [9].

Git

Git — потужний інструмент, який дозволяє зберігати історію змін у коді, працювати в команді, створювати гілки для нових функцій та швидко повертатися до попередніх версій у разі помилок [10].

Git дозволяє:

- створювати **репозиторій**, у якому фіксуються всі зміни (коміти), [10]
- працювати з **гілками (branches)**, що дає змогу вести паралельну розробку без конфліктів, [10]
- об’єднувати зміни (**merge**), вирішувати конфлікти, [10]
- вести централізоване зберігання коду на **платформах типу GitHub**, що полегшує співпрацю та резервне зберігання. [10]

Для проєкту було створено локальний Git-репозиторій, а також віддалений репозиторій на GitHub для збереження та відстеження версій коду. Це дозволяло ефективно керувати кодовою базою, перевіряти зміни, документувати функціонал через коментарі до комітів та забезпечити стабільність проєкту в цілому.

Для тестування, локального запуску та перегляду результатів використовуються локальний сервер, браузер Google Chrome, а також онлайн-інструмент для перевірки запитів Postman.

Postman

Postman — спеціалізований програмний інструмент для тестування REST API. Його головна функція — надсилання HTTP-запитів (GET, POST, PUT, DELETE тощо) до серверу з можливістю перегляду відповіді, аналізу помилок та валідації роботи бекенд-логіки [11].

3.2 Проєктування та створення бази даних

Для зберігання, обробки та структурування даних у дипломному проєкті використано документно-орієнтовану базу даних MongoDB, яка реалізована через хмарну платформу MongoDB Atlas. Такий підхід дозволяє швидко масштабувати застосунок, зручно працювати з гнучкими структурами даних, а також інтегрувати бекенд на Node.js за допомогою бібліотеки Mongoose.

База даних умовно поділяється на чотири основні колекції, які відповідають ключовим сутностям проєкту: користувачі, товари, замовлення та відгуки.

Колекція користувачі (users)

Ця колекція дозволяє реалізувати систему автентифікації, розмежування прав доступу та адміністрування користувачів, вона містить облікові записи користувачів.

Основні поля:

- firstName, lastName — ім’я та прізвище;
- email — електронна пошта, яка є унікальним ідентифікатором;
- password — зберігається в хешованому вигляді;
- phone — номер телефону;
- role — роль користувача: звичайний користувач (user) або адміністратор (admin);
- isBlocked — логічне поле, яке визначає, чи обліковий запис заблокований;

- createdAt, updatedAt — дати створення та оновлення запису.

Колекція товари (products)

Колекція products містить інформацію про товари, що представлені в інтернет-магазині. Така структура дозволяє реалізувати повноцінну систему управління асортиментом, з можливістю фільтрації за категорією та статтю.

Основні поля:

- name, description — назва та опис товару;
- price — вартість;
- category — категорія (наприклад, "Вишиванка");
- image — шлях до зображення;
- inStock — логічне поле наявності;
- gender — цільова категорія товару (наприклад, "him" або "her");
- createdAt — дата створення товару.

Колекція замовлення (orders)

Колекція orders містить дані про замовлення, оформлені користувачами. Це дозволяє відстежувати історію покупок, підтвердження замовлень, а також формувати панель адміністратора з управлінням.

Основні поля:

- user — посилання на користувача (може бути null для гостьового замовлення);
- firstName, lastName, phone, email, address — контактні дані;
- items — масив товарів у замовленні;
- total — загальна вартість;
- confirmed — булеве поле, що вказує, чи підтверджене замовлення;
- status — статус обробки (наприклад, "Confirmed", "In process");
- createdAt — дата створення замовлення.

Колекція відгуків (feedbacks)

Колекція feedbacks містить повідомлення від користувачів. Ця колекція виконує функцію зворотного зв’язку, дозволяючи клієнтам залишати відгуки чи скарги.

Основні поля:

- firstName, lastName, email, phone — дані відправника;
- orderNumber — номер замовлення, до якого стосується звернення (необов'язкове поле);
- subject — тема звернення (наприклад, "order", "other");
- message — текст повідомлення;
- createdAt — дата надсилання.

Уся взаємодія з базою даних відбувається через бекенд-частину на Node.js, яка реалізує API за допомогою фреймворку Express та бібліотеки Mongoose. Кожен HTTP-запит до API (наприклад, створення замовлення, авторизація, перегляд товарів) супроводжується відповідною операцією з базою даних: створенням, оновленням, читанням або видаленням документів.

Основні типи взаємодії з базою даних:

- Створення даних (Create) — додавання нових записів до відповідних колекцій бази даних. Такі запити використовуються при створенні користувачів, замовлень, товарів, відгуків тощо. Здійснюється перевірка введених даних, їхня валідація та збереження у базі.
- Читання даних (Read) — отримання інформації з бази даних, наприклад, для відображення товарів у магазині, перегляду профілю користувача чи перегляду історії замовлень. Ці запити забезпечують вибірку одного або декількох документів із фільтрацією та сортуванням.
- Оновлення даних (Update) — внесення змін до існуючих документів у базі. Це можуть бути зміни в профілі користувача, оновлення інформації про товар, змінення статусу замовлення тощо. Оновлення здійснюється з урахуванням перевірки прав доступу та відповідних правил.

- Видалення даних (Delete) — вилучення записів з бази даних, наприклад, у випадку видалення товару, відгуку або деактивації користувача. Видалення передбачає перевірку відповідності запиту авторизованому користувачу чи адміністратору.

3.3 Реалізація функцій веб-сайту

На цьому етапі роботи було безпосередньо реалізовано функціонал веб-застосунку відповідно до технічного завдання. Система побудована за принципами клієнт-серверної архітектури з використанням сучасного стеку технологій: React — для розробки клієнтської частини, Node.js з Express — для серверної логіки, MongoDB — для зберігання даних.

У цьому підрозділі розглянуто реалізацію ключових функцій веб-сайту, які забезпечують його повну працездатність: реєстрацію та автентифікацію користувачів, керування товарами та замовленнями, функціонал кошика, обробку звернень клієнтів, а також адміністративні можливості для керування системою. Для кожної функції наведено опис, графічні матеріали інтерфейсу, фрагменти коду, пояснення логіки роботи та приклади взаємодії з базою даних.

Користувач (авторизація, профіль, сесія)

Ця група функцій відповідає за базову взаємодію користувача з системою: реєстрацію, авторизацію, редагування особистих даних, збереження сесії. Забезпечується підтримка перевірки унікальності при створенні акаунта, а також захист доступу до приватних маршрутів за допомогою JWT-токенів.(всі функції наведено у таблиці 3.1)

Таблиця 3.1. Перелік реалізованих функцій для Користувача

Функція	**Пояснення**
Реєстрація користувача	Користувач створює обліковий запис, заповнюючи дані (ім’я, email, телефон, пароль). Дані зберігаються у базі з хешованим паролем.
Вхід користувача	Користувач вводить email або телефон і пароль. У разі успішної перевірки повертається JWT-токен.
Вихід із системи	Очищення токена зі сховища клієнта. Серверний логін не потребує активного стану сесії.
Завантаження профілю	Користувачеві повертаються його персональні дані після авторизації за токеном.
Редагування профілю	Користувач може змінити своє ім’я, email або телефон, зберігаючи зміни в базі.
Перевірка унікальності	Під час реєстрації система перевіряє, чи не зареєстровані вже такі email або телефон.
JWT middleware	Захищає маршрути, перевіряючи JWT у заголовку запиту.

Адміністратор (панель керування)

Функції адміністратора дають змогу керувати ключовими сутностями системи: користувачами, товарами, замовленнями та відгуками. Це дозволяє повноцінно обслуговувати онлайн-магазин, контролювати замовлення, перевіряти звернення клієнтів та оновлювати асортимент товарів. (всі функції наведено у таблицях 3.2, 3.3, 3.4, 3.5)

Таблиця 3.2. Перелік реалізованих функцій управлінням клієнтами для Адміну

Функція Управління клієнтами	**Пояснення**
Перегляд користувачів	Таблиця всіх зареєстрованих користувачів із можливістю сортування/пошуку.
Редагування даних користувача	Зміна імені, прізвища, email, телефону користувача.
Зміна ролі користувача	Призначення або зняття ролі адміністратора.
Блокування користувача	Обмеження доступу до системи (isBlocked = true).

Таблиця 3.3. Перелік реалізованих функцій управлінням товарами для Адміну

Управління товарами	**Пояснення**
Додавання нового товару	Через адміністративну форму із завантаженням фото, описом, ціною, тощо.
Редагування товару	Зміна назви, опису, категорії, статі, ціни, наявності.
Видалення товару	Повне видалення товару з бази даних.
Завантаження зображень	Інтеграція форми з бекендом для зберігання файлів у папці /uploads.

Таблиця 3.4. Перелік реалізованих функцій управлінням замовленнями для Адміну

Управління замовленнями	**Пояснення**
Перегляд усіх замовлень	Список усіх замовлень від клієнтів із деталями: товари, контактні дані, статус.
Підтвердження замовлення	Встановлення статусу confirmed = true.
Встановлення статусу виконання	Зміна статусу: “In process”, “Confirmed”, “Completed” тощо.

Таблиця 3.5. Перелік реалізованих функцій управлінням відгуками для Адміну

Управління відгуками та зверненнями	**Пояснення**
Перегляд зворотного зв’язку	Таблиця звернень з форми відгуків: ім’я, email, тема, повідомлення.
Видалення відгуку	Адміністратор може видалити некоректне або неактуальне звернення.

Товари (products)

Ця група охоплює повний цикл управління товаром: від створення до редагування та видалення. Також реалізовано публічну частину перегляду товарів із можливістю фільтрації та детального ознайомлення. Додатково реалізовано функцію завантаження зображень для візуального представлення товару. (всі функції наведено у таблиці 3.6)

Таблиця 3.6. Перелік реалізованих функцій для управління товаром

Функція	**Пояснення**
Створення товару	Адміністратор може додати новий товар через форму в панелі керування.
Редагування товару	Зміна даних товару: опису, ціни, категорії тощо.
Видалення товару	Можливість повністю видалити товар із колекції.
Завантаження фото	Підтримка завантаження зображень через бекенд та відображення їх на сайті.
Перегляд усіх товарів	Користувач бачить весь доступний каталог товарів.
Перегляд одного товару	Детальна сторінка товару за його ідентифікатором.

Кошик (bag)

Кошик є проміжною ланкою між переглядом товарів і оформленням замовлення. Функціонал включає додавання, видалення, редагування кількості товарів, а також збереження стану кошика у локальному сховищі браузера. Це

забезпечує зручність користування навіть без авторизації. (всі функції наведено у таблиці 3.7)

Таблиця 3.7. Перелік реалізованих функцій кошика

Функція	**Пояснення**
Додавання до кошика	Користувач може додати товар до кошика для подальшого замовлення.
Редагування кількості	Зміна кількості кожного товару безпосередньо в кошику.
Збереження стану	Кошик зберігається в локальному сховищі для збереження між сесіями.
Очистка кошика	Після оформлення замовлення кошик очищається автоматично.

Замовлення (orders)

Центральна частина логіки онлайн-магазину. Користувачі мають змогу оформлювати замовлення, а система — зберігати їх у базі, підтверджувати адміністратором і відображати відповідний статус. Реалізовано як підтримку зареєстрованих, так і гостьових користувачів. (всі функції наведено у таблиці 3.8)

Таблиця 3.8. Перелік реалізованих функцій замовлень

Функція	**Пояснення**
1	2
Оформлення замовлення	Форма для введення контактних і платіжних даних та підтвердження покупки.
Збереження в MongoDB	Усі дані замовлення записуються до бази даних.
Сторінка підтвердження	Користувач бачить повідомлення про успішне замовлення.
Перегляд замовлень (user)	Користувач бачить історію своїх покупок у профілі.
Перегляд замовлень (admin)	Адміністратор бачить таблицю всіх замовлень у панелі.

1	2
Підтвердження замовлення	Адміністратор відмічає статус "підтверджено" у базі.
Статус замовлення	Система відображає поточний статус кожного замовлення.

Відгуки (feedbacks)

Реалізовано механізм зворотного зв’язку з клієнтами: форма звернення дозволяє надсилати повідомлення щодо проблеми, замовлення чи іншої теми. Адміністратор має можливість переглядати всі отримані звернення та при потребі реагувати на них. (всі функції наведено у таблиці 3.9)

Таблиця 3.9. Перелік реалізованих функцій відгуків

Функція	**Пояснення**
Залишення відгуку	Користувач може залишити звернення або скаргу через форму.
Збереження у базі	Відгук зберігається в колекції feedbacks із зазначенням контактних даних.
Перегляд адміністратором	Адміністратор може переглядати всі отримані звернення.

Гостьовий доступ і маршрутизація

Система передбачає можливість взаємодії без обов’язкової реєстрації. Користувач може оформити замовлення як гість, а також обирати між входом, реєстрацією або гостьовим доступом. Приватні маршрути захищені middleware, що перевіряє дійсність токена. (всі функції наведено у таблиці 3.10)

Таблиця 3.10. Перелік реалізованих функцій для гостьового доступу та маршрутизації

Функція	Пояснення
Гостьове оформлення замовлення (без авторизації)	Користувач може оформити замовлення без входу в систему. Дані зберігаються у базі.
Захист приватних маршрутів (JWT middleware)	Приватні маршрути захищені middleware, яке перевіряє дійсність токена доступу.

Інфраструктура

Ця категорія охоплює технічні компоненти, необхідні для стабільної роботи системи: підключення до хмарної бази даних MongoDB Atlas, структурування маршрутів за допомогою Express, захист через JWT і CORS, хешування паролів, обробку зображень. (всі функції наведено у таблиці 3.11)

Таблиця 3.11. Перелік реалізованих функцій інфраструктури проекту

Функція	Пояснення
MongoDB Atlas	Хмарна база даних для зберігання всіх структурованих даних.
Express маршрути	Серверні маршрути організують обробку запитів клієнта.
Upload зображень	Система дозволяє завантажувати й зберігати файли на сервері.
JWT, bcrypt, cors	Використання токенів, хешування паролів і політика доступу до API.

Для кожної моделі кластерів з БД розроблено окрему модель

Приклад models/Feedback.js (рисунок 3.1)

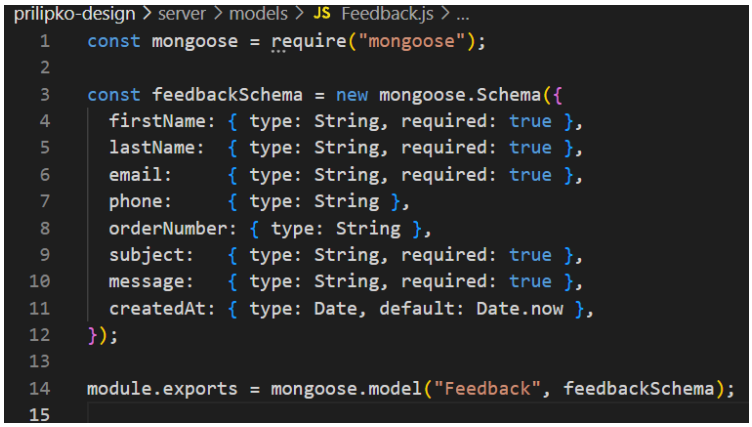


Рисунок 3.1 – Модель відгуків

Захист приватних марштурів (роутів)

Для захисту приватних маршрутів у проєкті реалізовано файл authMiddleware.js, який перевіряє автентифікацію користувача на основі JWT-токенів. Це дозволяє контролювати доступ до персональних даних, замовлень та адміністративного функціоналу. Middleware-функції required, optional та requiredAdmin забезпечують відповідні рівні перевірки доступу.

prilipko-design > server > middleware > JS authMiddleware.js > optional

```
const jwt = require("jsonwebtoken");

// 🔐 Middleware: авторизований користувач
const required = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    return res.status(401).json({ message: "Unauthorized" });
  }

  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.userId = decoded.userId;
    req.userRole = decoded.role;
    next();
  } catch (err) {
    return res.status(401).json({ message: "Invalid token", error: err.message });
  }
};

// 🔐 Middleware: лише для адміністратора
const requiredAdmin = (req, res, next) => {
  required(req, res, () => {
    if (req.userRole !== "admin") {
      return res.status(403).json({ message: "Forbidden: admin only" });
    }
    next();
  });
};

// 🟡 Middleware: не обов'язкова авторизація
const optional = (req, res, next) => {
  const authHeader = req.headers.authorization;
  if (!authHeader || !authHeader.startsWith("Bearer ")) {
    return next(); // неавторизований — дозволити
  }

  const token = authHeader.split(" ")[1];
  try {
    const decoded = jwt.verify(token, process.env.JWT_SECRET);
    req.userId = decoded.userId;
    req.userRole = decoded.role;
  } catch {
    req.userId = null;
    req.userRole = null;
  }

  next();
};

module.exports = {
  required,
  requiredAdmin,
  optional,
};
```

Рисунок 3.2 – Код authMiddleware.js

Підключення БД із файлу server.js з налаштованими роутингами

prilipko-design > server > JS server.js > ...

```
require("dotenv").config();
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

const authRoutes = require("./routes/authRoutes");  // логін / реєстрація / Профіль (GET /profile)
const userRoutes = require("./routes/user");
const adminRoutes = require("./routes/admin");
const adminProductsRoutes = require("./routes/adminProducts");
const publicRoutes = require("./routes/publicProducts");
const orderRoutes = require("./routes/orderRoutes");
const feedbackRoutes = require("./routes/feedbackRoutes");

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Підключення до бази даних
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => console.log("MongoDB connected"))
  .catch((err) => console.error("MongoDB error:", err));

// Роутинги
app.use("/api/auth", authRoutes);   // /api/auth/profile (GET)
app.use("/api/user", userRoutes);   // /api/user/profile (PUT)
app.use("/api/admin", adminRoutes); // /api/admin
app.use("/api/admin/products", adminProductsRoutes); //api/admin/products
app.use("/api/products", publicRoutes); ///api/products
app.use("/api/orders", orderRoutes);
app.use("/api/feedback", feedbackRoutes);

const path = require("path");
app.use("/api/upload", require("./routes/upload"));
app.use("/uploads", express.static(path.join(__dirname, "/uploads")));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Рисунок 3.3 – Код server.js

Підключення бд в файлі .env (також пароль для JWT хешування)

```
prilipko-design > server > .env
1   MONGO_URI=mongodb+srv://prilipko_admin:12345@cluster0.ttplw5v.mongodb.net/?retryWrites=true&w=majority&appName=Cluster0
2   PORT=5000
3
4   JWT_SECRET=supersecretkey123
```

Рисунок 3.4 – Код .env

3.4 Інструкція користувача

Реєстрація та вхід

Вхід до облікового запису

1. Перейдіть на вкладку Login.
2. Введіть свій Email та Password.
3. Натисніть кнопку LOG IN.
4. Після успішного входу ви будете перенаправлені на сторінку профілю

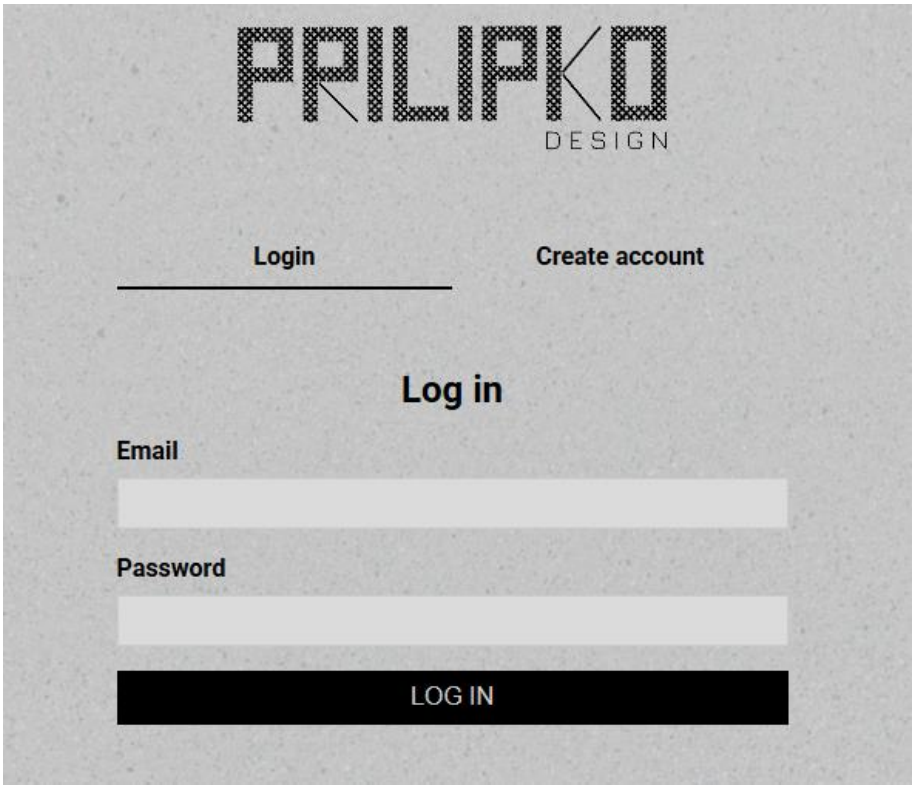


Рисунок 3.5 - Вхід до облікового запису

Створення облікового запису

1. Перейдіть на вкладку Create account.
2. Заповніть усі поля:
3. Натисніть кнопку REGISTER.
4. Після реєстрації вас автоматично залогінить під вашим акаунтом.

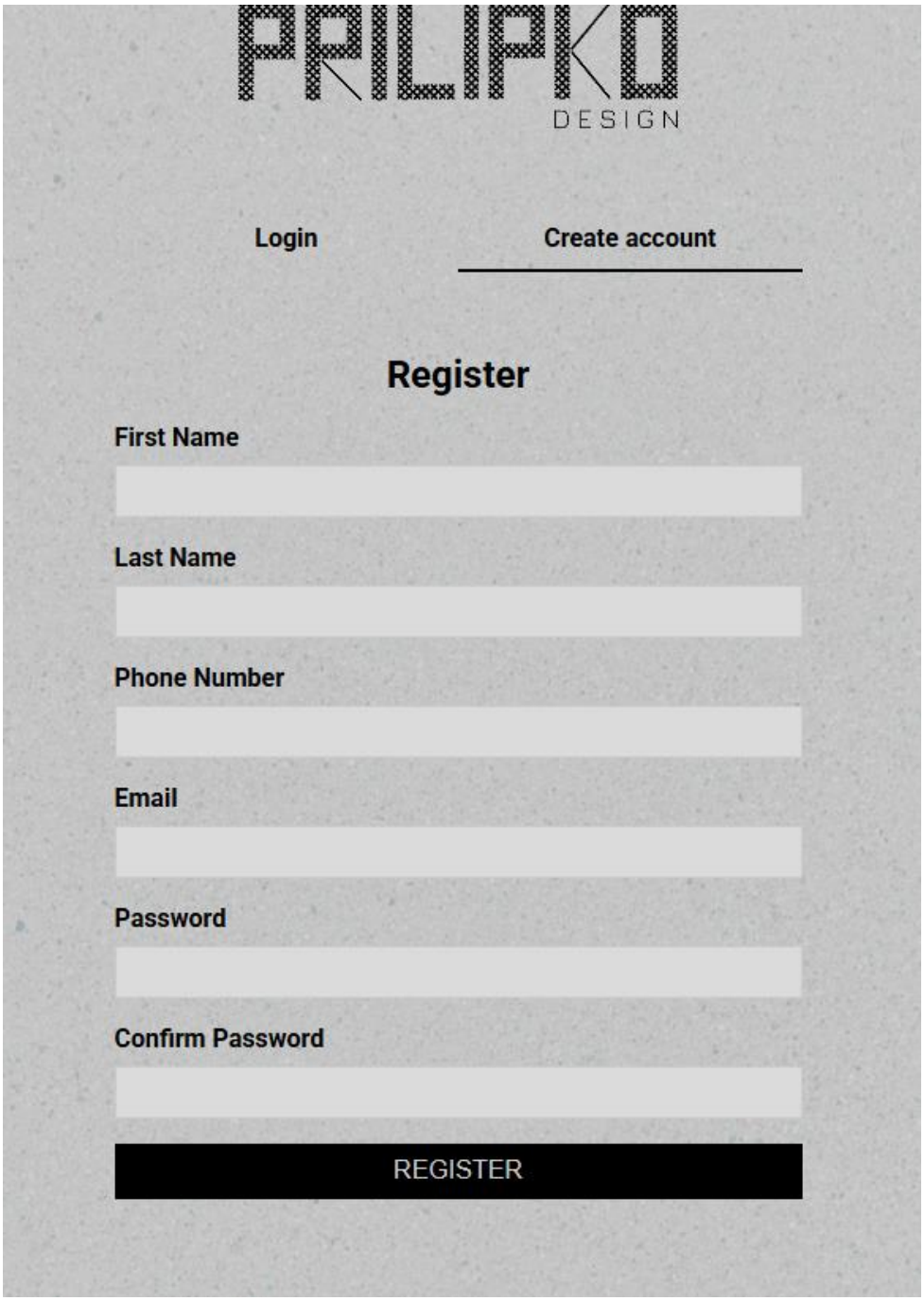


Рисунок 3.6 – Реєстрація на сайті

Профіль

Перегляд профілю (Додаток Б, Рисунок Б.1)

На сторінці MY ACCOUNT ви можете переглянути свої особисті дані:

- Ім’я, прізвище
- Email
- Номер телефону
- Прихований пароль

Щоб змінити дані профілю — натисніть Edit

Щоб змінити пароль — натисніть Change

Перегляд замовлень (Додаток Б, Рисунок Б.2)

У блоці My orders відображаються ваші замовлення:

- ID замовлення
- Загальна сума (Total)
- Дата оформлення (Date)
- Список товарів
- Статус (наприклад: *IN PROCESS*)

Навігація по сайту

У верхній частині сайту розташоване головне меню:

- MY ACCOUNT — перехід до особистого кабінету користувача
- BAG — перегляд кошика з обраними товарами

У лівій частині екрана відкривається бокове меню з основними розділами:

- SHOP
  - *FOR HIM* — категорія товарів для чоловіків
  - *FOR HER* — категорія товарів для жінок
- HISTORY — перегляд історії активностей
- CONTACT — сторінка з контактною інформацією

Рисунок 3.7 – Меню навігації сайту

Сторінка магазину

На цій сторінці відображаються товари, доступні для покупки. (Додаток Б, Рисунок Б.3)

- Кожен товар представлений фотографією, назвою та ціною.
- Натиснувши на зображення товару, можна перейти до його детального перегляду.
- В асортименті можна знайти вишиванки та інші елементи одягу.

Товари розділені за категоріями: наприклад, *FOR HIM* або *FOR HER*, відповідно до вибраного розділу у меню.

Сторінка товару

Після натискання на товар відкривається його детальна сторінка:

- Велике зображення товару
- Назва і ціна (наприклад: *Вишиванка біло-червона — 4500 UAH*)
- Кнопка ADD TO BAG — додає товар у кошик

Нижче розміщено додаткову інформацію:

- DETAILS — короткий опис товару
- CARE — правила догляду
- RETURNS / EXCHANGES — умови повернення й обміну

Кнопка BACK TO ALL повертає до загального списку товарів.

Рисунок 3.8 – Сторінка обраного товару «Вишиванка біло-червона»

Кошик

На сторінці BAG користувач може переглянути товари, додані до замовлення:

- Назва товару, зображення та ціна
- Можна змінити кількість одиниць або видалити товар, натиснувши на хрестик

У правій частині вказано:

- Subtotal — проміжна сума
- Sales Tax — податок (якщо застосовується)
- Estimated Total — загальна сума до сплати

Для переходу до оформлення замовлення натисніть кнопку CHECKOUT. Нижче є посилання для повернення до покупок (Continue Shopping). (Додаток Б, Риуснок Б.4).

Формування замовлення для авторизованого користувача

На сторінці CHECKOUT користувач заповнює форму доставки. (Додаток Б, Рисунок Б.5)

- Ім’я
- Прізвище
- Телефон
- Email
- Адреса

Праворуч відображається Order Summary з коротким описом замовлення і загальною сумою.

Для завершення покупки натисніть кнопку PLACE ORDER.

Після успішного оформлення замовлення з’являється повідомлення:
Order successfully created
Нижче вказано номер замовлення.
Натиснувши RETURN TO HOME PAGE, користувач повертається на головну сторінку сайту.

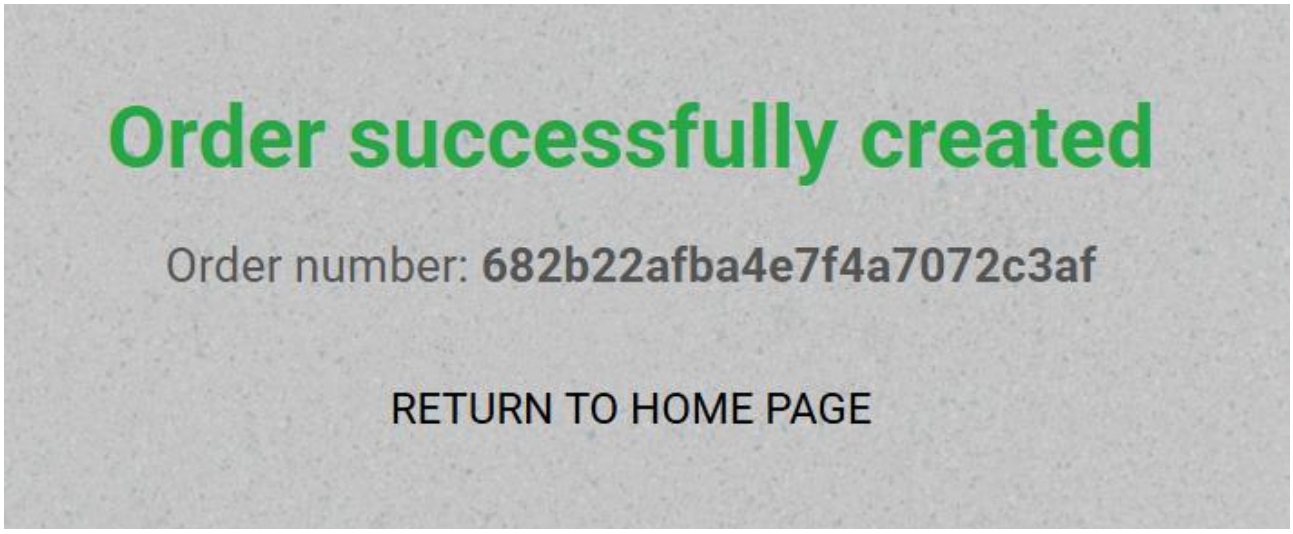


Рисунок 3.9 – Підтвердження замовлення та його оформлення

Формування замовлення без реєстрації

Після переходу до оформлення замовлення користувач бачить три варіанти:

- GUEST CHECKOUT — оформлення замовлення без створення облікового запису.

  Натисніть CHECKOUT AS GUEST, щоб перейти до форми доставки.
- RETURNING CUSTOMER — вхід для зареєстрованих користувачів. Введіть email і пароль, натисніть LOGIN.
- CREATE ACCOUNT — створення нового акаунта. Натисніть CREATE ACCOUNT, щоб перейти до реєстрації.

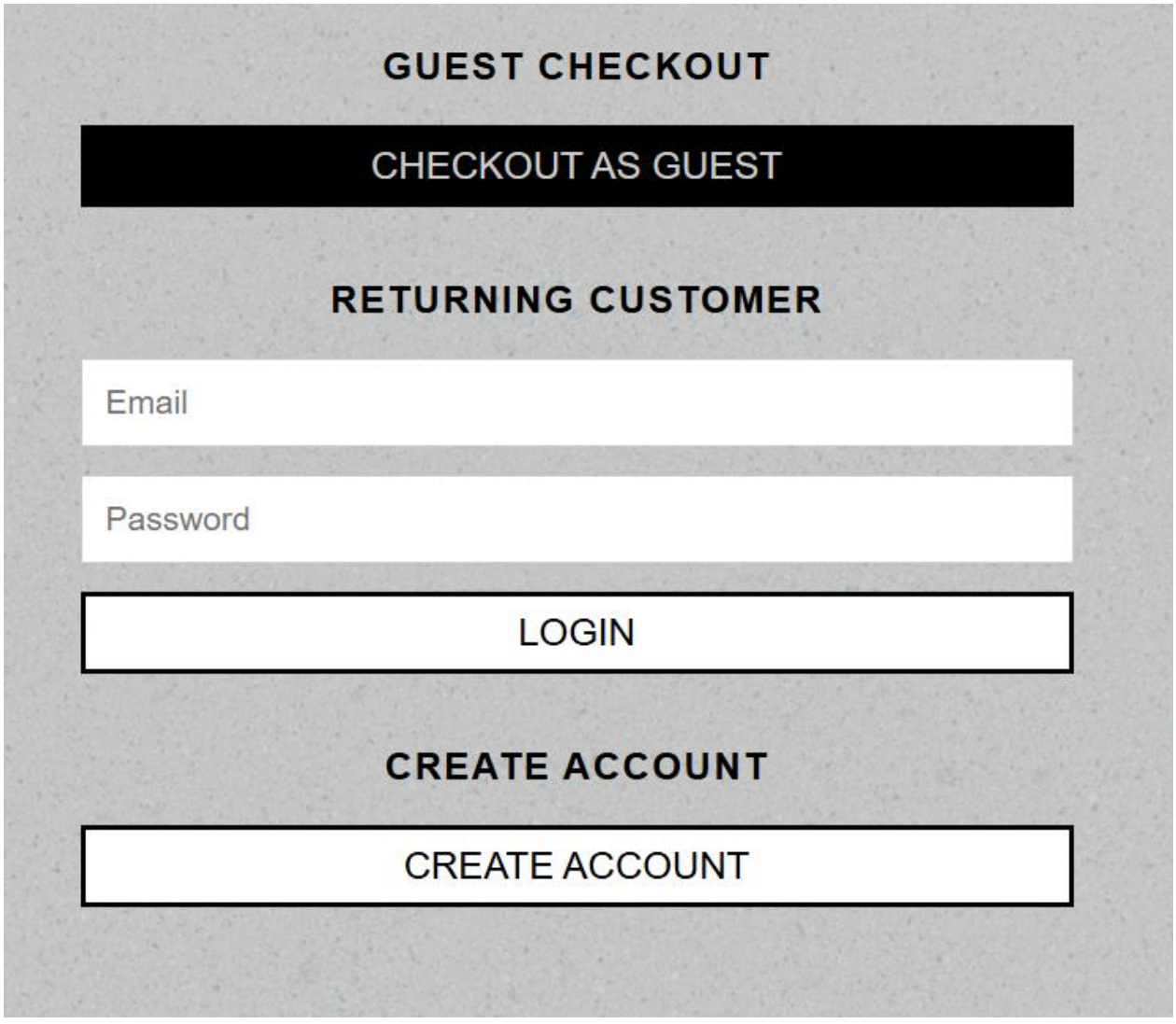


Рисунок 3.10 – Формування замовлення без реєстрації

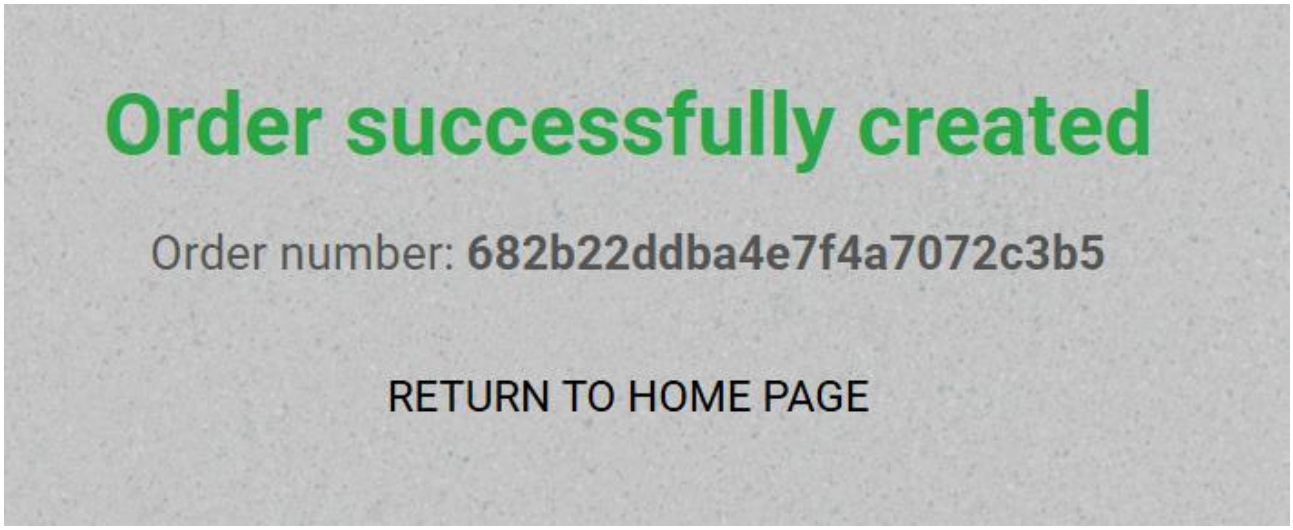


Рисунок 3.11 - Формування замовлення без реєстрації, підтверджене замовлення

Заповенення форми відгуку

На сторінці SEND US A MESSAGE користувач може залишити повідомлення або відгук.

Для цього необхідно заповнити форму:

- First Name — ім’я
- Last Name — прізвище
- Email — електронна пошта
- Phone Number — номер телефону
- Order Number — номер замовлення (необов’язково)
- Subject — тема звернення (наприклад: Feedback, Question, Return)
- Message — текст повідомлення

Після заповнення всіх полів натисніть кнопку SUBMIT.
Під формою вказано, що надсилаючи повідомлення, ви погоджуєтеся з умовами сайту.

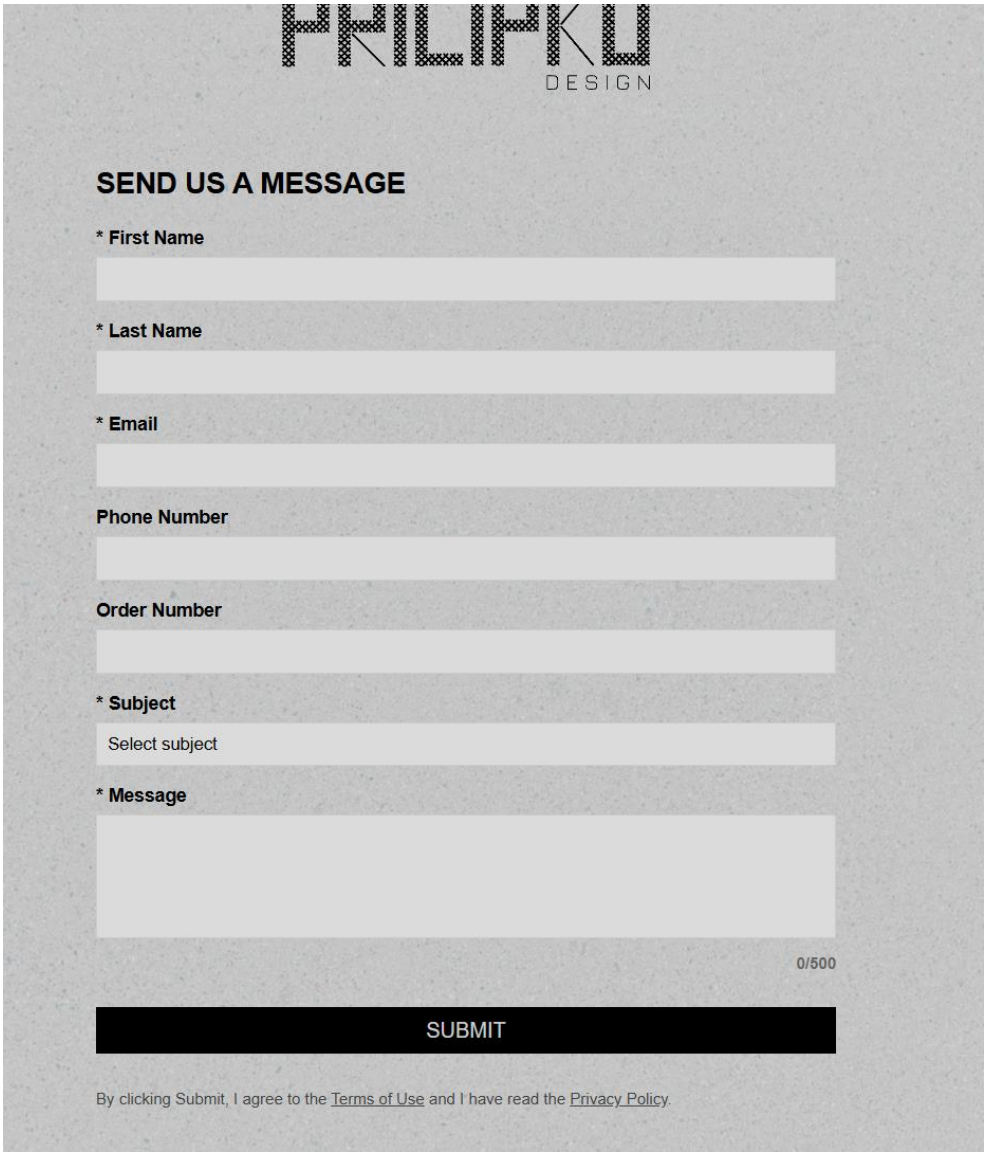


Рисунок 3.12 – Вкладка формування фідбеку від користувача

Користувач адмін

Адмін-панель: Користувачі

У розділі Користувачі адміністратор бачить таблицю зі списком зареєстрованих користувачів. (Додаток Б, Рисунок Б.6)

Для кожного користувача відображено:

- Ім’я
- Email
- Телефон
- Роль (*admin* або *user*)

- Статус (*Активний* або інший)

Функції адміністратора:

- Зберегти / Видалити — доступно для користувача з роллю *admin*
- Редагувати — відкриває форму зміни даних користувача (ім’я, email, телефон тощо)
- Заблокувати — змінює статус користувача (наприклад, для тимчасового обмеження доступу)

Адмін-панель: Товари

У розділі Список товарів адміністратор має змогу. (Додаток Б, Рисунок Б.7)

- Переглядати всі наявні товари в таблиці (назва, ціна, категорія, фото, наявність)
- Шукати та редагувати товари
- Додавати нові позиції

Для редагування чи додавання товару відкривається форма:

- *Назва* — вкажіть назву товару
- *Ціна* — вартість у гривнях
- *Категорія* — наприклад, Вишиванка
- *Опис* — короткий опис товару
- *Стать* — вибір призначення: для чоловіків чи жінок
- *Фото* — прикріплення зображення товару
- *Наявність* — позначка про доступність у продажу

Після заповнення даних натисніть Зберегти, або Скасувати, щоб вийти без змін.

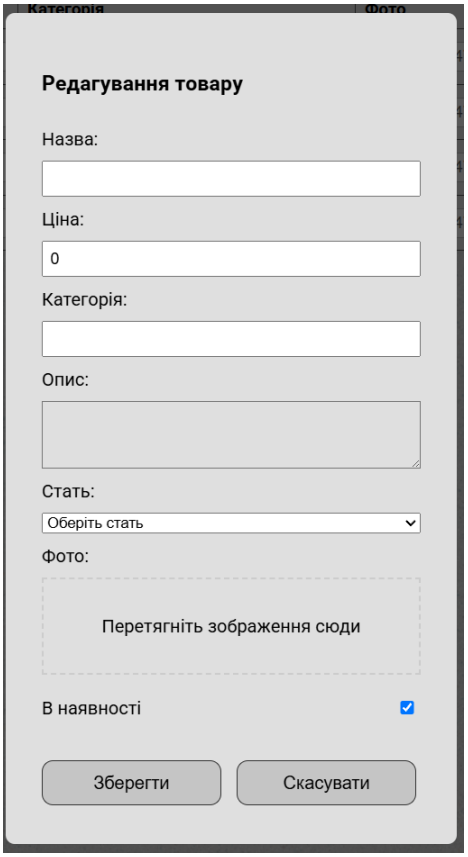


Рисунок 3.13 - Управління товарами через адмін -панель, додавання товару

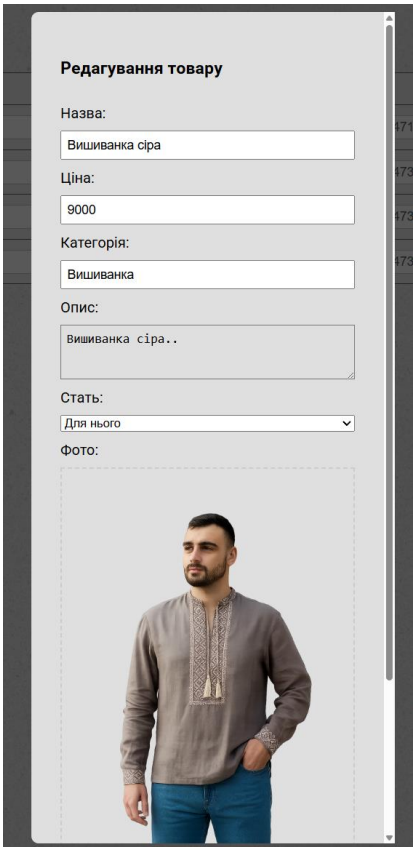


Рисунок 3.14 - Управління товарами через адмін-панель, редагування товару

Адмін-панель: Замовлення

В розділі Список замовлень адміністратор бачить деталі всіх оформлених замовлень (Додаток Б, Рисунок Б.8)

Для кожного замовлення відображається:

- Email покупця
- Телефон
- Перелік замовлених товарів
- Сума замовлення
- Статус (In process, Confirmed, Canceled тощо)
- Дата оформлення

Функції адміністратора:

- Редагувати — дає змогу змінити статус замовлення (наприклад, з *In process* на *Confirmed*)
- Видалити — повне видалення замовлення
- Зберегти / Скасувати — кнопки з’являються після внесення змін

Адмін-панель: Відгуки

У розділі Повідомлення від користувачів відображається таблиця всіх надісланих через форму зворотного зв’язку повідомлень. (Додаток Б, Рисунок Б.9)

Для кожного повідомлення зазначено:

- Ім’я та email відправника
- Телефон
- Номер замовлення (якщо вказано)
- Тема повідомлення (наприклад: *support*, *order*, *other*)
- Текст повідомлення
- Дата та час надсилання

Функція адміністратора:

- Видалити — дозволяє видалити повідомлення зі списку

3.5 Тестування програмного продукту

В таблиці 3.12 наведено приклади тестів, що було проведено для перевірки ключових функцій сайту.

Таблиця 3.12 Результати тестування веб-сайту

№	Пріоритет	Перевірка	Очікуваний результат	Google Chrome	Microsoft Edge	Safari (моб.)
1	Високий	Реєстрація з валідними даними	Користувач успішно реєструється, автоматичний перехід до профілю	Passed	Passed	Passed
2	Високий	Вхід з правильними email/паролем	Користувач авторизується, з’являється доступ до приватного профілю	Passed	Passed	Passed
3	Високий	Вхід з неправильним паролем	Повідомлення про помилку автентифікації	Passed	Passed	Passed
4	Середній	Перегляд списку товарів	Відображаються всі товари з зображеннями та цінами	Passed	Passed	Passed
5	Високий	Перегляд одного товару	Відкривається сторінка з детальною інформацією про товар	Passed	Passed	Passed
6	Високий	Додавання товару до кошика	Товар зберігається у кошику, лічильник оновлюється	Passed	Passed	Passed
7	Високий	Оформлення замовлення як гість	Замовлення успішно створюється, відображається підтвердження	Passed	Passed	Passed

№	Пріоритет	Перевірка	Очікуваний результат	Google Chrome	Microsoft Edge	Safari (моб.)
8	Середній	Редагування профілю користувача	Дані змінюються, зберігаються у базі, виводяться оновленими	Passed	Passed	Passed
9	Високий	Захист приватного маршруту без токена	Повертається статус 401 / переадресація на сторінку входу	Passed	Passed	Passed
10	Низький	Перевірка адаптивності сторінки товару	Контент коректно масштабується для мобільного перегляду	Passed	Passed	Passed

Проведене тестування функціональних можливостей вебсайту показало, що основні сценарії роботи системи виконуються коректно в усіх перевірених браузерах: Google Chrome, Microsoft Edge та Safari (мобільна версія). Усі критично важливі функції — авторизація, реєстрація, перегляд товарів, оформлення замовлень, захист приватних маршрутів — успішно пройшли перевірку. Адаптивність інтерфейсу також забезпечує коректне відображення на мобільних пристроях. Результати тестування підтверджують готовність системи до реального використання.

ВИСНОВОК

У результаті виконання роботи було проведено аналіз діяльності компанії Prilipko Design, зокрема її внутрішніх бізнес-процесів, особливостей роботи з клієнтами, структури асортименту та способів комунікації. На основі виявлених проблем та потреб було прийнято рішення розробити вебсайт для оформлення онлайн-замовлень, який дозволить автоматизувати та покращити ключові етапи взаємодії з клієнтами.

Запропоноване рішення передбачає створення клієнт-серверного вебзастосунку з зручним інтерфейсом, адаптованим під різні пристрої, можливістю авторизації, реєстрації, перегляду асортименту, формування кошика, оформлення замовлення як зареєстрованим, так і гостьовим користувачем. Для адміністратора реалізовано окрему панель керування, що дозволяє працювати з товарами, замовленнями, користувачами та зверненнями.

Це рішення дозволяє значно покращити доступність продукції компанії завдяки створенню зручної онлайн-платформи, яка доступна користувачам у будь-який час і з будь-якого пристрою. Впровадження вебсайту сприяє скороченню часу обробки замовлень, автоматизуючи процеси прийому та обліку, що, своєю чергою, зменшує навантаження на персонал. Крім того, система дозволяє систематизувати інформацію про клієнтів і замовлення, зберігаючи її в централізованій базі даних. Реалізований модуль зворотного зв’язку забезпечує прямий канал комунікації з клієнтами, що підвищує рівень сервісу. У довгостроковій перспективі розроблений сайт може стати основою для подальшої цифрової трансформації бізнесу, включаючи інтеграцію з додатковими сервісами та аналітичними інструментами.

Розробка та запуск вебсайту для компанії Prilipko Design виявилися обґрунтованим і раціональним рішенням з економічної точки зору. Витрати на реалізацію становили близько 30 000 грн, при цьому очікуваний річний прибуток або заощадження, отримані завдяки автоматизації бізнес-процесів, становлять приблизно 60 000 грн. Це свідчить про те, що вкладення окупаються вже протягом першого року використання.

Сайт дозволяє зменшити залежність від ручної праці, пришвидшує обробку замовлень, а також відкриває компанії нові можливості для залучення клієнтів через онлайн-канал.

Таким чином, впровадження вебресурсу — це не лише технічне оновлення, а й важливий крок у напрямку розвитку компанії, який сприяє підвищенню ефективності, покращенню клієнтського сервісу та створенню основи для масштабування бізнесу в майбутньому.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Wikipedia React (software). URL: https://en.wikipedia.org/wiki/React_(software) (дата звернення: 09.05.2025).
2. Brainlab. Що таке React JS? Як почати вивчати Реакт?. URL: https://brainlab.com.ua/uk/blog-uk/shho-take-react-js-yak-pochaty-vyvchaty-reakt (дата звернення: 09.05.2025).
3. Wikipedia. Redux. URL: https://uk.wikipedia.org/wiki/Redux (дата звернення: 09.05.2025).
4. Redux.js.org. React Redux. URL: https://redux.js.org/faq/react-redux (дата звернення: 10.05.2025).
5. Nodejs.org. About. URL: https://nodejs.org/en/about (дата звернення: 10.05.2025).
6. Expressjs. Guide. URL: https://expressjs.com/en/guide/routing.html (дата звернення: 10.05.2025).
7. Mdn web docs. JavaScript Guide. URL: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Guide (дата звернення: 11.05.2025).
8. MongoDB. About. URL: https://www.mongodb.com/ (дата звернення: 12.05.2025).
9. Visual Studio Code. Blog. URL: https://code.visualstudio.com/ (дата звернення: 12.05.2025).
10. Git. About. URL: https://git-scm.com/about/branching-and-merging (дата звернення: 12.05.2025).
11. Postman API Platform. learning. URL: https://www.postman.com/ (дата звернення: 12.05.2025).
12. ДСТУ 3008-2015. Документація. Звіти у сфері науки і техніки. Структура та правила оформлювання;
13. ДСТУ 3973–2000 Система розроблення та поставлення продукції на виробництво;
14. ДСТУ ISO/IEC/IEEE 29119-1:2015 Тестування програмного забезпечення. Частина 1. Поняття і визначення;

15. ДСТУ ISO 9241-210:2019 Ергономіка взаємодії людина-система. Частина 210. Людиноцентричне проєктування інтерактивних систем.

16. ДСТУ Б В.2.5–82:2016 Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом;

17. ДСТУ ISO/IEC 27001:2015 Інформаційні технології. Методи захисту. Системи управління інформаційною безпекою. Вимоги;

18. ДСТУ ISO/IEC TR 29110-5-1-2:2015 Інженерія програмного забезпечення. Профілі життєвого циклу для малих організацій;

19. ДСТУ 3973-2000. СИСТЕМА РОЗРОБЛЕННЯ ТА ПОСТАВЛЕННЯ ПРОДУКЦІЇ НА ВИРОБНИЦТВО. Чинний від 27.11.2000. Київ : ДЕРЖСТАНДАРТ УКРАЇНИ, 2001. 43 с. URL: https://www.pdau.edu.ua/sites/default/files/node/2805/dstu39732000.pdf (дата звернення: 15.05.2025).

20. ДСТУ 3008:2015. ЗВІТИ У СФЕРІ НАУКИ І ТЕХНІКИ. На заміну ДСТУ 3008-95 ; чинний від 22.06.2015. Київ : ДП «УкрНДНЦ», 2016. 26 с. URL: https://science.kname.edu.ua/images/dok/derzhstandart_3008_2015.pdf (дата звернення: 15.05.2025).

21. ДСТУ 2293:2014. ОХОРОНА ПРАЦІ. Терміни та визначення основних понять. На заміну ДСТУ 2293-99 ; чинний від 01.05.2015. Київ : МІНЕКОНОМРОЗВИТКУ УКРАЇНИ, 2015. 18 с. (дата звернення: 15.05.2025).

22. ДСТУ ISO 6309:2007. ПРОТИПОЖЕЖНИЙ ЗАХИСТ. Знаки безпеки Форма та колір. На заміну ГОСТ 12.4.026–76 ; чинний від 30.03.2007. Київ : ДЕРЖСПОЖИВСТАНДАРТ УКРАЇНИ, 2008. 12 с. URL: https://www.ksv.biz.ua/GOST/DSTY_ALL/DSTY1/dsty_iso_6309-2007.pdf (дата звернення: 15.05.2025).

23. ДСТУ EN 13306:2019. ТЕХНІЧНЕ ОБСЛУГОВУВАННЯ. Терміни та визначення понять. Чинний від 01.10.2007. Київ : ДЕРЖСПОЖИВСТАНДАРТ УКРАЇНИ, 2008. 34 с. (дата звернення: 15.05.2025).

24. ДСТУ ISO/IEC 27001:2015. МЕТОДИ ЗАХИСТУ СИСТЕМИ УПРАВЛІННЯ ІНФОРМАЦІЙНОЮ БЕЗПЕКОЮ. На заміну ДСТУ ISO/IEC 27001:2010 ;

чинний від 01.01.2017. Київ : ДП «УкрНДНЦ», 2016. 28 с. URL: https://www.assistem.kiev.ua/doc/dstu_ISO-IEC_27001_2015.pdf (дата звернення: 15.05.2025).

25. ДСТУ EN 12464-1:2016. Світло та освітлення. Освітлення робочих місць. Частина 1. Внутрішні робочі місця.. Чинний від 01.12.2017. Київ : ДП «УкрНДНЦ», 2018. 47 с. (дата звернення: 15.05.2025).
26. ДБН В.2.5-28-2006. ПРИРОДНЕ І ШТУЧНЕ ОСВІТЛЕННЯ. На заміну СНиП II-4-79 ; чинний від 15.05.2006. Київ : Мінбуд України, 2006. 96 с.
27. ECMAScript 2022 Language Specification. *ECMAScript*. 01.01.2022. URL: https://262.ecma-international.org/13.0/ (дата звернення: 15.05.2025).
28. ДСТУ Б В.2.5-82:2016. ЕЛЕКТРОБЕЗПЕКА В БУДІВЛЯХ І СПОРУДАХ Вимоги до захисних заходів від ураження електричним струмом. На заміну ДБН В.2.5-27-2006 ; чинний від 01.04.2017. Київ : ДП "УкрНДНЦ", 2016. 110 с. URL: http://www.tsatu.edu.ua/ettp/wp-content/uploads/sites/25/dstu-b-v.2.5-82-2016-elektrobezpeka-v-budivljah-i-sporudah-1.pdf (дата звернення: 15.05.2025).
29. Методичні рекомендації до виконання кваліфікаційної роботи на здобуття освітнього ступеня «бакалавр» спеціальності 122 «Комп’ютерні науки» освітньо-професійної програми «Інформаційні системи та штучний інтелект» денної форми здобуття освіти [Електрон. ресурс] / уклад. С. В. Грибков, Н. В. Лімaнська, М. П. Костіков. – К.: НУХТ, 2025. – 43 с.
30. ДСТУ ISO/IEC 25051:2016. ВИМОГИ ДО ЯКОСТІ СИСТЕМ І ПРОГРАМНИХ ЗАСОБІВ ТА ЇЇ ОЦІНЮВАННЯ (SQuaRE). На заміну на заміну ДСТУ ISO/IEC 25051:2015 ; чинний від 27.12.2016. Київ : Не є офіційним виданням., 2017. 15 с. (дата звернення: 15.05.2025).
31. ДСТУ ISO/IEC/IEEE 29119-1:2017. ІНЖЕНЕРІЯ СИСТЕМ І ПРОГРАМНИХ ЗАСОБІВ ТЕСТУВАННЯ ПРОГРАМНИХ ЗАСОБІВ. На заміну НА ЗАМІНУ ДСТУ ISO/IEC/IEEE 29119-1:2015 ; чинний від 19.12.2017. Київ : Не є офіційним виданням., 2018. 20 с. (дата звернення: 15.05.2025).

32. ДСТУ EN ISO 9241-210:2022. Ергономіка взаємодії людина-система.. На заміну На заміну ДСТУ EN ISO 9241-210:2019 ; чинний від 28.12.2022. Київ : Не є офіційним виданням. 21 с. (дата звернення: 15.05.2025).
33. ДСТУ 3974-2000. СИСТЕМА РОЗРОБЛЕННЯ ТА ПОСТАВЛЕННЯ ПРОДУКЦІЇ НА ВИРОБНИЦТВО. На заміну - ; чинний від 27.11.2000. Київ : ДЕРЖСТАНДАРТ УКРАЇНИ, 2001. 48 с. (дата звернення: 15.05.2025).
34. ДСТУ ISO/IEC/IEEE 26512:2018. ІНЖЕНЕРІЯ СИСТЕМ І ПРОГРАМНИХ ЗАСОБІВ. На заміну - ; чинний від 18.12.2018. Київ : Не є офіційним виданням. 20 с. (дата звернення: 15.05.2025).
35. ДСТУ 2293-99. ОХОРОНА ПРАЦІ. На заміну - ; чинний від 01.01.2000. Київ : Не є офіційним виданням., 1999. 17 с. (дата звернення: 15.05.2025).
36. ДСТУ 7237:2011. СИСТЕМА СТАНДАРТІВ БЕЗПЕКИ ПРАЦІ ЕЛЕКТРОБЕЗПЕКА. На заміну на заміну ДСТУ IEC 60050-604:2004 ; чинний від 02.02.2011. Київ : ДЕРЖСТАНДАРТ УКРАЇНИ, 2011. 12 с. (дата звернення: 15.05.2025).
37. ДСТУ 7238:2011. СИСТЕМА СТАНДАРТІВ БЕЗПЕКИ ПРАЦІ. Чинний від 02.02.2011. Київ : ДЕРЖСТАНДАРТ УКРАЇНИ, 2011. 9 с. (дата звернення: 15.05.2025).

ДОДАТКИ

ДОДАТОК А. Фрагменти коду програми

Структура проекту

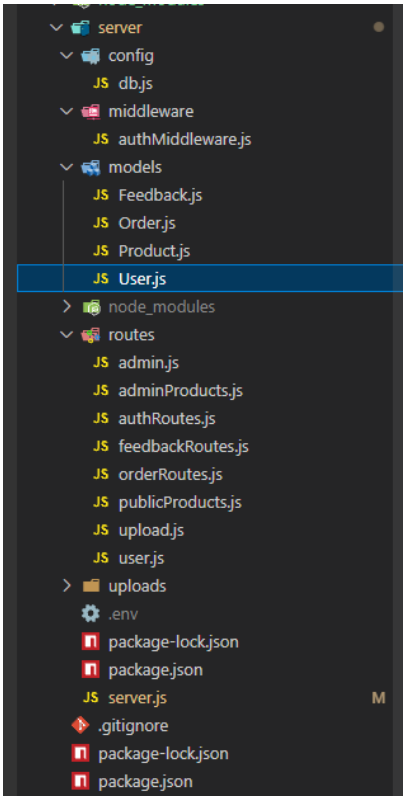


Рисунок А.1 – Структура backend проекту

Рисунок А.2 – Код authMiddleware.js

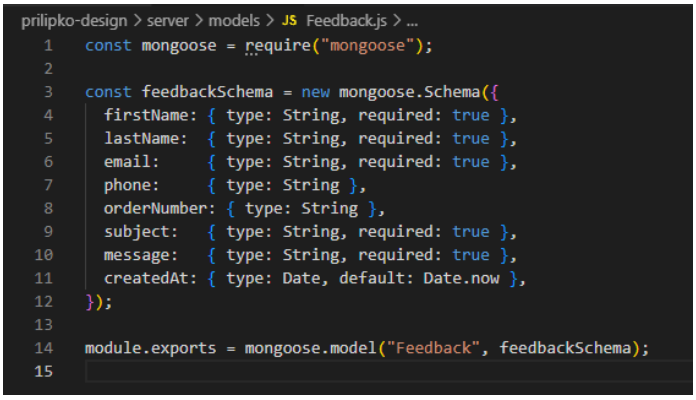


Рисунок А.3 – Код Feedback.js

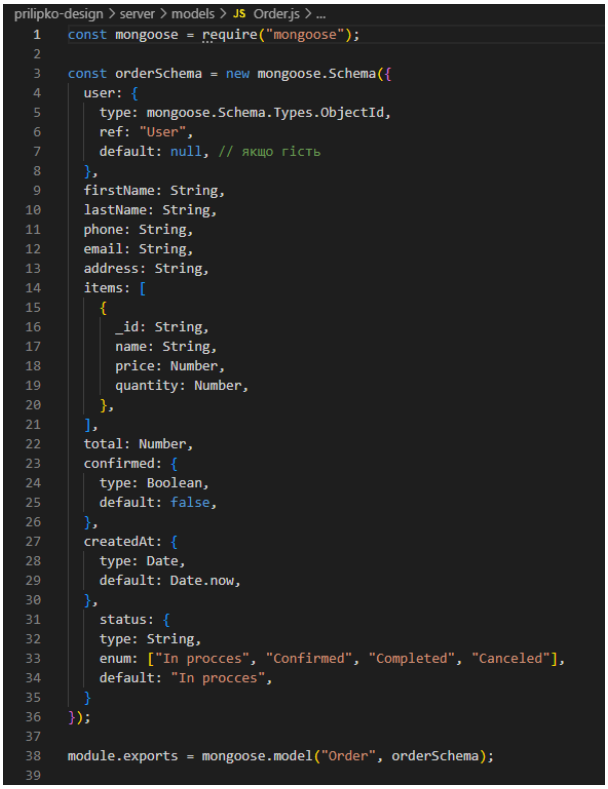


Рисунок А.4 – Код Order.js

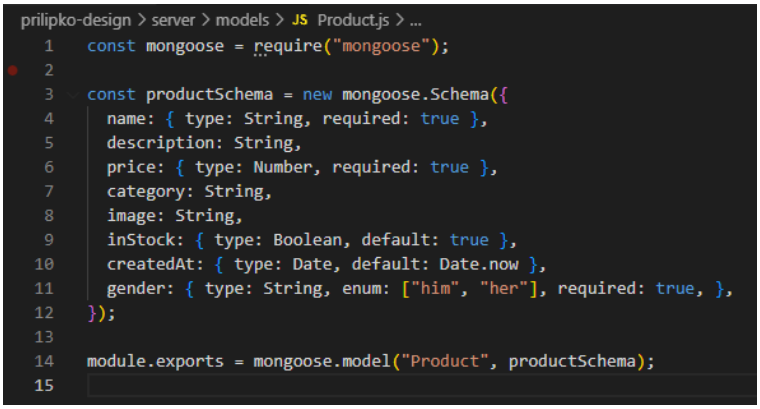
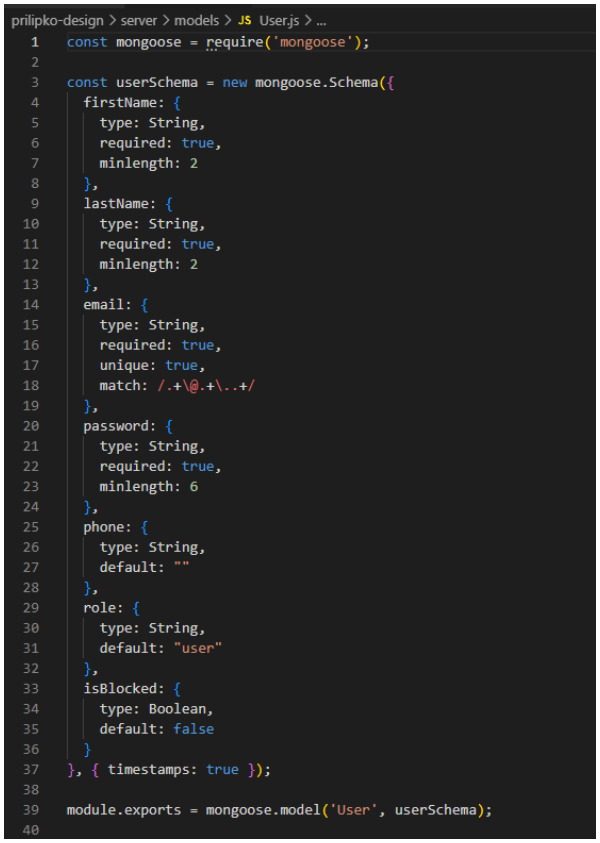
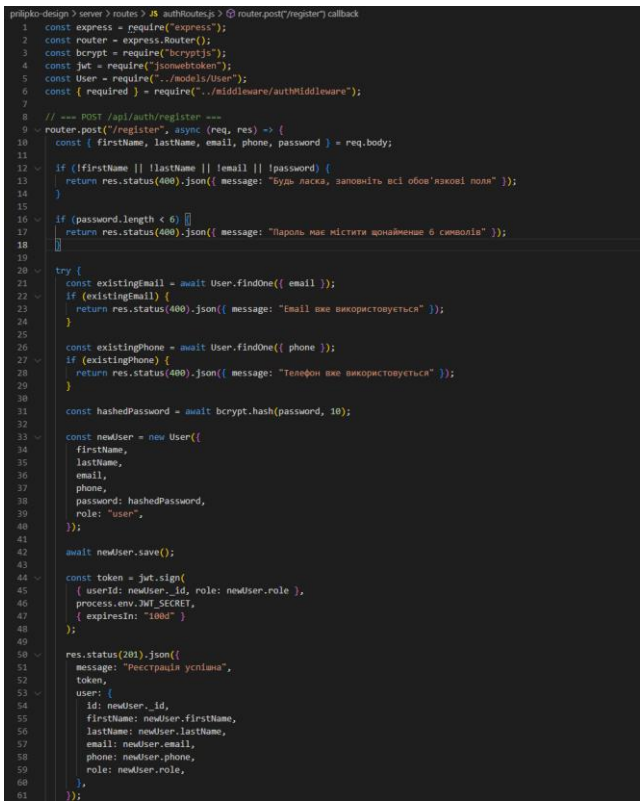


Рисунок А.5 – Код Product.js



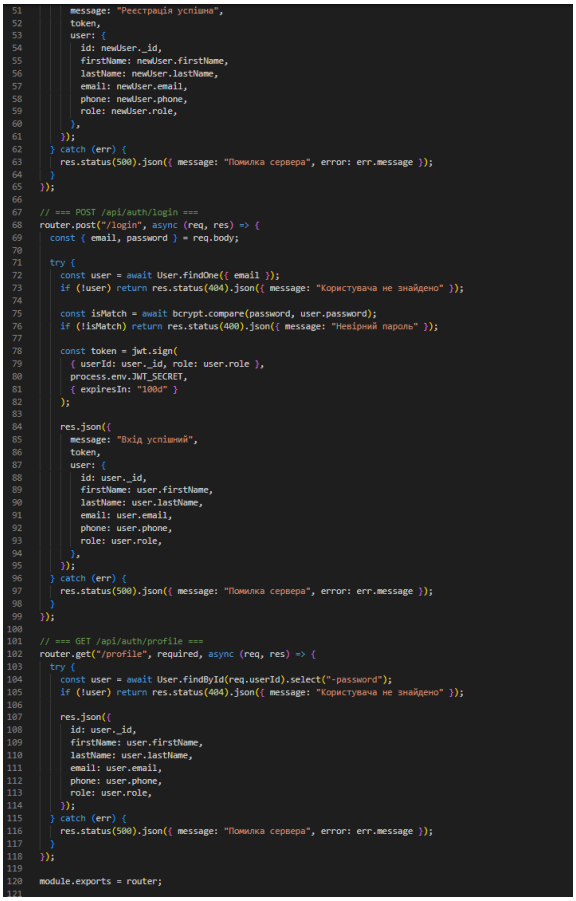
```
prilipko-design > server > models > JS User.js > ...
const mongoose = require('mongoose');

const userSchema = new mongoose.Schema({
  firstName: {
    type: String,
    required: true,
    minlength: 2
  },
  lastName: {
    type: String,
    required: true,
    minlength: 2
  },
  email: {
    type: String,
    required: true,
    unique: true,
    match: /.+\@.+\..+/
  },
  password: {
    type: String,
    required: true,
    minlength: 6
  },
  phone: {
    type: String,
    default: ""
  },
  role: {
    type: String,
    default: "user"
  },
  isBlocked: {
    type: Boolean,
    default: false
  }
}, { timestamps: true });

module.exports = mongoose.model('User', userSchema);
```

Рисунок А.6 – Код User.js



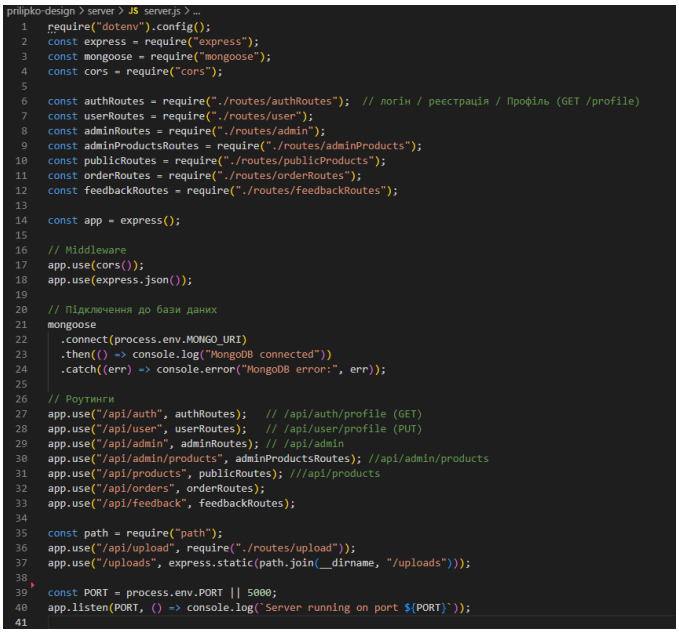
```
prilipko-design > server > routes > JS authRoutes.js > router.post("/register") callback
const express = require("express");
const router = express.Router();
const bcrypt = require("bcryptjs");
const jwt = require("jsonwebtoken");
const User = require("../models/User");
const { required } = require("../middleware/authMiddleware");

// === POST /api/auth/register ===
router.post("/register", async (req, res) => {
  const { firstName, lastName, email, phone, password } = req.body;

  if (!firstName || !lastName || !email || !password) {
    return res.status(400).json({ message: "Будь ласка, заповніть всі обов'язкові поля" });
  }

  if (password.length < 6) {
    return res.status(400).json({ message: "Пароль має містити щонайменше 6 символів" });
  }

  try {
    const existingEmail = await User.findOne({ email });
    if (existingEmail) {
      return res.status(400).json({ message: "Email вже використовується" });
    }

    const existingPhone = await User.findOne({ phone });
    if (existingPhone) {
      return res.status(400).json({ message: "Телефон вже використовується" });
    }

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

    const newUser = new User({
      firstName,
      lastName,
      email,
      phone,
      password: hashedPassword,
      role: "user",
    });

    await newUser.save();

    const token = jwt.sign(
      { userId: newUser._id, role: newUser.role },
      process.env.JWT_SECRET,
      { expiresIn: "100d" }
    );

    res.status(201).json({
      message: "Реєстрація успішна",
      token,
      user: {
        id: newUser._id,
        firstName: newUser.firstName,
        lastName: newUser.lastName,
        email: newUser.email,
        phone: newUser.phone,
        role: newUser.role,
      },
    });
```

Рисунок А.7 – Код authRoutes.js (частина 1)



```
      message: "Реєстрація успішна",
      token,
      user: {
        id: newUser._id,
        firstName: newUser.firstName,
        lastName: newUser.lastName,
        email: newUser.email,
        phone: newUser.phone,
        role: newUser.role,
      },
    });
  } catch (err) {
    res.status(500).json({ message: "Помилка сервера", error: err.message });
  }
});

// === POST /api/auth/login ===
router.post("/login", async (req, res) => {
  const { email, password } = req.body;

  try {
    const user = await User.findOne({ email });
    if (!user) return res.status(404).json({ message: "Користувача не знайдено" });

    const isMatch = await bcrypt.compare(password, user.password);
    if (!isMatch) return res.status(400).json({ message: "Невірний пароль" });

    const token = jwt.sign(
      { userId: user._id, role: user.role },
      process.env.JWT_SECRET,
      { expiresIn: "100d" }
    );

    res.json({
      message: "Вхід успішний",
      token,
      user: {
        id: user._id,
        firstName: user.firstName,
        lastName: user.lastName,
        email: user.email,
        phone: user.phone,
        role: user.role,
      },
    });
  } catch (err) {
    res.status(500).json({ message: "Помилка сервера", error: err.message });
  }
});

// === GET /api/auth/profile ===
router.get("/profile", required, async (req, res) => {
  try {
    const user = await User.findById(req.userId).select("-password");
    if (!user) return res.status(404).json({ message: "Користувача не знайдено" });

    res.json({
      id: user._id,
      firstName: user.firstName,
      lastName: user.lastName,
      email: user.email,
      phone: user.phone,
      role: user.role,
    });
  } catch (err) {
    res.status(500).json({ message: "Помилка сервера", error: err.message });
  }
});

module.exports = router;
```

Рисунок А.8 – Код authRoutes.js (частина 2)



```
pnlipko-design > server > JS server.js > ...
require("dotenv").config();
const express = require("express");
const mongoose = require("mongoose");
const cors = require("cors");

const authRoutes = require("./routes/authRoutes");  // логін / реєстрація / Профіль (GET /profile)
const userRoutes = require("./routes/user");
const adminRoutes = require("./routes/admin");
const adminProductsRoutes = require("./routes/adminProducts");
const publicRoutes = require("./routes/publicProducts");
const orderRoutes = require("./routes/orderRoutes");
const feedbackRoutes = require("./routes/feedbackRoutes");

const app = express();

// Middleware
app.use(cors());
app.use(express.json());

// Підключення до бази даних
mongoose
  .connect(process.env.MONGO_URI)
  .then(() => console.log("MongoDB connected"))
  .catch((err) => console.error("MongoDB error:", err));

// Роутинги
app.use("/api/auth", authRoutes);   // /api/auth/profile (GET)
app.use("/api/user", userRoutes);   // /api/user/profile (PUT)
app.use("/api/admin", adminRoutes); // /api/admin
app.use("/api/admin/products", adminProductsRoutes); //api/admin/products
app.use("/api/products", publicRoutes); ///api/products
app.use("/api/orders", orderRoutes);
app.use("/api/feedback", feedbackRoutes);

const path = require("path");
app.use("/api/upload", require("./routes/upload"));
app.use("/uploads", express.static(path.join(__dirname, "/uploads")));

const PORT = process.env.PORT || 5000;
app.listen(PORT, () => console.log(`Server running on port ${PORT}`));
```

Рисунок А.9 – Код server.js

ДОДАТОК Б. Елементи інтерфейсу користувача

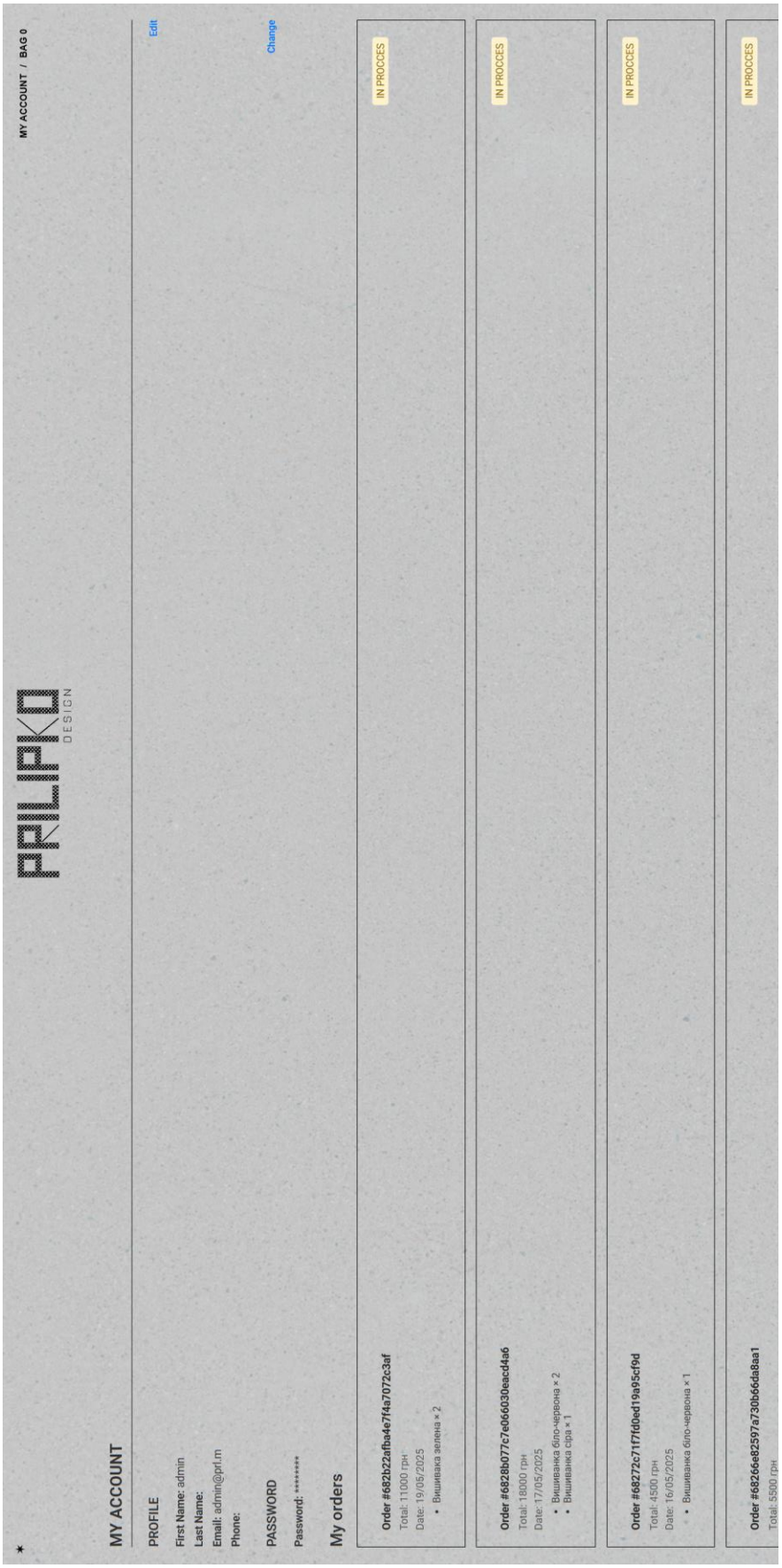


Рисунок Б.1 – Вкладка профіль користувача

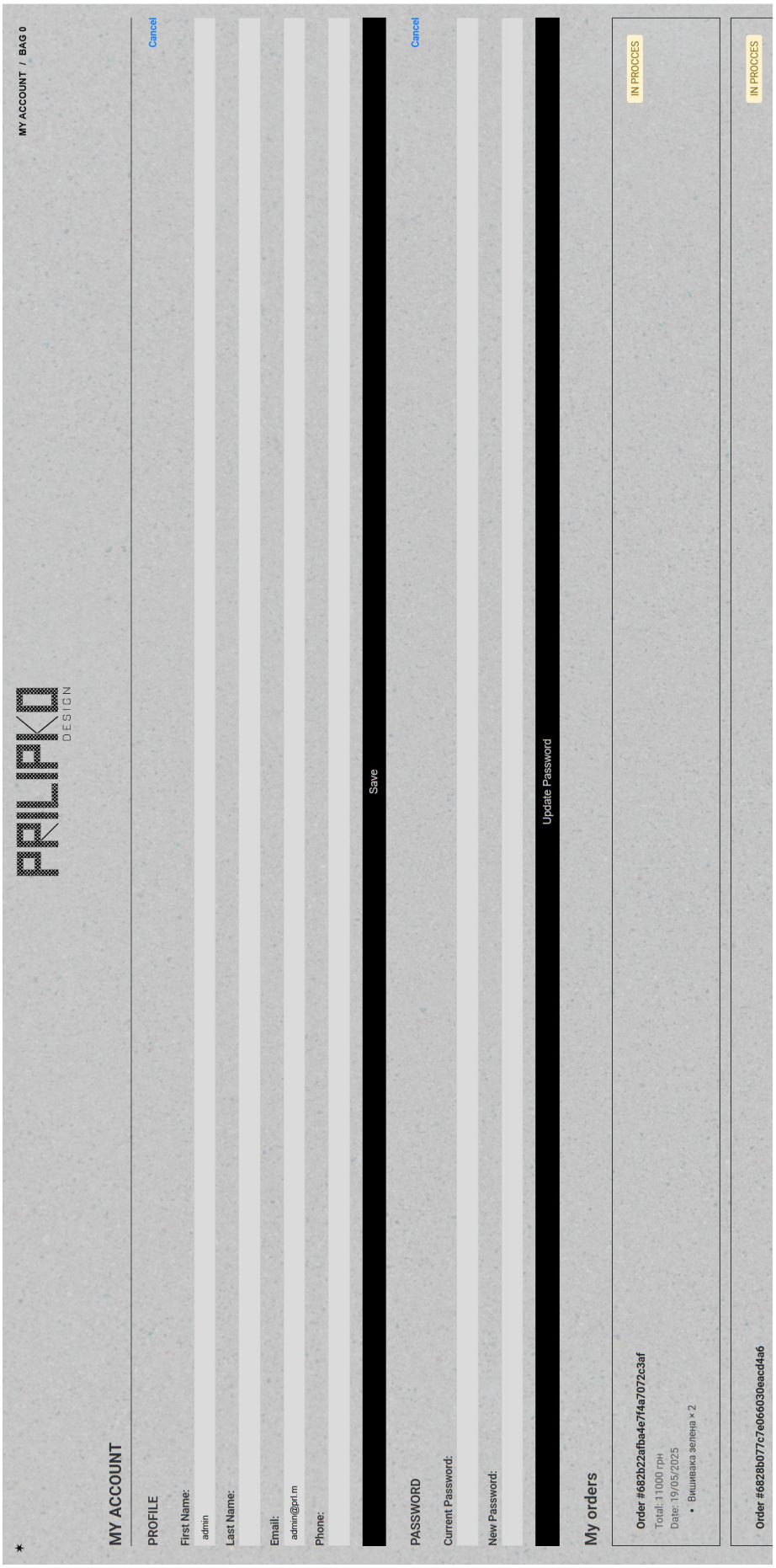


Рисунок Б.2 – Редагування даних користувача в профілі та перегляд замовлень

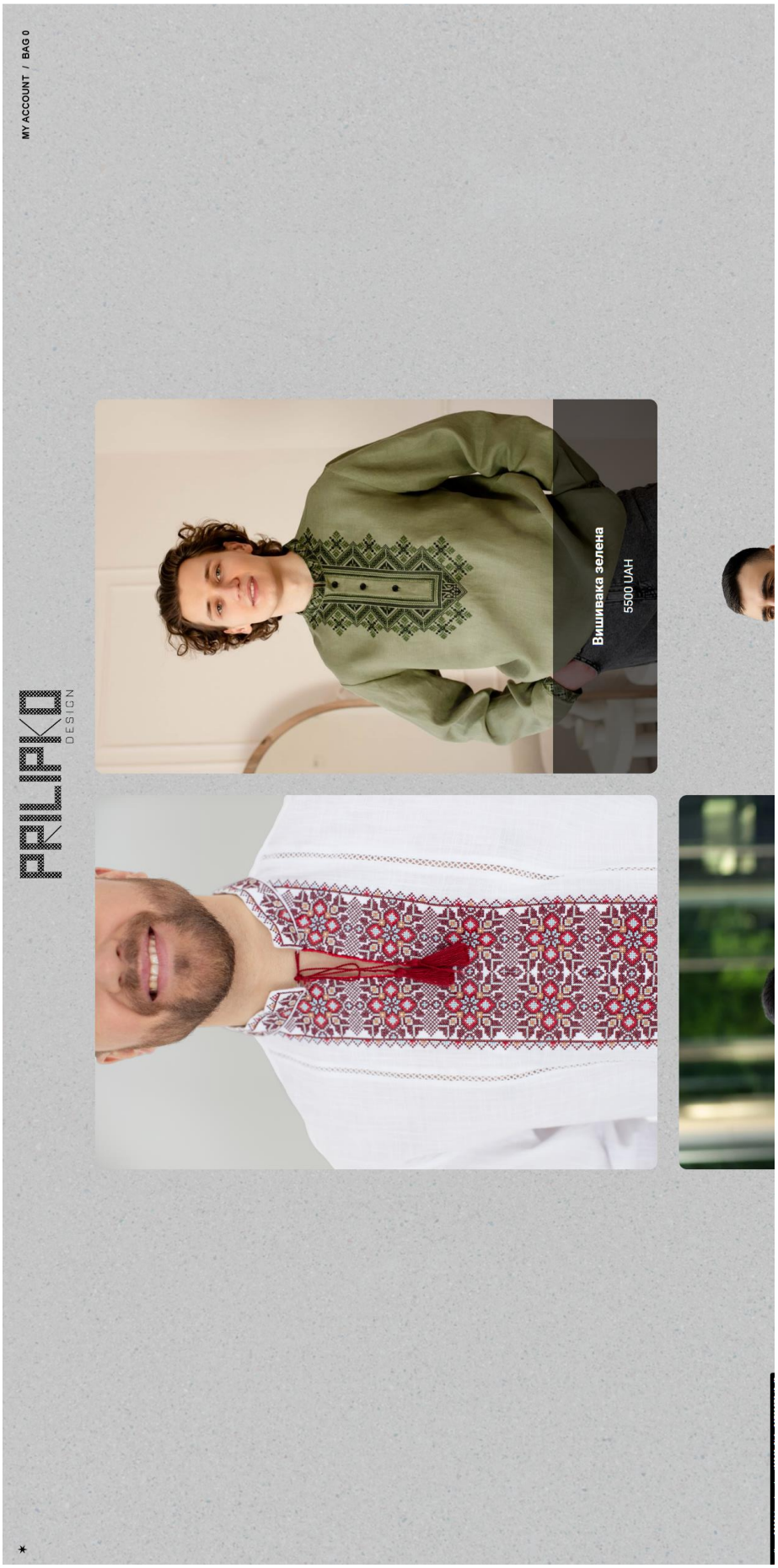


Рисунок Б.3 – Сторінка магазину (for him)

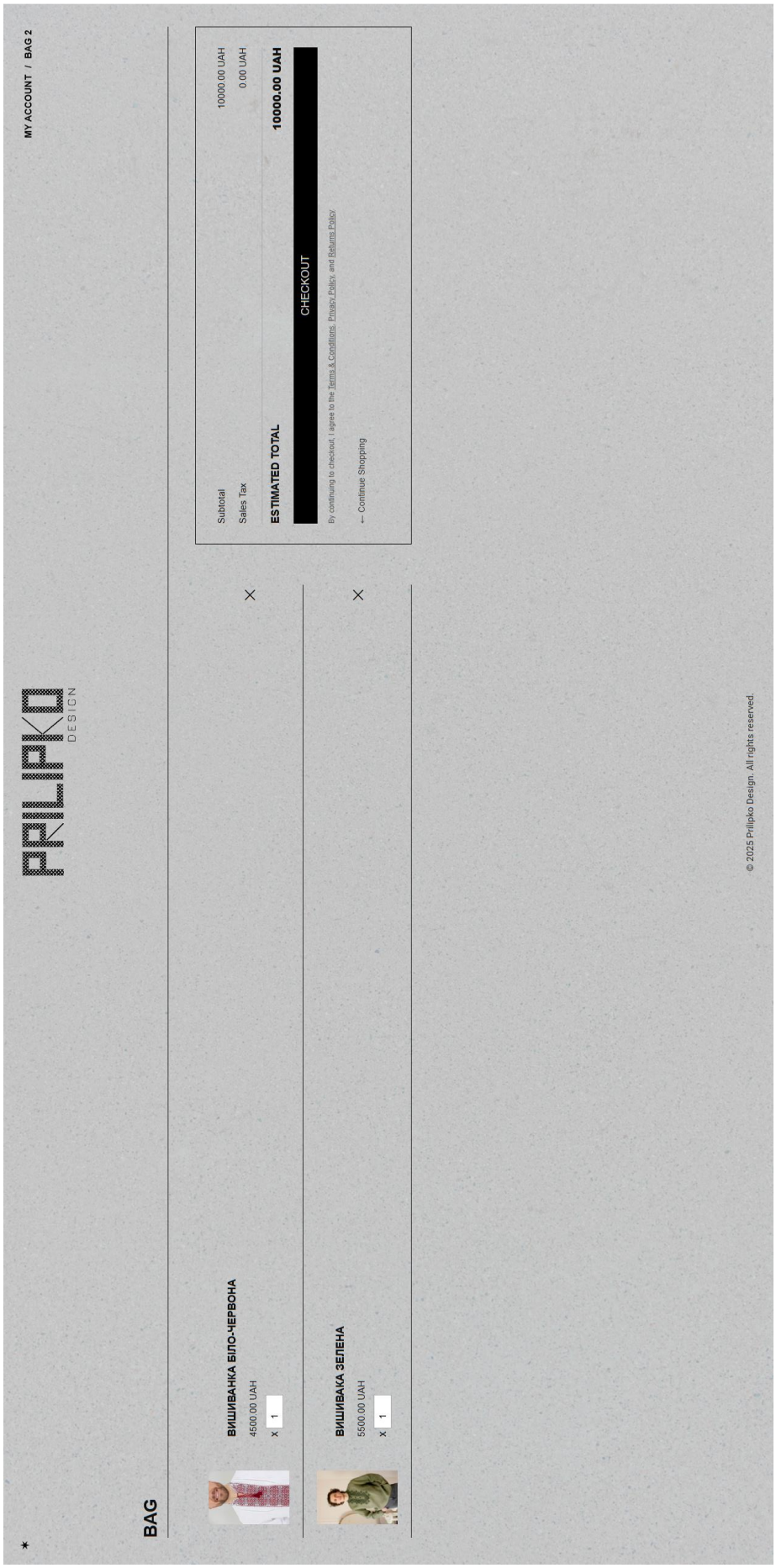


Рисунок Б.4 – Вкладка «Кошик» на сайті

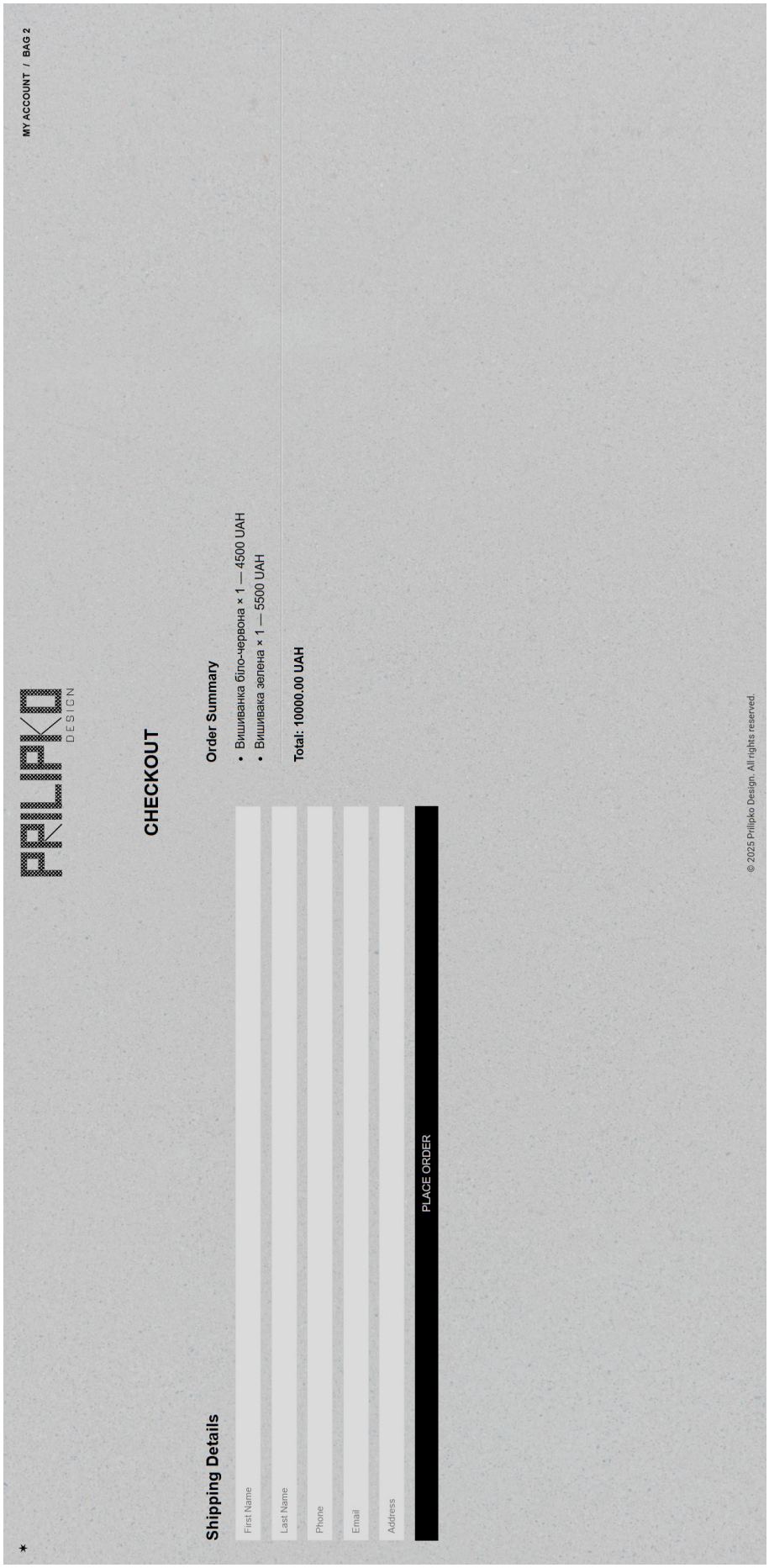


Рисунок Б.5 – Вкладка оформлення замовлення (checkout)

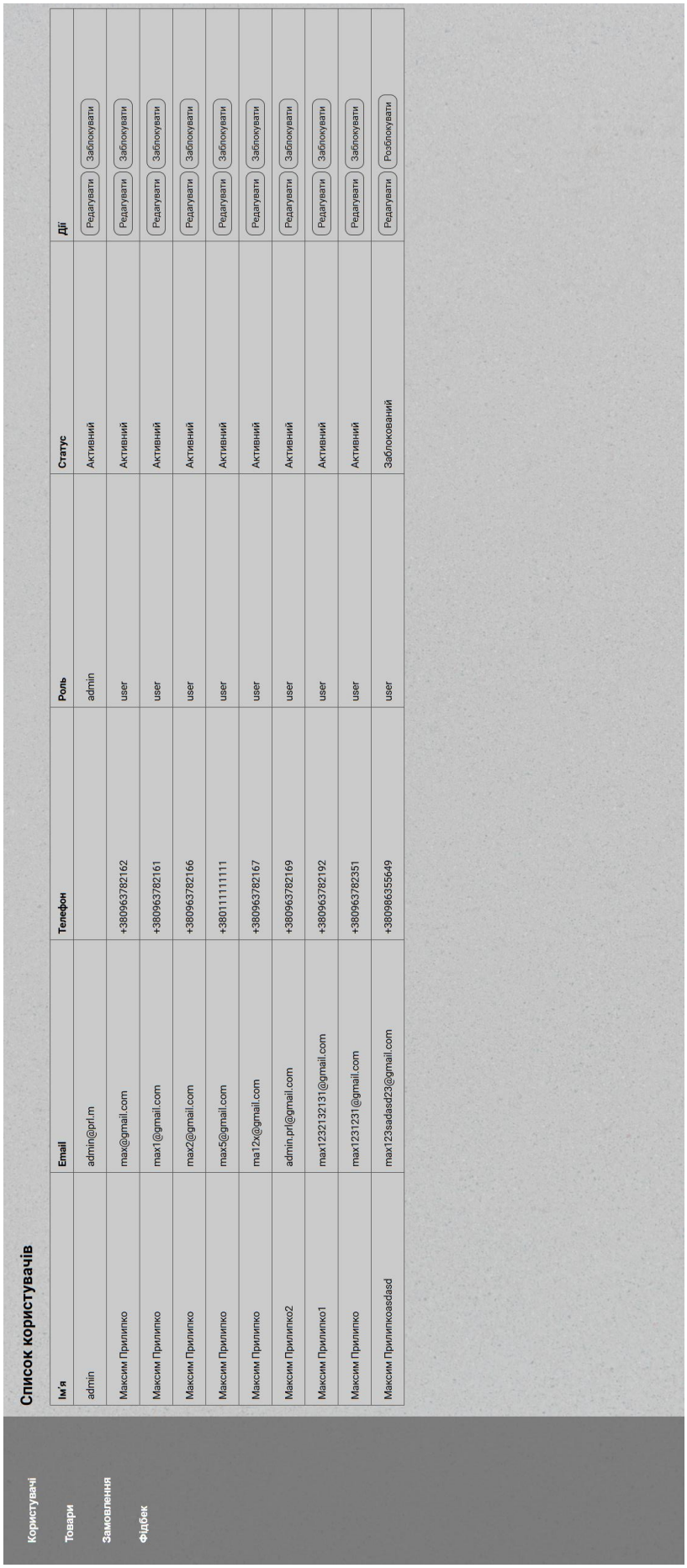


Рисунок Б.6 – Управління користувачами через адмін -панель

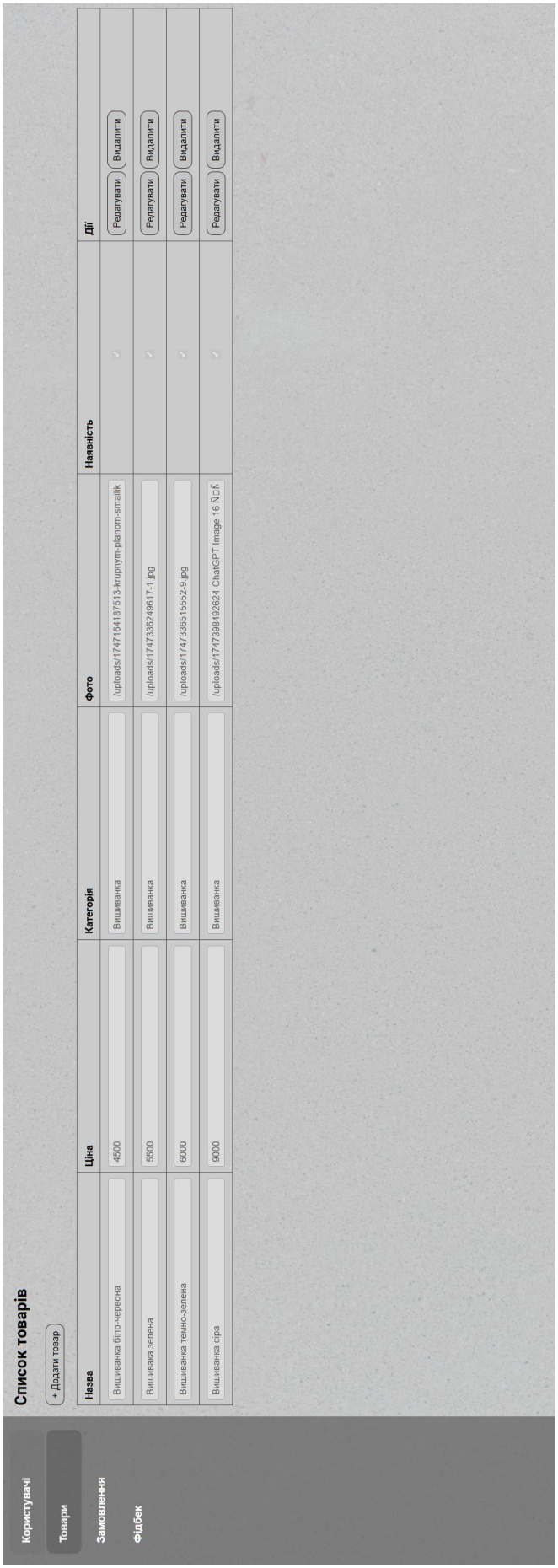


Рисунок Б.7 – Управління товарами через адмін -панель

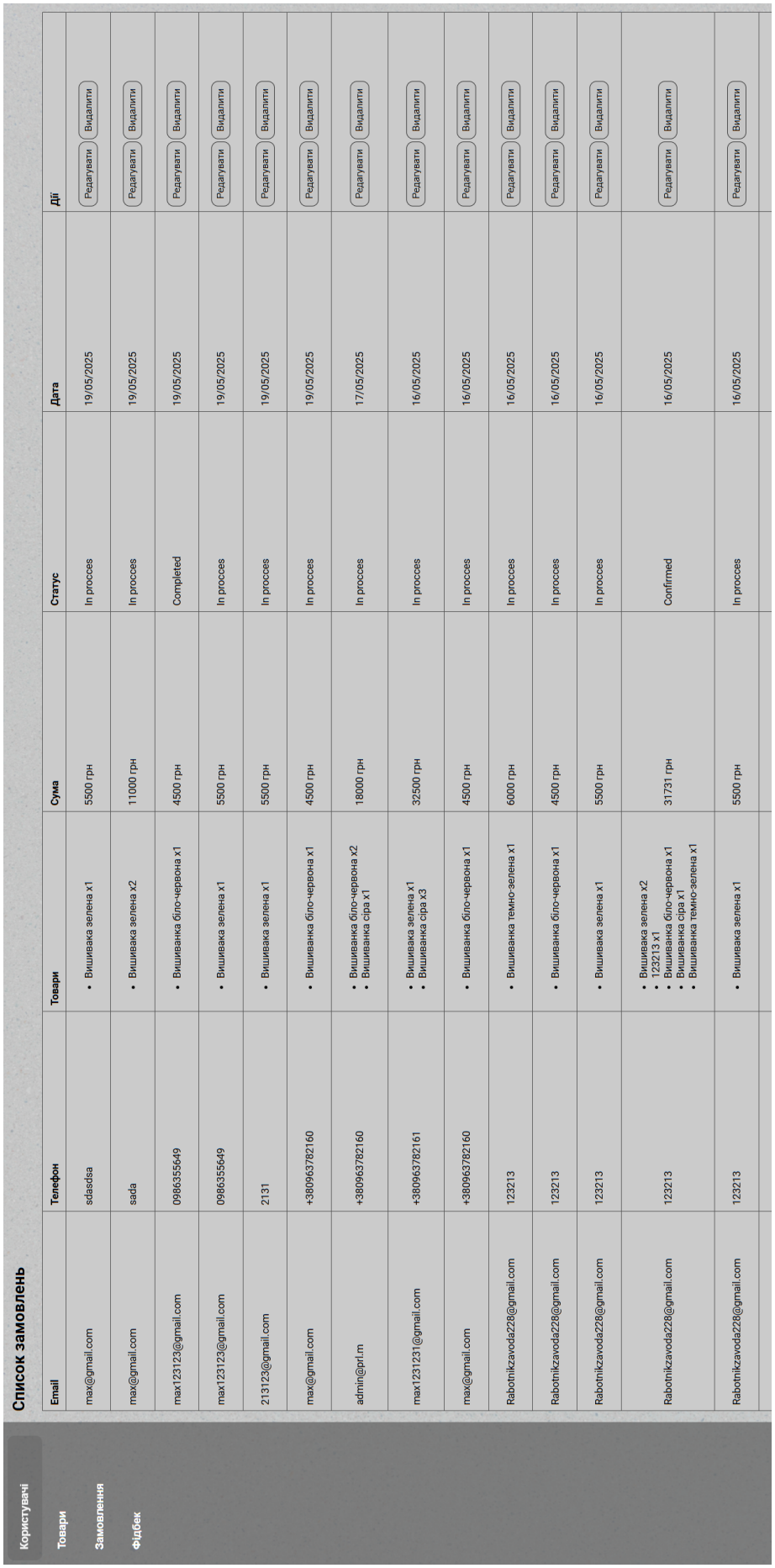


Рисунок Б.8 – Управління замовленнями через адмін -панель

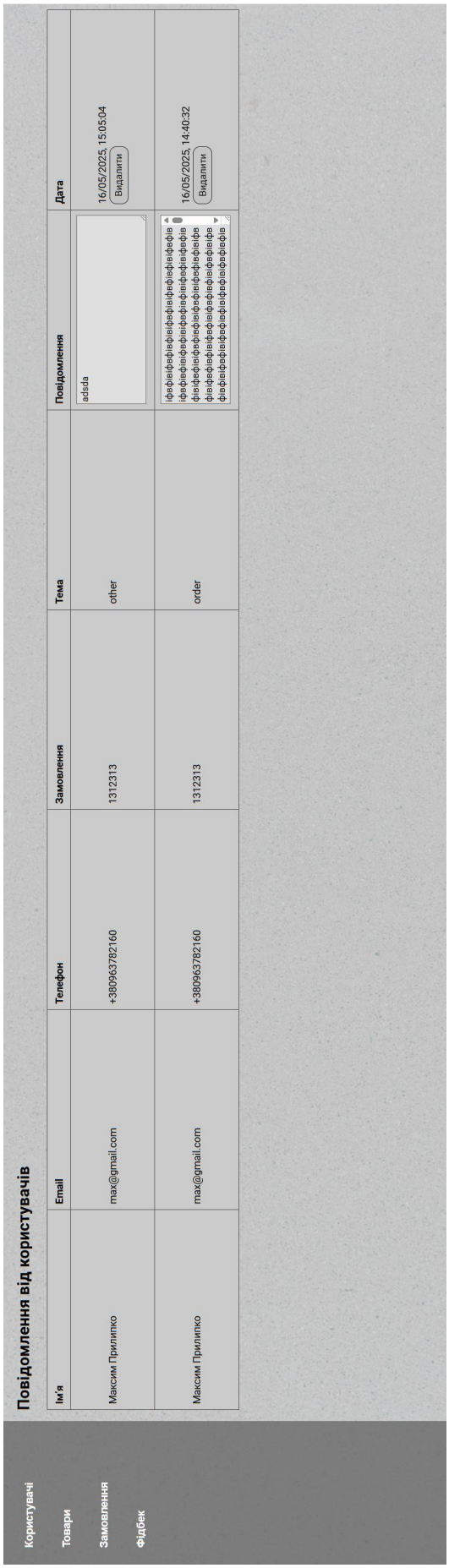


Рисунок Б.9 – Управління відгуками через адмін -панель