МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій

«До захисту допущено»
В.о. завідувача кафедри

_____ Світлана ВАЩЕНКО

____ _____ 2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 «Комп'ютерні науки»_____,

освітньо-професійної програми «Інформаційні технології проектування»_____

на тему: Вебдодаток підтримки діяльності магазину електротехнічних товарів побутового використання

Здобувача(ки) групи ІТ-01 Богулов Вадим Романович
(шифр групи) (прізвище, ім'я, по батькові)

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

_____ Вадим БОГУЛОВ
(підпис) (Ім'я та ПРІЗВИЩЕ здобувача)

Керівник к. т. н., доц. Яна ЧИБІРЯК _____
(посада, науковий ступінь, вчене звання, Ім'я та ПРІЗВИЩЕ) (підпис)

Суми – 2024

Сумський державний університет
Факультет електроніки та інформаційних технологій
Кафедра інформаційних технологій
Спеціальність 122 «Комп'ютерні науки»
Освітньо-професійна програма «Інформаційні технології проектування»

ЗАТВЕРДЖУЮ
В. о. зав. кафедри ІТ

_____Світлана ВАЩЕНКО
«____» _____ 2024 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА СТУДЕНТУ

Богулова ВадимаРомановича

1 Тема роботи *Вебдодаток підтримки діяльності магазину електротехнічних товарів побутового використання*

керівник роботи *Чибіряк Яна Іванівна, к.т.н., доцент*,

затверджені наказом по університету від « *07* » травня 2024 р. №0482-VI

2 Строк подання студентом роботи « 26 » травня 2024 р.
3 Вхідні дані до роботи

технічне завдання на розробку вебдодатку підтримки діяльності магазину електротехнічних товарів побутового використання

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

аналіз предметної області, проектування вебдодатку, програмна реалізація вебдодатку

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

_____

____ вступ, актуальність, постановка задачі, аналіз web-додатків, порівняння

____

сайтів-аналогів, моделювання web-додатку підтримки

____

діяльності діяльності магазину електротехнічних товарів побутового

____

використання, контекстна діаграма, діаграма декомпозиції,

____

діаграма варіантів використання, діаграма послідовності, практична реалізація

____

проекту, демонстрація web-додатку

____

6. Консультанти розділів роботи:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7.Дата видачі завдання ____08.02.2024_____

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Оформлення планування робіт	07.05.2024 - 09.05.2024	
2	Оформлення технічного завдання	10.05.2024 - 12.05.2024	
3	Проведення аналізу предметної області	13.05.2024 - 15.05.2024	
4	Проведення проектування вебдодатку	16.05.2024 - 18.05.2024	
5	Розробка та тестування вебдодатку	19.05.2024 - 21.05.2024	
6	Оформлення пояснювальної записки	22.05.2024 - 26.05.2024	

Студент _____ Вадим БОГУЛОВ
(підпис)

Керівник роботи _____ к. т. н., доц. Яна ЧИБІРЯК
(підпис)

АНОТАЦІЯ

Тема кваліфікаційної роботи бакалавра: «Вебдодаток підтримки діяльності магазину електротехнічних товарів побутового використання».

Пояснювальна записка складається зі вступу, трьох розділів, висновків, списку використаних джерел із 18 найменувань, додатків. Загальний обсяг роботи − 231 сторінки, у тому числі 48 сторінок основного тексту, 3 сторінки списку використаних джерел, 180 сторінок додатків.

Перший розділ присвячений аналізу предметної області та аналогів існуючих програмних продуктів, що підкреслює актуальність розробки вебдодатку для підтримки діяльності магазину електротехнічних товарів.

У другому розділі описано проектування вебдодатку, включаючи структурно-функціональне моделювання системи, вибір технологій та інструментів, таких як Node.js, Express.js та MongoDB, а також вимоги до інтерфейсу користувача та функціональності додатку.

Третій розділ охоплює програмну реалізацію вебдодатку, зокрема архітектуру Model-View-Controller (MVC), розробку компонентів системи, їх взаємодію, забезпечення безпеки.

Ключові слова: вебдодаток, електротехнічні товари, електронна комерція, онлайн-покупки, інтерфейс користувача, безпека, Node.js, Express.js, MongoDB.

ЗМІСТ

ВСТУП 7

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ 9

1.1 Огляд останніх досліджень і публікацій 9

1.2 Аналіз аналогів програмних продуктів 11

1.3 Мета та задачі проекту 18

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ 21

2.1 Структурно-функціональне моделювання 21

2.2 Моделювання варіантів використання web-додатку 22

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ 25

3.1 Архітектура web-додатку 25

3.2 Програмна реалізація 27

3.3 Використання програмного продукту 35

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 50

ДОДАТОК А 53

ДОДАТОК Б 71

ДОДАТОК В 83

ВСТУП

Сучасний розвиток технологій та постійні зміни в способах споживання товарів вимагають від підприємств постійної адаптації та вдосконалення своїх методів роботи. Одним із важливих напрямків цієї адаптації є використання інтернет-технологій та створення вебдодатків для підтримки діяльності підприємств. Вибір теми "Вебдодаток підтримки діяльності магазину електротехнічних товарів побутового використання" обумовлений актуальністю цієї проблеми в сучасному світі.

Актуальність цієї теми полягає у швидкому розвитку електронної комерції та зростанні популярності онлайн-покупок, зокрема в галузі електротехнічних товарів для побутового використання. Завдяки інтернет-магазинам та вебдодаткам, клієнти можуть здійснювати покупки зручно та ефективно, не виходячи з дому. Така модель роботи стає все більш популярною серед споживачів, що відкриває нові можливості для бізнесу.

Предметом нашого дослідження є створення вебдодатку, спрямованого на підтримку діяльності магазинів електротехнічних товарів побутового використання. Об’єктом дослідження є процес взаємодії магазину з клієнтами через вебдодаток, включаючи пошук та замовлення товарів, відстеження статусу замовлення та отримання зворотного зв'язку від клієнтів.

Метою роботи є створення вебдодатку для підтримки діяльності магазину електротехнічних товарів побутового використання, який надасть клієнтам зручний, швидкий і безпечний спосіб здійснення покупок. Для досягнення цієї мети виконання наступних ключових задач є необхідним:

– Проведення детального аналізу предметної області та огляд останніх публікацій, включаючи аналіз існуючих аналогів вебдодатків для визначення їх переваг і недоліків.

- Моделювання роботи вебдодатку та розробка його структури і функціоналу, що включає створення структурно-функціональних моделей та діаграм варіантів використання.
- Проектування та реалізація структури бази даних, серверної та клієнтської частин вебдодатку з використанням сучасних технологій, таких як Node.js і Express.js.
- Комплексне тестування вебдодатку для перевірки його функціональності, безпеки та продуктивності, оптимізація та налаштування хостингу на платформі Heroku.

Розроблений вебдодаток має бути ефективним і корисним як для магазину, так і для його клієнтів. Його впровадження має сприяти покращенню обслуговування клієнтів, збільшенню обсягів продажів та зміцненню конкурентоспроможності магазину на ринку електротехнічних товарів побутового використання.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Огляд останніх досліджень і публікацій

Останні дослідження та публікації у сфері електротехнічних товарів для побутового використання в Україні відображають значний ріст популярності цього сегмента [1]. Розмаїття та постійне оновлення асортименту вимагають нових підходів до підтримки діяльності магазинів, що спеціалізуються на цих товарах.

Крім того, у рамках підготовки даної роботи було опубліковано статтю, яка висвітлює розробку веб-додатка для підтримки діяльності магазину електротехнічних товарів побутового використання [2].

У 2022 році ринок електронної комерції в Україні став складним через початок повномасштабного вторгнення та зростання курсу валют, призвівши до підвищення цін на імпортні товари. Проте, купівельна активність відновлюється. Обсяги ринку в цьому сегменті зросли на 17% у 2023 році порівняно з попереднім роком. Найбільше зростання продажів спостерігається у сегменті побутової техніки.

За останніми дослідженнями, споживачі все більше віддають перевагу онлайн-покупкам [3]. Обсяг замовлень на популярних закордонних платформах, таких як AliExpress, зменшився майже на третину. Після падіння обсягів у 2022 році, онлайн-торгівля почала відновлюватися, а з нею зросла популярність інструментів економії. Українські інтернет-магазини відрізняються доступністю товарів та швидкою доставкою, що сприяє їхній подальшій популярності. З цього приводу, розробка вебдодатку для підтримки діяльності магазинів електротехнічних товарів стає надзвичайно актуальною, оскільки вона сприятиме зручності та доступності для клієнтів.

Дослідження також підкреслюють важливість підвищення рівня конкурентоспроможності для магазинів у цьому сегменті ринку [4]. Вебдодаток,

який забезпечує швидку і ефективну роботу магазину, може стати ключовим конкурентним перевагою, забезпечуючи задоволення потреб клієнтів.

Важливим аспектом досліджень є також аналіз споживчих тенденцій [5]. Вони показують, що сучасні покупці активно шукають способи ефективної та швидкої покупки товарів, у тому числі і в електротехнічному секторі. Більшість платежів зараз здійснюються цифрово.

У контексті економічних та промислових змін важливо враховувати розвиток промислової політики. Все більш очевидно, що підходи, де влада вирішує, які сектори економіки потребують більше уваги та ресурсів, не є успішними [6]. Для досягнення росту ефективності та конкурентоспроможності необхідний інший підхід - спрощення бізнес-процесів та підтримка інновацій. Вебдодаток, який автоматизує та оптимізує процеси продажу та обслуговування, стає ключовим інструментом для підвищення ефективності магазину електротехнічних товарів. Він дозволяє швидше реагувати на попит споживачів та поліпшувати взаємодію з клієнтами, що сприяє залученню та утриманню аудиторії в умовах зростаючої конкуренції.

Посилення цифрової трансформації в українському бізнесі відкриває широкі можливості для підтримки діяльності магазину електротехнічних товарів побутового використання через розробку спеціалізованого вебдодатку [7]. Такий додаток не лише відповідає сучасним вимогам, але й сприяє оптимізації бізнес-процесів, підвищенню ефективності і покращенню обслуговування клієнтів.

Дослідження вказують на потребу в удосконаленні взаємодії магазинів з клієнтами [8]. Щоб збільшити ефективність вебдодатка, можна розглянути використання психологічних методів впливу. Застосування методів нейролінгвістичного програмування (НЛП) може допомогти створити ефективні механізми комунікації з клієнтами, забезпечуючи їм зручні сервіси та підтримку. Наприклад, використання правильно підібраних слоганів, написів та закликів до дії в додатку може впливати на підсвідомість клієнтів, стимулюючи їх до покупки необхідних товарів. Також, оптимальне розташування товарів у додатку

та використання повторюваних елементів можуть сприяти покращенню продажів та задоволенню потреб клієнтів.

Велика увага також приділяється аналізу ефективності витрат та оптимізації бізнес-процесів [9]. Вебдодаток може сприяти зменшенню витрат на обслуговування та збільшенню ефективності продажів.

У контексті розробки вебдодатку для підтримки магазинів електротехнічних товарів, важливо враховувати сучасні технології [10]. Використання популярних інструментів, таких як Node.js для серверної частини, Express для створення вебдодатків та MongoDB для бази даних, може забезпечити ефективну та масштабовану розробку.

Застосування Node.js дозволяє розробникам швидко створювати ефективні та масштабовані серверні додатки [11]. Express, у свою чергу, надає простий та ефективний фреймворк для створення вебдодатків на основі Node.js [12]. MongoDB, як NoSQL база даних, забезпечує гнучку та масштабовану схему зберігання даних, що дозволяє ефективно управляти великим обсягом інформації [13].

Узагальнюючи, огляд останніх досліджень і публікацій підтверджує важливість розробки вебдодатку для підтримки магазинів електротехнічних товарів, який відповідає потребам ринку та сприяє їхньому успішному розвитку.

1.2 Аналіз аналогів програмних продуктів

Розробка вебдодатку для магазину електротехнічних товарів вимагає ретельного аналізу існуючих аналогів на ринку. З цією метою розглянуто характеристики кількох ведучих платформ: Amazon, eBay, Best Buy та Newegg, які з успіхом реалізують продаж електротехнічних товарів.

Amazon є однією з найбільших платформ електронної комерції, яка пропонує широкий спектр побутової електротехніки. Особливістю Amazon є

висока стандартизація процесів продажу та обслуговування клієнтів. Платформа забезпечує детальні описи продуктів, фотографії високої якості та відгуки користувачів, що допомагає покупцям ухвалювати обґрунтовані рішення. Система рекомендацій на основі історії покупок і переглядів також сприяє підвищенню продажів, адаптуючись під потреби кожного користувача [14]. Однак висока стандартизація процесів може призвести до втрати індивідуального підходу до клієнтів, а конкуренція з великою кількістю продавців на платформі може ускладнювати підтримку власного бренду.

eBay, у свою чергу, пропонує унікальну можливість покупки товарів через систему аукціонів, що може бути привабливим для бюджетних покупців. Однак аукціони можуть затягувати процес купівлі, що є недоліком для тих, хто шукає швидке обслуговування. eBay також дозволяє продавцям налаштовувати свої магазини, що забезпечує різноманітність товарів і цін [15]. Проте відсутність стандартизації серед продавців може призвести до різної якості товарів та обслуговування.

Best Buy фокусується на продажі електроніки та побутової техніки і пропонує консультативні послуги як онлайн, так і в фізичних магазинах. Ця платформа забезпечує додаткову вартість через персоналізовані послуги та має велику мережу магазинів, що дозволяє споживачам перевірити товар перед покупкою. Веб-сайт Best Buy легко навігується і містить розширені опції фільтрації, що допомагає клієнтам знаходити потрібні товари за специфікаціями [16]. Однак фокус на фізичні магазини може відволікати ресурси від онлайн-платформи, а доступність товарів може бути обмеженою залежно від географічного розташування.

Newegg спеціалізується на продажі комп’ютерної техніки та електроніки, пропонуючи детальні технічні специфікації та огляди від експертів, що є важливим для технічно обізнаних покупців. Платформа також пропонує конкурентоспроможні ціни та промоції, які спонукають до швидкого прийняття рішення про покупку. Проте сайт орієнтований переважно на технічно обізнаних користувачів, що може ускладнити навігацію для звичайних споживачів, а також

обмеженість асортименту може не задовольняти користувачів, які шукають товари не з області комп'ютерної техніки та електроніки.

Розглянувши особливості цих платформ, можна зазначити, що успіх вебдодатку залежить від його здатності забезпечити зручний і швидкий доступ до товарів, інтуїтивно зрозумілий інтерфейс, високу якість обслуговування та надійну систему підтримки. Для нового вебдодатка слід взяти до уваги такі основні компоненти, як інтерактивність, персоналізація, мобільна адаптивність та ефективна логістика. Важливо зосередити увагу на захисті даних користувачів, щоб забезпечити їхню довіру та лояльність.

Аналіз функціональності та дизайну вебсторінок великих онлайн-магазинів, таких як eBay, Amazon, Rozetka, та Allo, допомагає визначити ключові елементи, які можуть бути застосовані при створенні власного вебдодатка для продажу побутової електротехніки. Основна увага у цьому аналізі приділяється сторінці пошуку товарів, можливостям автентифікації користувачів, управлінню обліковими записами, системі кошика, перегляду історії замовлень, системі відгуків та процесу оплати.

Сторінка пошуку товарів на Amazon має високофункціональну систему фільтрації з багатьма параметрами, що дозволяє здійснювати детальний цілеспрямований пошук (рис. 1.1). Функції сортування та пагінації добре інтегровані, що забезпечує зручний перегляд результатів.

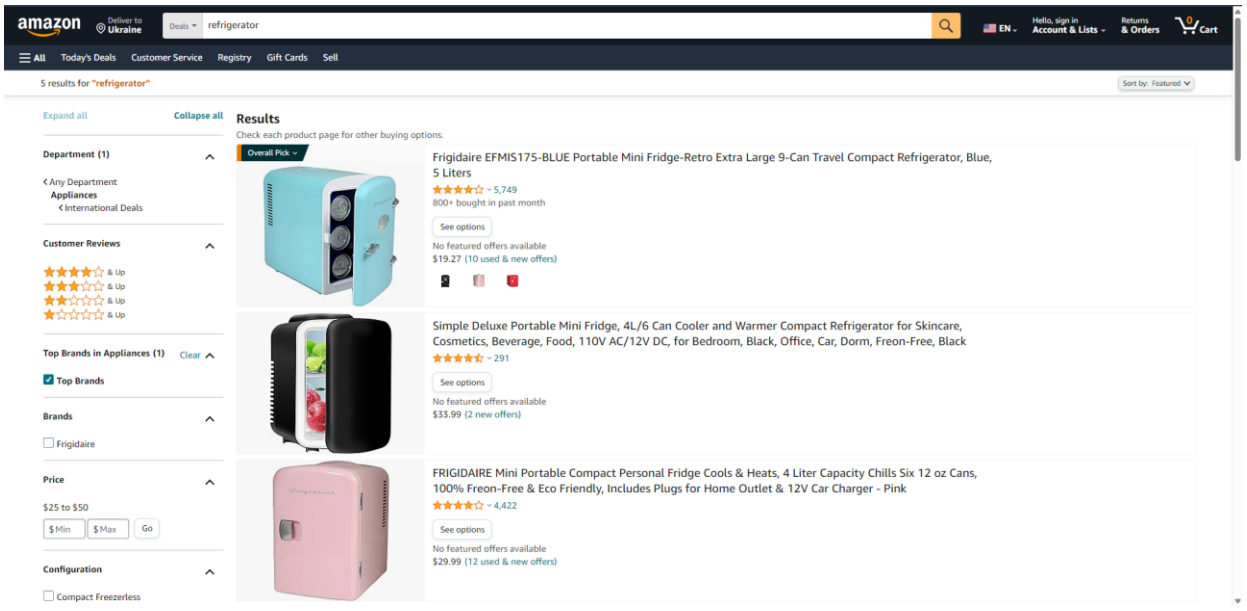

Рисунок 1.1 – сторорінка пошуку товарів на Amazon

eBay пошукова сторінка включає унікальні опції для аукціонів, а також стандартні фільтри по категоріях та цінах. Сортування може бути налаштоване за часом закінчення аукціону, що важливо для такої платформи (рис 1.2).

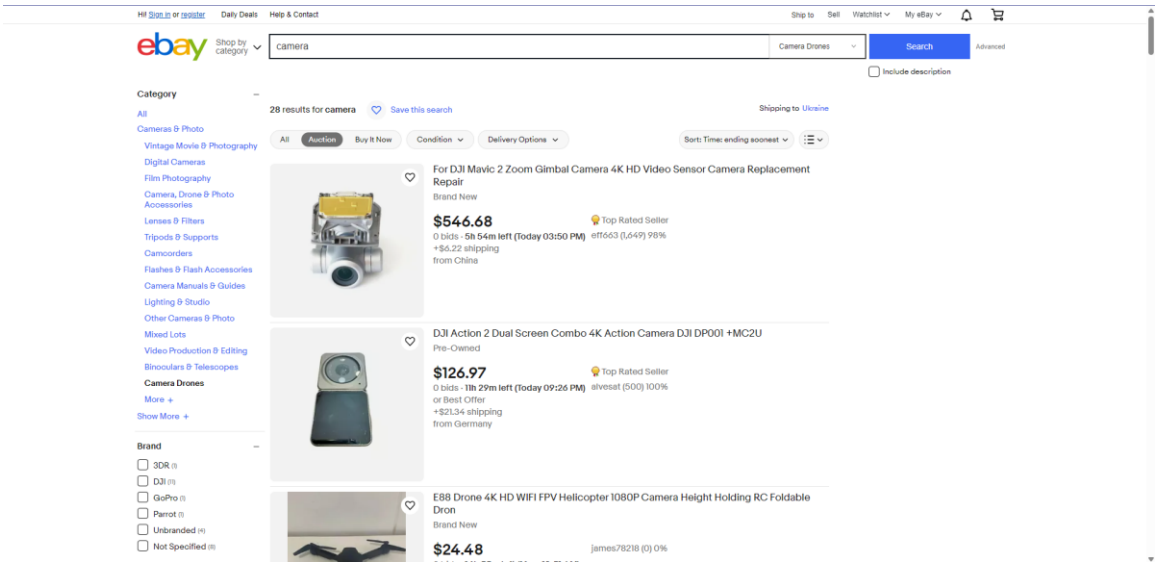

Рисунок 1.2 – Сторорінка пошуку товарів на eBay

Використання модальних вікон для автентифікації є поширеною практикою на багатьох сайтах, оскільки це забезпечує зручність для користувачів, які не хочуть переривати свій шопінг-процес. Наприклад, на сайті

Rozetka, модельне вікно автентифікації дозволяє швидко і легко увійти в свій обліковий запис, не покидаючи сторінку, на якій знаходиться користувач (рис. 1.3).

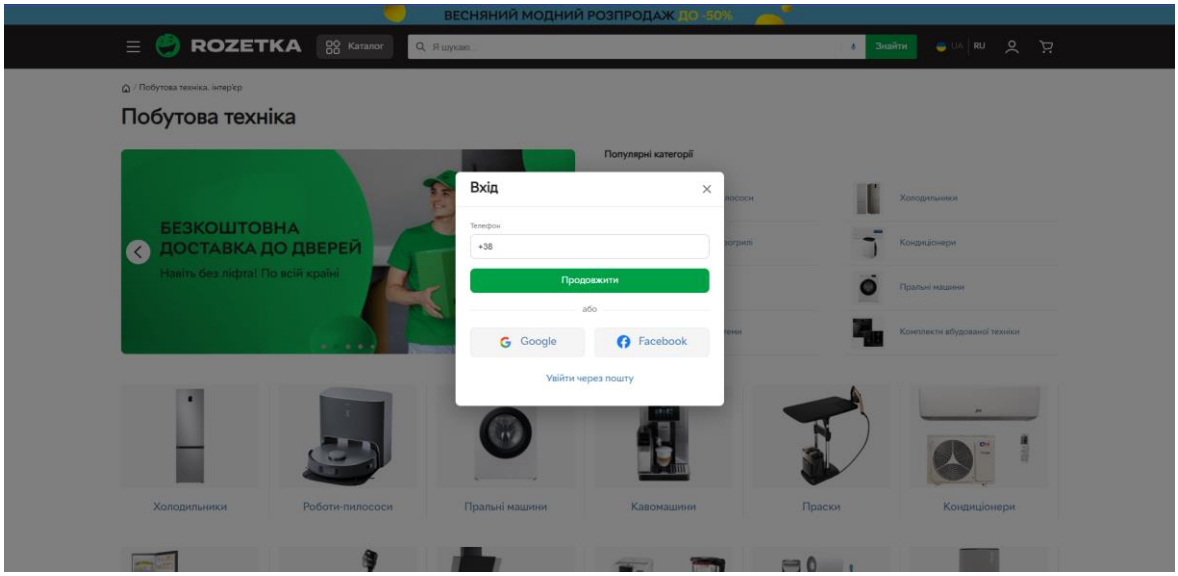

Рисунок 1.3 – Модельне вікно автентифікації на Rozetka

На рисунку 1.4 зображена сторінка кошика обраних товарів на веб-сайті Allo. Такі системи дозволяють зручно додавати і видаляти товари, а також контролювати вартість замовлення перед оплатою. Історія замовлень, доступна через обліковий запис, надає користувачам інформацію про їхні минулі покупки та статус поточних замовлень.

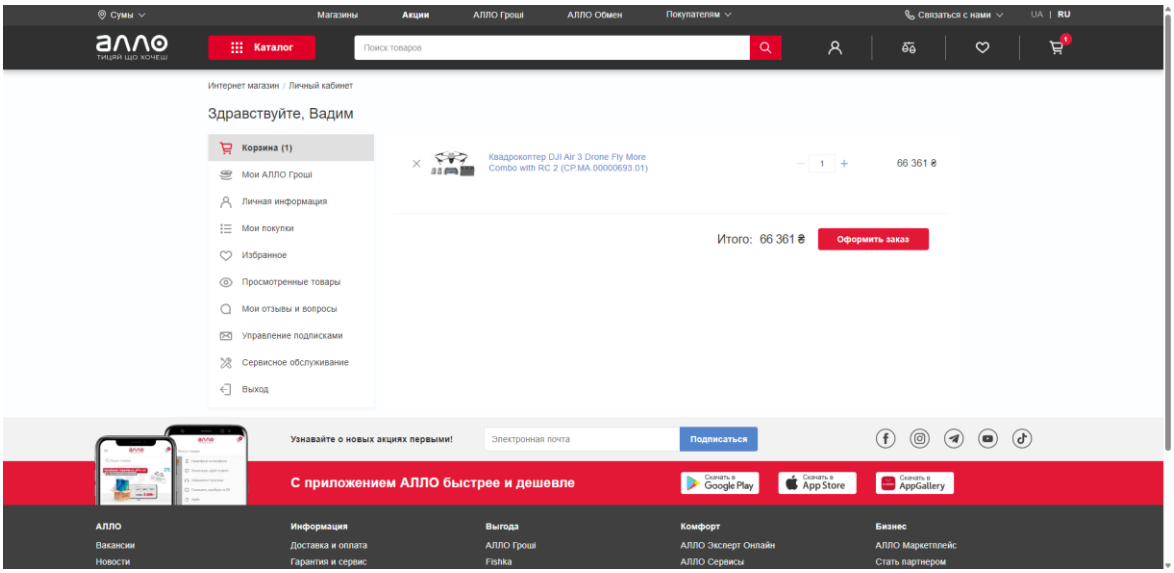

Рисунок 1.4 – Сторінка кошику обраних товарів на Allo

Система відгуків, аналогічна тій, що є на Amazon, є ключовою функцією, що сприяє підвищенню довіри та задоволеності клієнтів. Вона розташована на сторінці товару, нижче його опису, що робить її легкою у використанні та доступною (рис. 1.5).

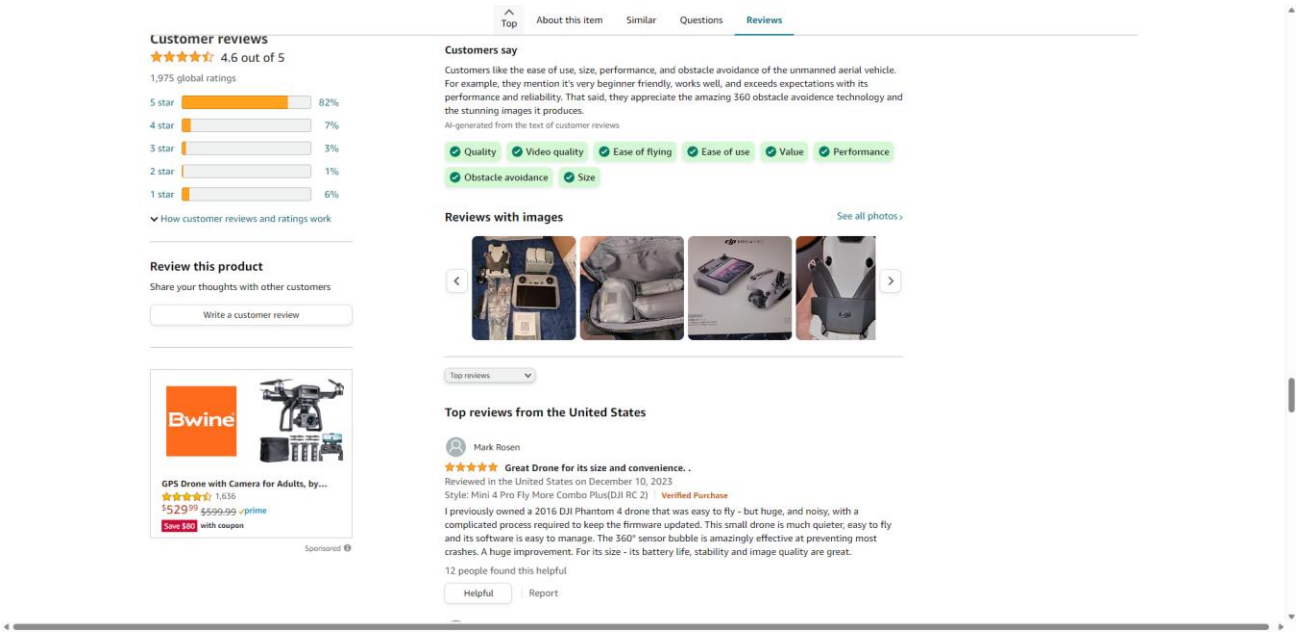

Рисунок 1.5 – Система відгуків на Amazon

Сторінка оплати повинна бути захищеною, інтуїтивно зрозумілою та надавати кілька опцій оплати (кредитні картки, PayPal, інші електронні гаманці), що збільшує зручність користувачів та їхню готовність до покупки. Застосування цих аналізів дозволить створити вебдодаток, який не тільки наслідує успішні практики, але й вводить інноваційні функції для підвищення унікальності та конкурентоздатності на ринку.

Складено таблицю в якій представляємо аналіз функціональних особливостей кожного з вибраних онлайн-магазинів: Amazon, eBay, Rozetka, Allo, а також власного сайту з назвою «VB Shop». Кожен з цих сайтів було оцінено за наявністю (+) або відсутністю (-) певних функцій.

Таблиця 1.1 - Аналіз функціональності провідних онлайн-магазинів

Функціонал	Amazon	eBay	Rozetka	Allo	VB Shop
Сторінка блогу	-	-	+	-	+
Власні марки продуктів	+	-	-	-	+
Сторінка товару	+	+	+	+	+
Система відгуків	+	+	+	+	+
Кошик користувача	+	+	+	+	+
Перегляд історії замовлень	+	+	+	-	+
Модальне вікно авторизації	+	+	+	+	+
Оновлення профілю користувача	+	+	+	+	+
Адмін панель для управління запасами	+	+	+	-	+

На основі аналізу провідних онлайн-магазинів, у власному розроблюваному проекті планується реалізувати наступний набір функціональних можливостей. Сторінка блогу дозволить користувачам отримувати актуальну інформацію та спілкуватися. Сторінка пошуку

забезпечить можливість швидкого та зручного пошуку продуктів за ключовими словами чи категоріями. Детальний огляд товару включатиме фотографії, характеристики та відгуки користувачів. Система відгуків надасть можливість залишати оцінки та коментарі для товарів. Кошик користувача забезпечить зручне збереження товарів перед замовленням. Перегляд історії замовлень дозволить контролювати та відстежувати статус замовлень. Процес оплати буде безпечним та зручним для користувачів. Модальне вікно авторизації забезпечить безпеку та конфіденційність даних користувачів. Оновлення профілю користувача надасть можливість зміни та оновлення особистої інформації. Адмін панель дозволить керувати запасами товарів та іншими аспектами магазину. Також буде можливість представлення власних брендів і продуктів у магазині.

1.3 Мета та задачі проекту

Метою проекту є створення інтуїтивно зрозумілого та ефективного вебдодатку, призначеного для підтримки продажу електротехнічних товарів побутового використання. Основна задача додатку – забезпечити користувачам зручний інтерфейс для швидкого пошуку та придбання потрібних товарів, що сприяє значній економії часу.

Основні завдання, що мають бути виконані для реалізації цього проекту, включають:

– Розробку структурно-функціональної моделі вебдодатку, що дозволить визначити ключові компоненти системи та їх взаємозв'язки, забезпечуючи ефективну підтримку операцій магазину.

– Моделювання варіантів використання вебдодатку, яке включає розробку сценаріїв користування різними функціями додатку, щоб забезпечити високий рівень зручності для клієнтів при здійсненні покупок.

– Впровадження архітектури вебдодатку, орієнтованої на масштабованість та високу продуктивність, що допоможе забезпечити стабільність роботи при великій кількості користувачів та транзакцій.

– Реалізацію компонентів системи, бази даних та заходів безпеки, щоб створити надійну основу для зберігання і обробки даних, а також захистити інформацію та приватність користувачів.

– Проведення комплексного тестування всіх аспектів вебдодатку, включаючи перевірку функціональності, безпеки та продуктивності, а також оцінку зручності використання готового продукту для забезпечення його надійності та ефективності у відповідності до потреб магазину електротехнічних товарів.

Усі вимоги та специфікації детально описані в технічному завданні, яке включено до додатку А.

Технології, що розглядаються для розробки проекту, включають Node.js [17] для серверної частини, Express.js [18] для маршрутизації та обробки запитів, та MongoDB [19] як систему управління базами даних, оскільки ці інструменти забезпечують високу продуктивність, швидкість обробки великих обсягів даних і є добре підтримуваними в індустрії.

На основі потреб користувача, включаючи відвідувачів та персоналу (адміністраторів), визначені наступні функціональні вимоги:

1 Можливість перегляду сторінок: головної, блогу, про додаток та контактної інформації.

2 Можливість пошуку товарів за категоріями та сортуванням.

3 Можливість перегляду детальної інформації про товари, включаючи відгуки.

4 Необхідність логіну та реєстрації користувача для доступу до деяких функцій.

5 Можливість оновлення особистих даних користувача, включаючи зображення профіля.

6 Можливість створення та видалення відгуків на товари.

7 Можливість додавання товарів в особистий кошик.

8 Можливість редагування списку товарів в кошику.

9 Можливість перегляду вартості замовлення перед оформленням.

10 Можливість перегляду адміністративної панелі та управління товарами в адміністративній панелі для адміністраторів.

11 Можливість оплати обраних товарів.

12 Можливість перегляду історії замовлень.

2 ПРОЕКТУВАННЯ WEB-ДОДАТКУ

2.1 Структурно-функціональне моделювання

Діаграми, представлені далі, демонструють взаємодію між різними компонентами системи, виділяючи елементи входу, управління, механізмів та виходу на різних рівнях декомпозиції.

Контекстна діаграма IDEF0 (рис. 2.1) описує загальну структуру вебдодатку. Вхідні дані, такі як користувацькі запити, поступають через ліву кромку до бази даних та веб-сервера. Елементи управління, що включають політики безпеки, конфігураційні налаштування та функціонал додатку, впливають на систему зверху, направляючи процеси та робочі потоки. Механізми, які включають базу даних, веб-сервер та апаратне забезпечення, знаходяться в нижній частині діаграми та виконують основну обробку даних. Вихідні дані відправляються через праву кромку у вигляді замовленого товару.

Рисунок 2.1 – Контекстна діаграма IDEF0

Діаграма першого рівня декомпозиції (рис. 2.2) деталізує внутрішню взаємодію між процесами в рамках системи, зокрема у процесах авторизації

користувачів, пошуку та оброблення товарів. Тут також чітко вказано напрямки: вхідні потоки (зліва), керувальні потоки (зверху), механізми, як-от база даних, веб-сервер та апаратне забезпечення (знизу), та вихідні потоки (справа), що включають оброблені запити та замовлення.

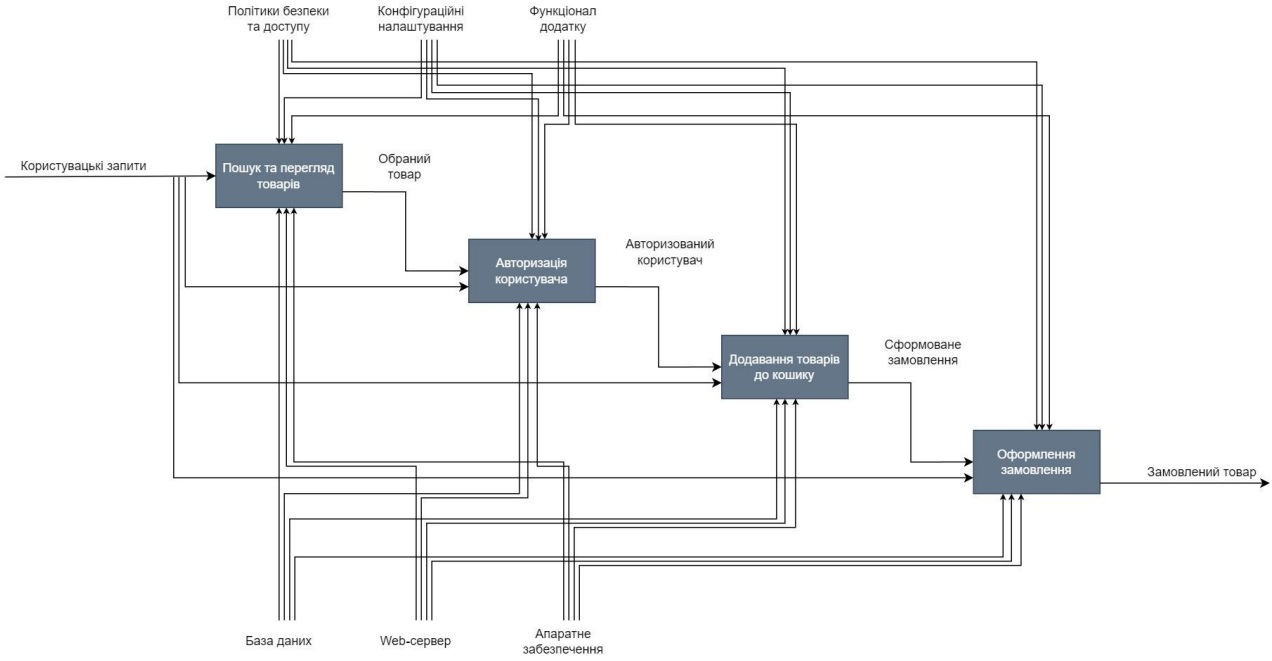

Рисунок 2.2 – Діаграма першого рівня декомпозиції

Ці діаграми є ключовими для глибокого розуміння структурних та функціональних аспектів вебдодатку, надаючи розробникам засоби для ефективного проектування та аналізу системи.

2.2 Моделювання варіантів використання web-додатку

Діаграма, представлена на рисунку 2.3, є моделлю варіантів використання для вебдодатку, що забезпечує платформу для діяльності інтернет-магазину електротехнічних товарів побутового використання. На цій діаграмі представлено три основні ролі користувачів: гість, авторизований користувач і

адміністратор, кожна з яких має свої відповідні можливості та взаємодії з системою.

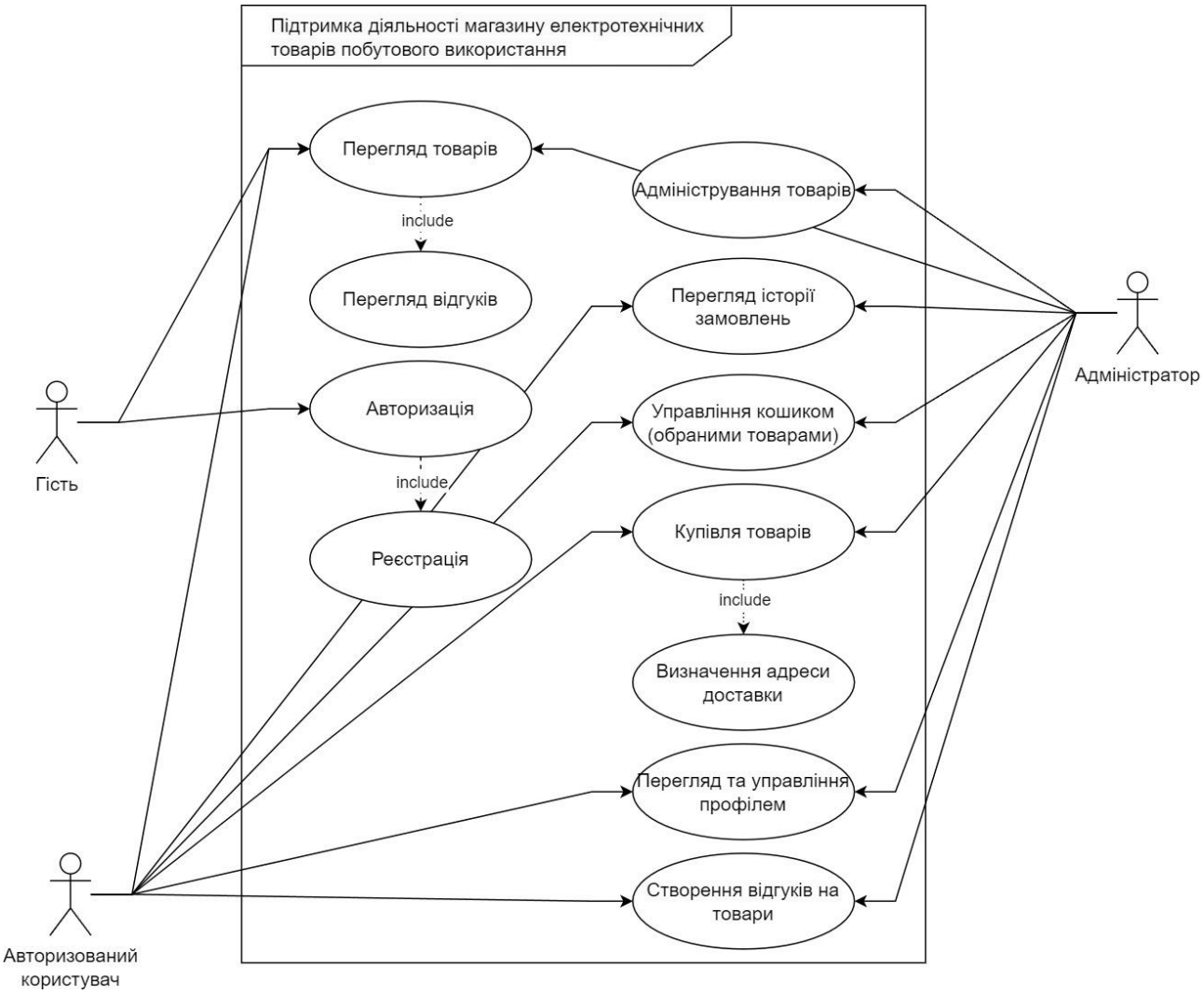

Рисунок 2.3 – Діаграма варіантів використання

- Гість має можливість реєстрації та авторизації на платформі, що дозволяє йому переходити до статусу авторизованого користувача. Він також може переглядати товари і переходити до перегляду деталей товарів.
- Авторизований користувач нарощує свої можливості в системі, включаючи купівлю товарів, визначення адреси доставки та створення

відгуків на товари. Він також може управляти своїм кошиком, що включає додавання товарів до кошика та перегляд його вмісту.

– Адміністратор відповідає за адміністрування товарів, що охоплює додавання, видалення та модифікацію інформації про товари. Він також взаємодіє з іншими користувачами через управління системою і може переглядати різні аспекти функціонування вебдодатку.

Ця діаграма варіантів використання є ключовою для розуміння структури взаємодій між користувачами та системою, надаючи чітке уявлення про різні доступні дії залежно від ролі користувача у системі вебдодатку.

Таблиця 2.2 — Опис варіантів використання додатку

№	Актор	Опис
1	Гість	Неавторизований користувач, може переглядати товари, реєструватися та авторизуватися в системі.
2	Авторизований користувач	Користувач з реєстрацією, може купувати товари, управляти кошиком, оформлювати замовлення, створювати відгуки.
3	Адміністратор	Управляє інформацією про товари, відповідає за адміністрацію вмісту та функціональності системи.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ДОДАТКУ

3.1 Архітектура web-додатку

Вебдодаток розроблено з використанням архітектури Model-View-Controller (MVC), яка сприяє ефективному розділенню відповідальностей, підвищує можливості масштабування та спрощує процес тестування.

1. Model (Модель):

Модель визначає структуру даних, логіку обробки та правила бізнес-логіки. Вона відповідає за зберігання даних та їх взаємодію з контролером. Моделі в MVC не тільки відображають структуру бази даних, але й можуть включати методи для обробки даних (наприклад, валідація, обрахунок тощо).

2. View (Вид):

Вид відповідає за представлення даних користувачу. Це може бути будь-який вивід відповідно до запиту в браузері, такий як HTML-сторінка або JSON для веб-API. Вид забезпечує шаблони або компоненти для відображення інтерфейсу користувача, залежно від отриманих з моделі даних.

3. Controller (Контролер):

Контролер виконує роль посередника між моделлю та видом. Він обробляє вхідні запити від користувача, взаємодіє з моделлю для отримання даних або внесення змін, та направляє ці дані до відповідного виду для відображення. Контролер може також здійснювати додаткову обробку даних, необхідну перед передачею їх до виду.

На рисунку 3.1 зображено архітектуру додатку.

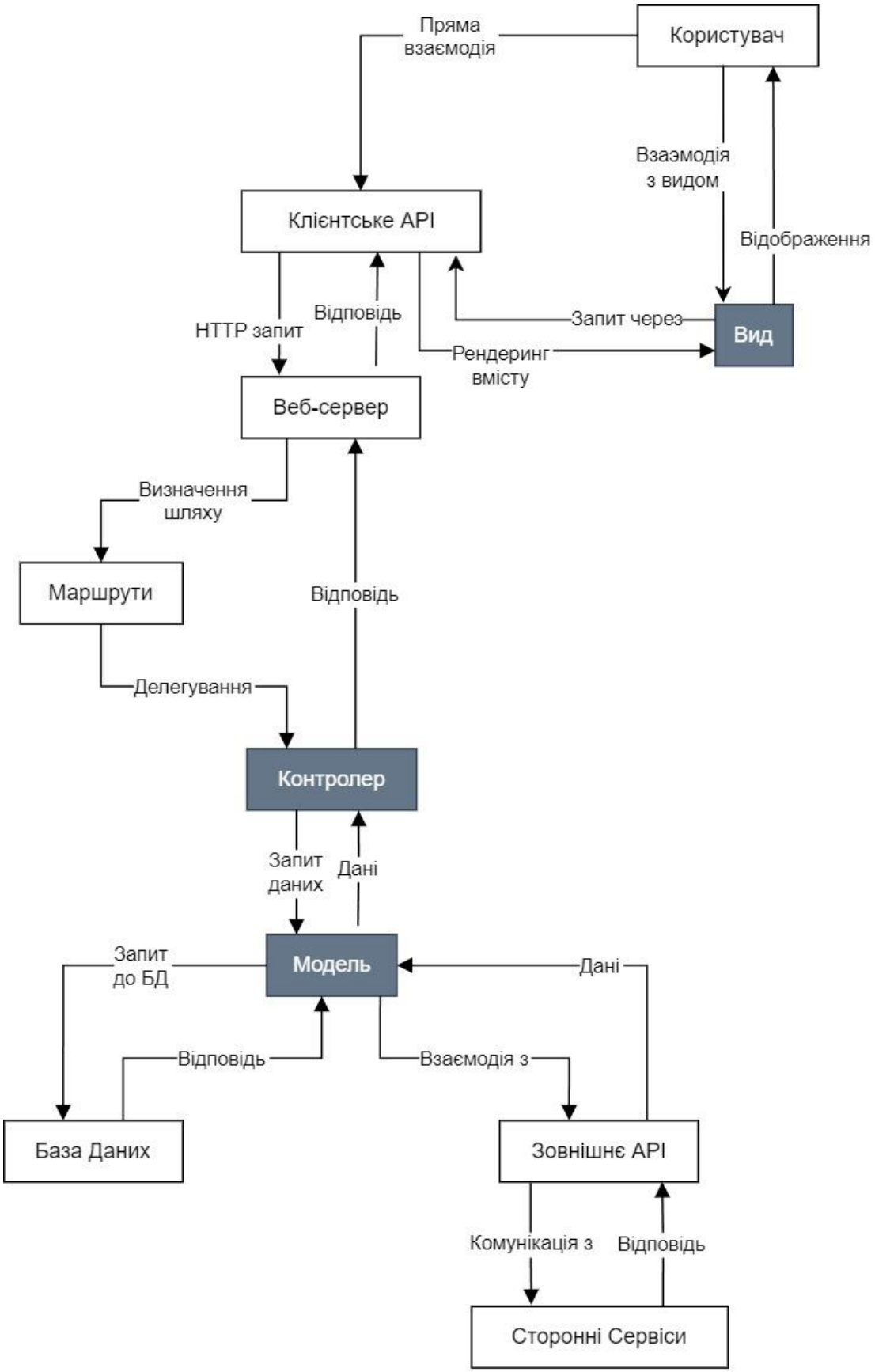

Рисунок 3.1 – Архітектура додатку

Діаграма представляє розширену архітектуру вебдодатку, що включає клієнтський інтерфейс (вид), клієнтське API, веб-сервер, маршрутизацію, контролери, модель, базу даних та взаємодію зі сторонніми API та сервісами.

Користувач має можливість взаємодіяти з додатком двома шляхами: напряму через клієнтське API або опосередковано через інтерфейс користувача. Інтерфейс користувача рендерить контент та забезпечує користувацькі взаємодії, що відправляють запити до клієнтського API.

Клієнтське API обробляє вхідні запити та перенаправляє їх через веб-сервер, який використовує маршрутизацію для визначення шляхів та делегування відповідних запитів контролерам. Контролери приймають ці запити та взаємодіють з моделлю, що виконує бізнес-логіку та операції з даними. Модель, у свою чергу, може запитувати дані з локальної бази даних або здійснювати запити до сторонніх сервісів через зовнішні API.

Отримані дані повертаються моделлю до контролера, який формує відповідь і відправляє її через веб-сервер назад до клієнтського API. Заключний етап полягає в рендерингу отриманих даних в користувацькому інтерфейсі для відображення користувачу.

Ця архітектура підтримує гнучкість у взаємодії між клієнтом та сервером, а також інтеграцію різноманітних зовнішніх сервісів, що забезпечує багатошарову та модульну структуру додатку.

3.2 Програмна реалізація

3.2.1 Компоненти системи та їх взаємодія

Для детального розуміння архітектури вебдодатку важливо вивчити не тільки структурні компоненти системи, але й механізми їх взаємодії. Сіквенс діаграма надає візуальне представлення процесів обробки даних та комунікації

між компонентами системи в реальному часі, що відіграє ключову роль у проектуванні та реалізації ефективних та надійних вебдодатків.

На представленій діаграмі (рис. 3.2) відображено взаємодію між основними компонентами системи: користувачем, маршрутизатором, контролером, моделлю, базою даних та відображенням (видом). Ця діаграма ілюструє, як запит користувача проходить через систему, від ініціації до отримання відповіді.

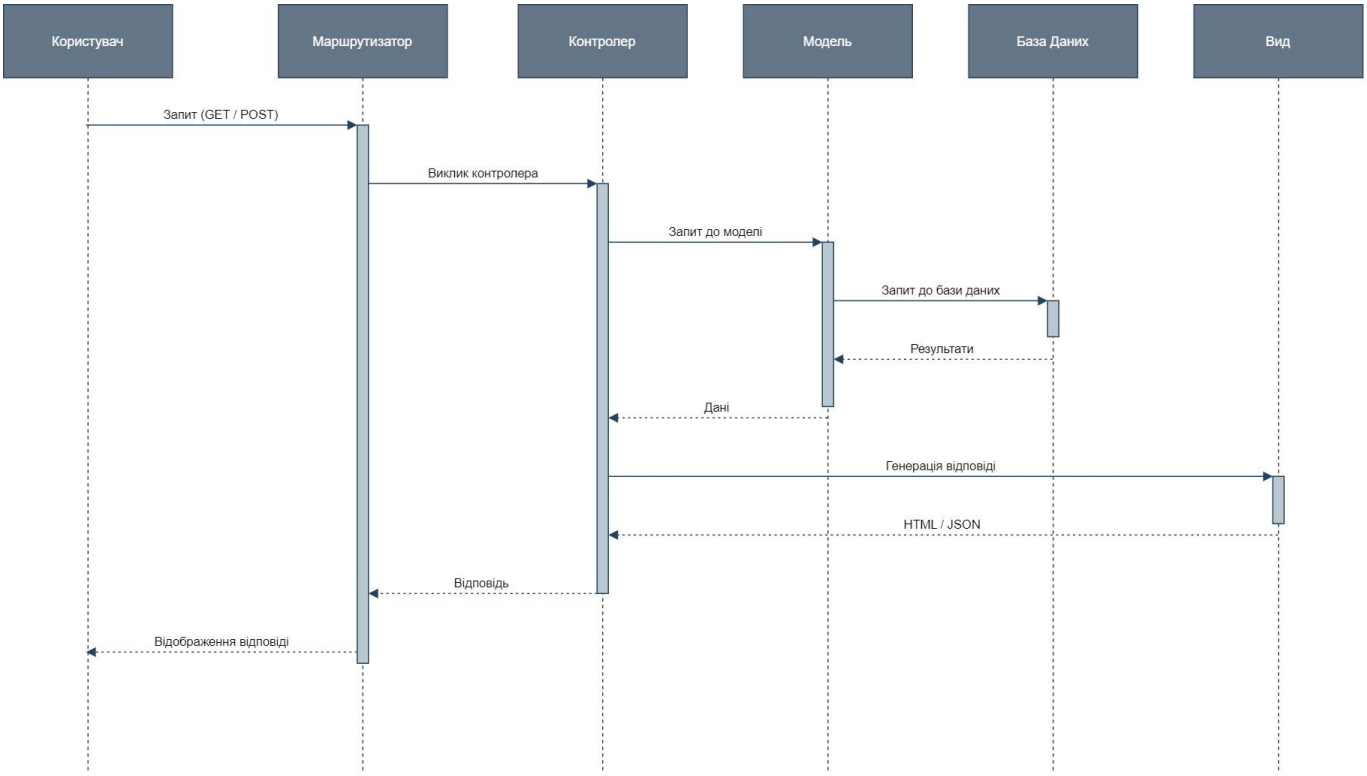

Рисунок 3.2 – Діаграма послідовності додатку

1 Головні файли конфігурації та файли конфігурації проекту відіграють критичну роль в організації та запуску Node.js додатків. Ці файли визначають як сервер повинен запускатися, які зовнішні залежності потрібні для роботи додатку, та як повинен оброблятися запит.

app.js є основним файлом додатку, який конфігурує сервер та середовище Express. У ньому визначаються:

– Підключення необхідних модулів.

- Налаштування шляхів для статичних файлів і шаблонів відображення.
- Конфігурація обробки тіла запиту та кукі.
- Налаштування middleware для компресії, захисту від атак і обмеження кількості запитів.
- Визначення маршрутів для користувацького інтерфейсу та API.
- Обробка помилок і відправлення відповідних відповідей.

Файл server.js ініціалізує запуск сервера. Він включає:

- Завантаження змінних середовища через dotenv.
- Встановлення обробників для невиловлених винятків та відхилених промісів.
- Підключення до бази даних MongoDB.
- Старт сервера на визначеному порту.
- Завершення роботи сервера при отриманні сигналу SIGTERM.

package.json - це маніфест проекту, який використовується npm для ідентифікації проекту та управління його залежностями. У ньому зазначено:

- Інформацію про проект: назву, версію, опис, автора.
- Скрипти для запуску середовища розробки, продакшну, зборки проекту, та інші команди.
- Залежності: перелік пакетів необхідних для запуску додатку (dependencies) та для розробки (devDependencies).
- Версію Node.js, яка використовується в проекті (engines).

2 Контролери виконують роль посередника між моделями даних і відображеннями, керуючи логікою обробки запитів користувачів. Ось детальний опис контролерів:

authController.js: Містить логіку, необхідну для реєстрації та входу користувачів, обробки відновлення паролів та підтвердження їх електронних адрес. Також включає методи для виходу з системи та перевірки поточного користувача на аутентифікацію та авторизацію.

cartController.js: Забезпечує функції для додавання товарів до корзини, їх видалення та оновлення кількості. Відповідає за виведення деталізованої інформації про товари в корзині та обробку самого процесу покупки.

errorController.js: Управляє помилками, які можуть виникнути під час виконання додатку. Включає методи для обробки виключень, логування помилок та відправлення відповіді з помилкою до клієнта.

handlerFactory.js: Фабрика, яка стандартизує створення базових CRUD операцій (створення, читання, оновлення, видалення), що використовується різними контролерами для уникнення дублювання коду.

productController.js: Управління всіма аспектами життєвого циклу продукту. Відповідає за отримання списку товарів, додавання нових товарів, оновлення інформації про існуючі товари та їх видалення.

transactionController.js: Контролер для обробки покупок і фінансових операцій. Він дозволяє здійснювати та відстежувати транзакції, включаючи повернення товарів.

userController.js: Займається обробкою даних користувачів. Включає функції для отримання профілю користувача, його оновлення, а також видалення користувачів з системи.

viewsController.js: Орієнтований на серверне рендеринг відображень. Використовує шаблонізатор для генерації HTML-сторінок на основі даних, отриманих з моделей або контролерів.

3 Моделі визначають структуру даних, їх взаємозв'язки та поведінку в контексті бази даних. Вони служать як шар абстракції для роботи з даними, дозволяючи легко маніпулювати записами без необхідності прямого звернення до бази. Ось детальний опис моделей:

productModel.js: Визначає схему для товарів, яка включає необхідні поля, такі як назва, ціна, опис, категорія, кількість на складі, виробник і зображення. Модель також може містити методи для валідації даних, передбачені правила для забезпечення унікальності та використання мідлварів для автоматизації процесів, таких як обновлення часу останнього редагування товару.

userModel.js: Визначає схему для користувачів, включаючи інформацію, необхідну для аутентифікації та управління профілем, наприклад, ім'я, електронна пошта, пароль, роль в системі та статус активності. Ця модель містить інформацію про вибрані товари, їх кількість, дату придбання і адресу доставки. Модель інкапсулює в собі логіку для хешування паролів перед зберіганням у базі даних, а також методи для перевірки введеного пароля при аутентифікації користувача. За допомогою кастомних методів модель може також генерувати токени для скидання паролів та їхнє підтвердження.

4 Роути визначають ендпоінти API, які використовуються для взаємодії з клієнтською стороною додатку, та визначають, які контролери мають відгукуватися на різні HTTP запити. Далі наведено опис кожного файлу маршрутів:

productRoutes.js: Містить маршрути, які керують запитами до продуктів. Це включає отримання списку всіх продуктів, отримання деталей про окремий продукт за його ID, створення нового продукту, оновлення існуючого продукту, та видалення продукту. Маршрути можуть використовувати параметри для фільтрації, сортування, обмеження кількості продуктів та пагінації.

transactionRoutes.js: Включає маршрути для обробки транзакцій, таких як оформлення замовлення та його оплати, перегляду історії транзакцій користувачем, та обробки повернень. Вони визначають логіку для підтвердження успішної оплати та зміни статусу замовлення.

userRoutes.js: Встановлює маршрути для управління даними користувачів. Основні операції включають реєстрацію нових користувачів, вхід в систему, оновлення профілю користувача, зміну паролю та видалення акаунта. Також можуть включати маршрути для забутого пароля та його відновлення.

viewRoutes.js: Задає маршрути для серверного рендерингу веб-сторінок. Це не API маршрути, а маршрути, які повертають відповіді у вигляді готових веб-сторінок. Вони використовують viewsController для надання відповідних шаблонів Pug з даними, які мають бути відображені.

5 Утиліти — це допоміжні інструменти, що використовуються для реалізації загальнопризначенових функцій, які можуть бути застосовані в різних частинах додатку, полегшуючи розробку та підтримку коду. Ось детальніше про кожен з утилітних скриптів:

apiFeatures.js: Цей утиліт розширює можливості API, надаючи додаткові функції для роботи з запитами. Він може включати такі можливості, як фільтрація результатів на основі параметрів запиту, сортування, пагінація та обмеження кількості результатів, які повертаються в одному запиті. Це дозволяє розробникам легко впроваджувати складні запити без необхідності писати багато додаткового коду.

appError.js: Цей скрипт визначає клас помилок, який можна використовувати для створення об'єктів помилок з налаштованими повідомленнями та статус-кодами HTTP. Використання цього класу дозволяє розробникам стандартизувати відповіді про помилки по всьому додатку та легше управляти помилками.

catchAsync.js: Утиліта для оптимізації роботи з асинхронним кодом. Функція catchAsync приймає асинхронну функцію та повертає нову функцію, яка автоматично передає будь-які винятки (errors) до наступного middleware з помилками в Express. Це відміняє необхідність використання блоків try/catch в кожному асинхронному контролері.

email.js: Утиліта для налаштування та відправлення електронних листів через додаток. Вона може включати інтеграцію з поштовими сервісами та функції для створення шаблонів листів, які відправляються користувачам — наприклад, підтвердження реєстрації, нагадування про забуті паролі, повідомлення про замовлення тощо.

3.2.2 База даних

У цьому проекті було вибрано розробку архітектури бази даних з використанням MongoDB, оскільки це нереляційна база даних, яка ідеально підходить для зберігання великої кількості документів з динамічними схемами.

MongoDB, будучи документоорієнтованою базою даних, дозволяє ефективно зберігати складні ієрархії та взаємозв'язки між даними, що є важливим для цієї схеми зберігання і обробки інформації про користувачів та продукти.

На рисунку 3.3 представлена структура бази даних у вигляді таблиць, хоча використовується документно-орієнтована база даних, в якій всі дані зберігаються у вигляді двійкових документів JSON (BSON). Ця діаграма надає візуальне уявлення про взаємозв'язки між основними компонентами бази даних.

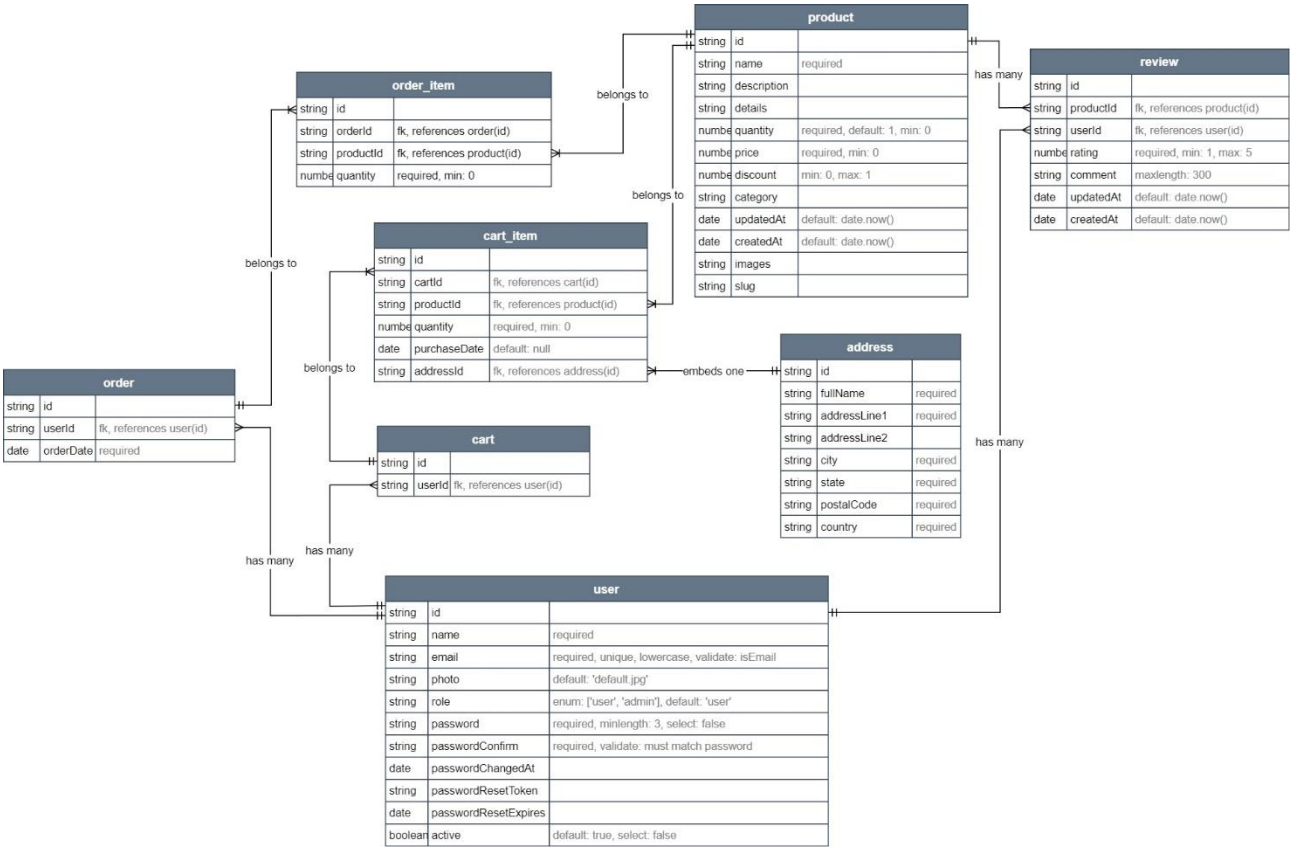

Рисунок 3.3 – Структура бази даних (Схема зв’язків документів)

У створеній базі даних присутні два основних типи документів: user та product. Документ user зберігає не лише особисті дані користувача, але й дані для автентифікації, а також включає поле кошику. Кошик реалізований як масив, що містить посилання на продукт і кількість цього продукту, що дозволяє ефективно керувати покупками користувача.

Документ product містить детальну інформацію про товар, включно з відгуками. Відгуки складаються з посилання на профіль користувача, що залишив відгук, рейтингу товару, дати публікації та тексту відгуку. Ця структура сприяє простоті отримання відгуків та оцінювання продуктів, що підвищує довіру та задоволення користувачів.

Схема бази даних включає механізми валідації паролів користувачів, що забезпечує безпеку даних. Крім того, передбачено автоматичне видалення відгуків при видаленні облікового запису користувача та автоматичне оновлення і видалення кількості продуктів у кошику при зміні наявності або кількості товарів. Ці механізми підвищують ефективність управління даними і гарантують актуальність інформації для користувачів.

3.2.3 Безпека web-додатку

У рамках розробки вебдодатку основним завданням було створення стійкої та безпечної платформи, яка б забезпечувала надійний захист користувачів та їхніх даних. Реалізація цього завдання вимагала впровадження сучасних технологій та методологій, спрямованих на ефективний захист інформації та запобігання загрозам безпеці.

Основа системи аутентифікації — це використання JSON Web Tokens (JWT), яке дозволяє встановлювати сесії користувачів без необхідності зберігання сесійних ідентифікаторів на сервері. Такий підхід знижує ризик атак типу session hijacking. Токени зберігаються в HTTP-only cookies, що робить їх недоступними для скриптів на стороні клієнта та захищає від XSS атак.

Для захисту паролів користувачів застосовується технологія bcrypt.js. Ця технологія передбачає складне шифрування паролів перед їх збереженням у базі даних, що включає додавання випадково генерованої послідовності символів до пароля. Це значно ускладнює можливість розшифрування пароля, навіть якщо зловмисники отримають доступ до бази даних.

Щодо захисту від ін’єкцій, вебдодаток включає застосування бібліотеки mongoSanitize, яка видаляє потенційно небезпечні символи з запитів до бази

даних, тим самим запобігаючи можливість введення шкідливих команд. Додатковий рівень захисту проти SQL-ін’єкцій забезпечується завдяки використанню параметризованих запитів.

З метою протидії атакам типу cross-site scripting (XSS), вебдодаток використовує фільтрацію вводу користувачів і екранування спеціальних символів перед їх відображенням на сторінках за допомогою бібліотеки xss-clean. Це забезпечує автоматичне фільтрування потенційно небезпечного контенту.

Обмеження швидкості запитів реалізоване через використання express-rate-limit, що дозволяє встановлювати обмеження на кількість запитів, які може виконати одна IP-адреса за визначений період часу, що сприяє запобіганню атакам типу DDoS. Ці заходи є частиною комплексної стратегії забезпечення безпеки, яка розроблена для захисту даних та забезпечення безпеки користувачів на веб-платформі.

3.3 Використання програмного продукту

3.3.1 Загальний огляд клієнтської частини додатку

Сторінки "Головна" (рис. 3.4), "Блог" (рис. 3.5), "Про нас" (рис. 3.6) та "Контакти" (рис. 3.7) мають базовий функціонал і доступні для всіх відвідувачів вебсайту. Їхня основна мета — надання інформації. Сторінка перегляду товарів відкрита для всіх користувачів, однак доступ до окремих функцій є обмеженим. Сторінка "Кошик" доступна лише для авторизованих користувачів, а сторінка "Адміністрування" — виключно для користувачів з адміністративними привілеями.

Всі сторінки вебсайту розроблені в єдиному адаптивному дизайні, що забезпечує візуальну гармонію та зручність використання з будь-якого

пристрою. Кожна з них містить навігаційну панель, блок підписки на новини та футер, сприяючи збереженню візуальної консистенції вебсайту.

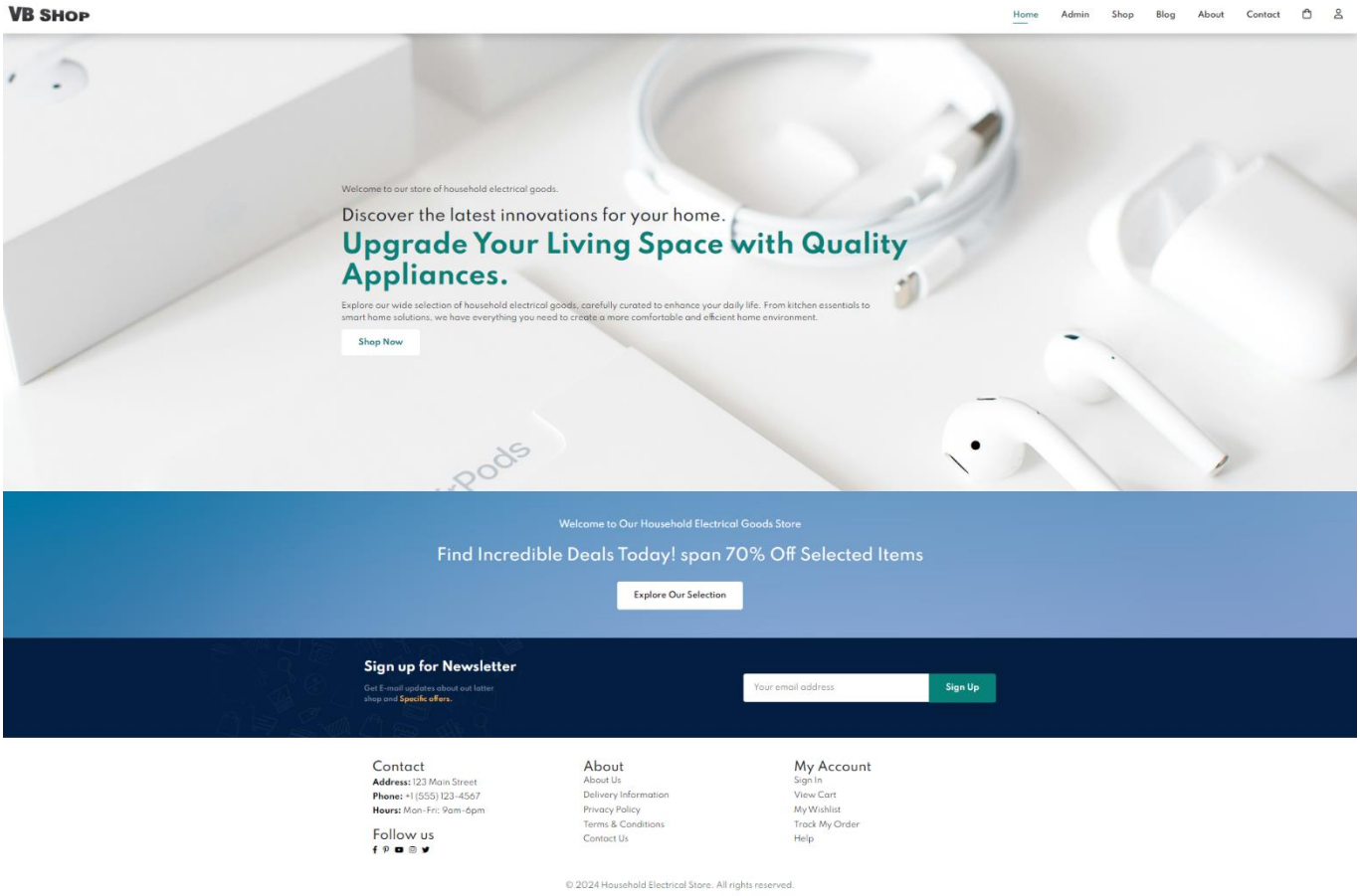

Рисунок 3.4 – Головна сторінка

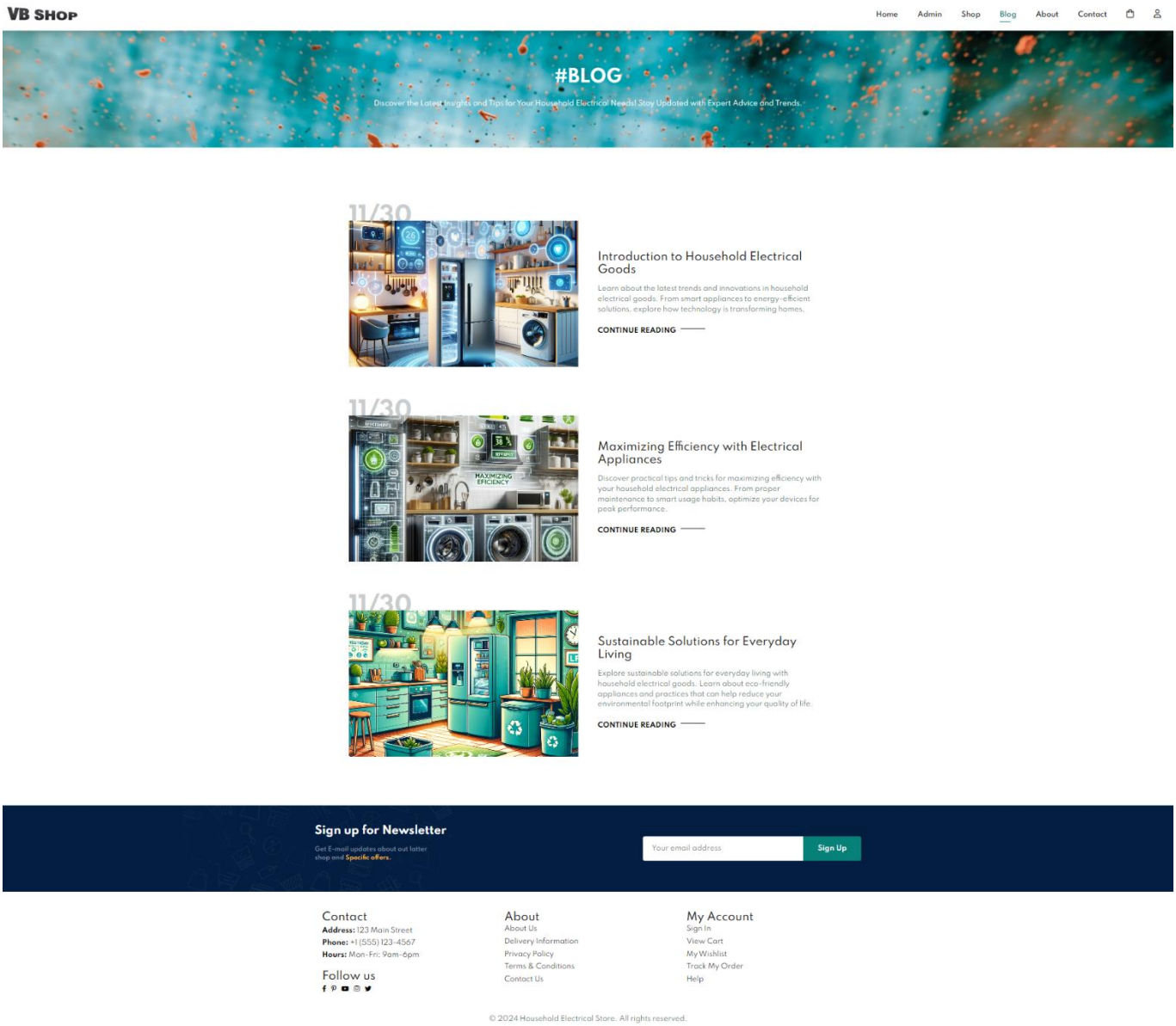

Рисунок 3.5 – Сторінка блогу

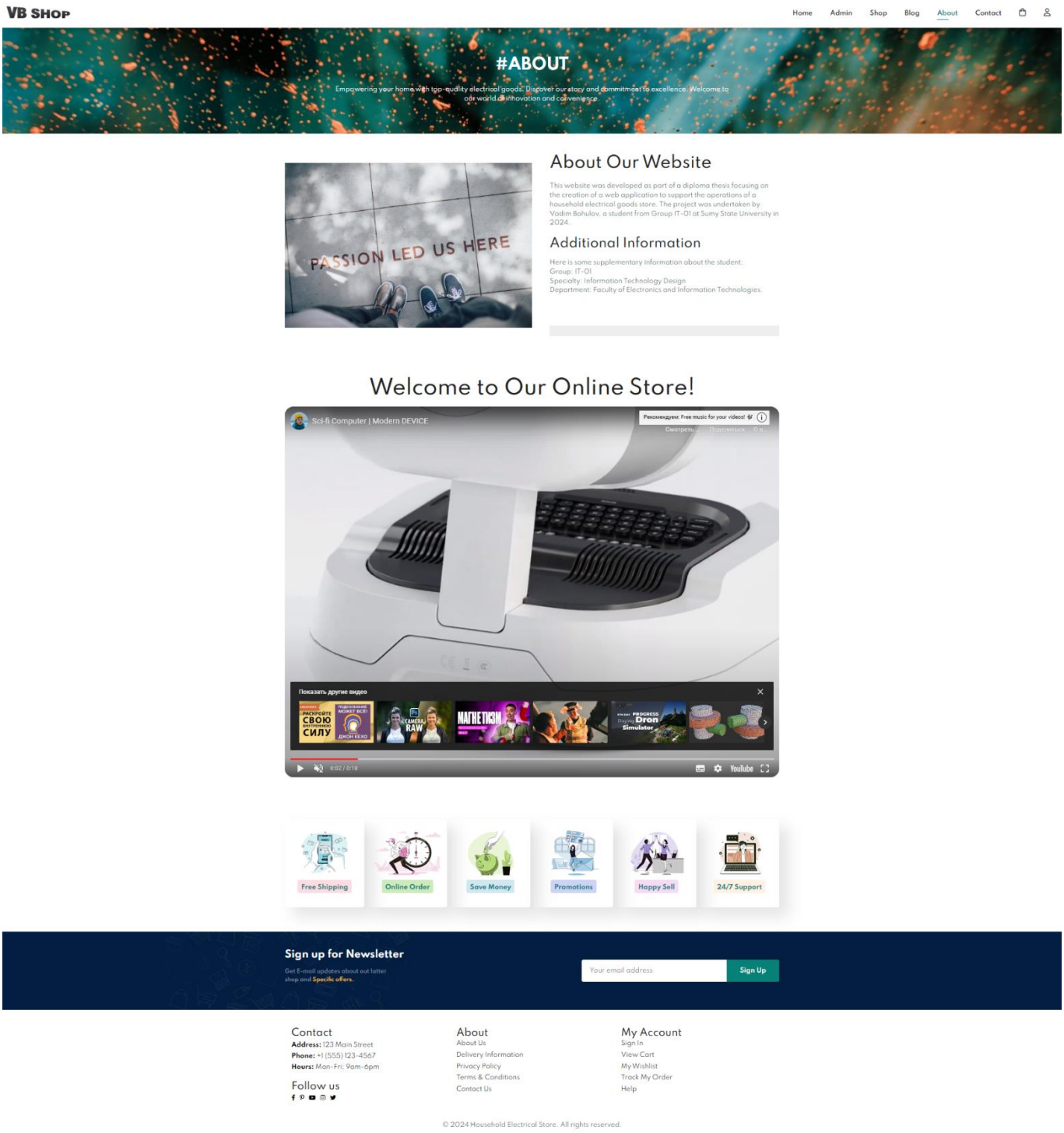

Рисунок 3.6 – Сторінка «Про нас»

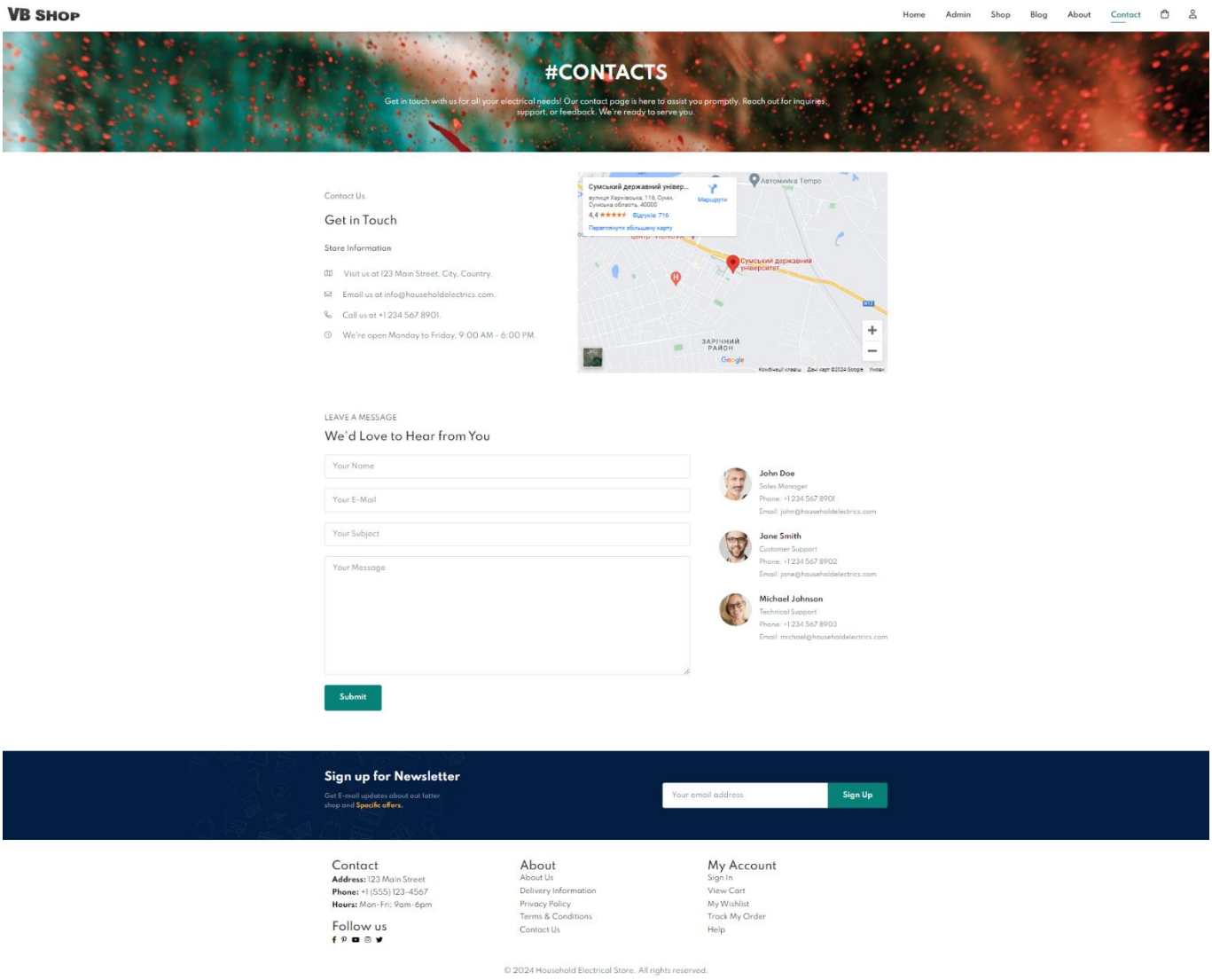

Рисунок 3.7 – Сторінка «Контакти»

Додатково, на вебсайті присутня спеціальна сторінка для відображення помилок, яка інформує користувачів про технічні проблеми, необхідність авторизації або відсутність прав доступу (рис. 3.8).

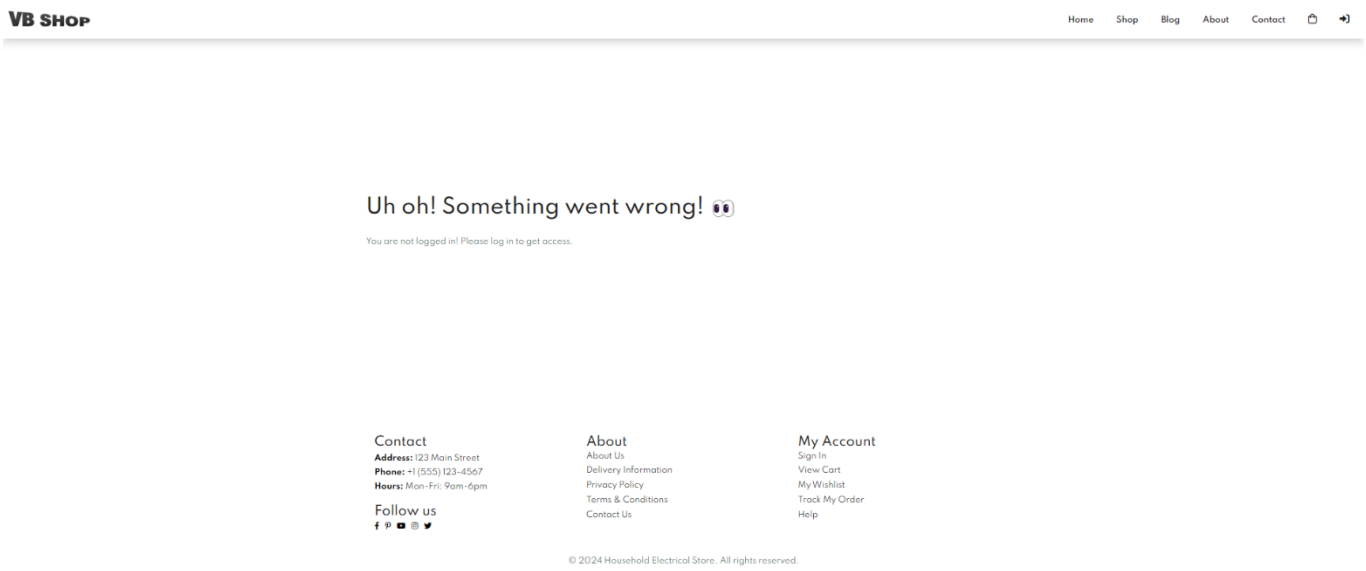

Рисунок 3.8 – Сторінка з повідопмленням про помилку

У навігаційній панелі, відповідно до статусу користувача, відображається іконка, що спрямовує до модального вікна автентифікації або профілю користувача. Для авторизованих користувачів з адміністративними правами, в панелі також доступне посилання на сторінку "Адміністрування" (рис. 3.9).

Рисунок 3.9 – Навігаційна панель

Модальне вікно входу (рис. 3.10) або реєстрації (рис. 3.11) відображається, якщо користувач не авторизований на платформі. Тут користувач може ввійти до свого облікового запису, використовуючи існуючі дані, або ж зареєструватися, якщо він новий на сайті.

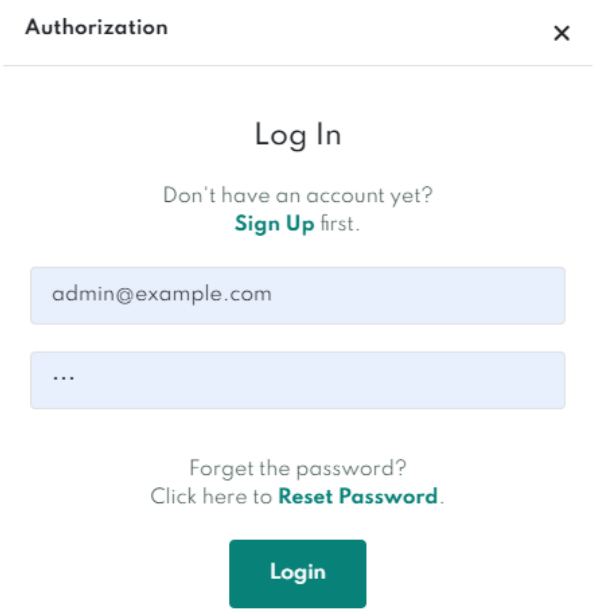

Рисунок 3.10 – Модальне вікно входу

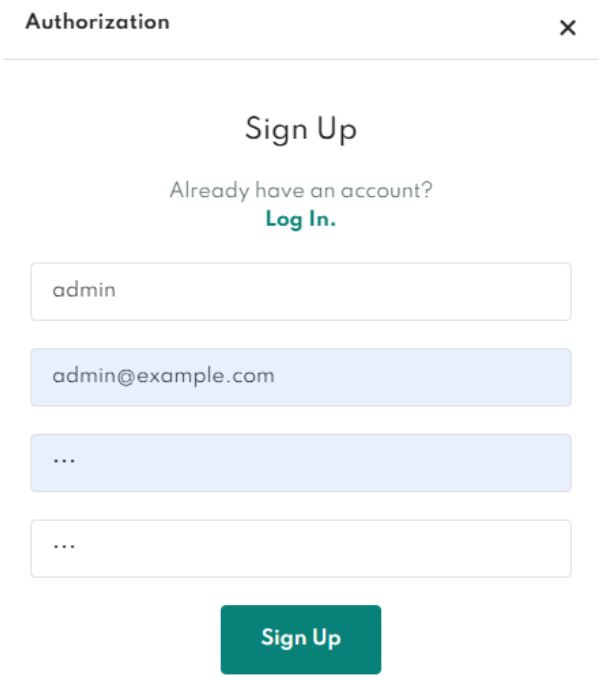

Рисунок 3.11 – Модальне вікно реєстрації

Інше модальне вікно - це профіль користувача (рис. 3.12). Після входу або реєстрації, користувач може переглядати свій профіль, де має можливість оновлювати дані, зокрема фотографію, ім'я або електронну пошту.

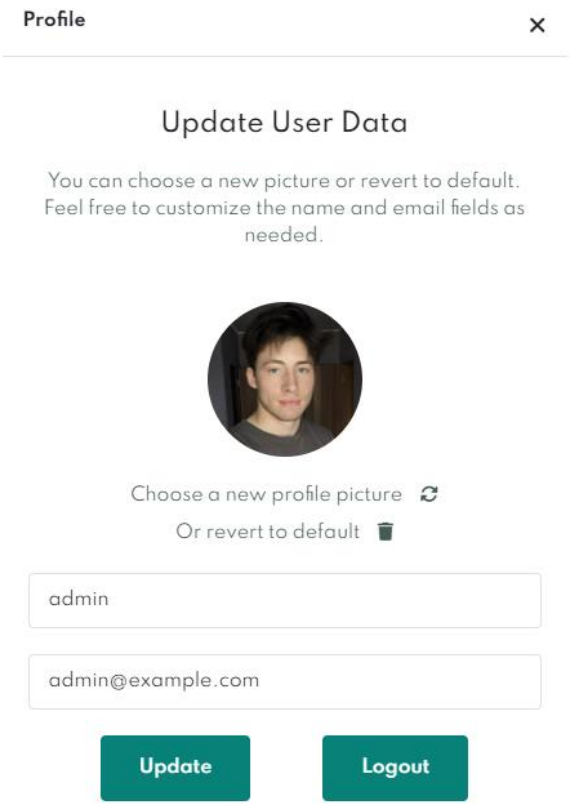

Рисунок 3.12 – Модальне вікно профілю

3.3.2 Сторінка каталогу товарів

Сторінка каталогу товарів (рис 3.13) розроблена для доступу всіма відвідувачами сайту. Вона обладнана функціями фільтрації за категоріями, пошуком за назвою та сортуванням за параметрами, такими як назва, ціна та дата додавання. Для підвищення зручності користувачів і оптимізації процесу перегляду результатів, впроваджено систему пагінації. Результати пошуку оновлюються в реальному часі без потреби перезавантажувати сторінку. Налаштування пошуку зберігаються у localstorage, що дозволяє відвідувачам повертатися до своїх попередніх запитів без втрати даних. Кожна картка товару

у списку включає зображення, назву, короткий опис, ціну, середню оцінку з кількістю відгуків та кнопку для навігації до детальної сторінки товару.

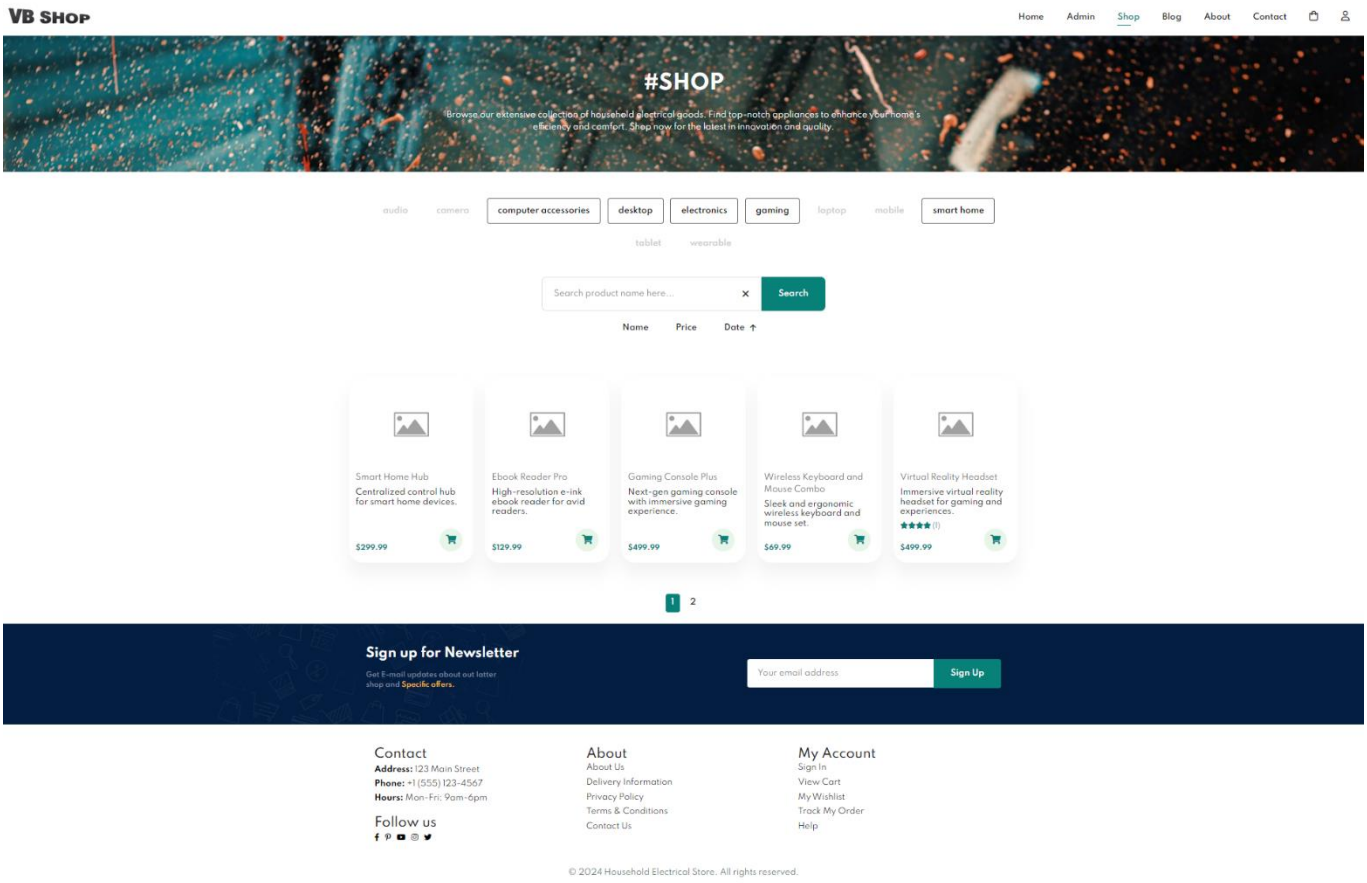

Рисунок 3.13 – Сторінка каталогу товарів

3.3.3 Сторінка товару

Сторінка товару (рис 3.14) доступна всім відвідувачам, але функції, такі як додавання товару до кошика та створення відгуку, доступні лише для авторизованих користувачів. На сторінці реалізований зручний перегляд зображень та детально відображається інформація про товар, включаючи назву, ціну, короткий та детальний опис, рейтинг та відгуки інших користувачів. Відгуки зібрані в нижній частині сторінки, де відображаються імена користувачів, їхні фотографії профілів, оцінка товару від 1 до 5 зірок та, за наявності, текст відгуку.

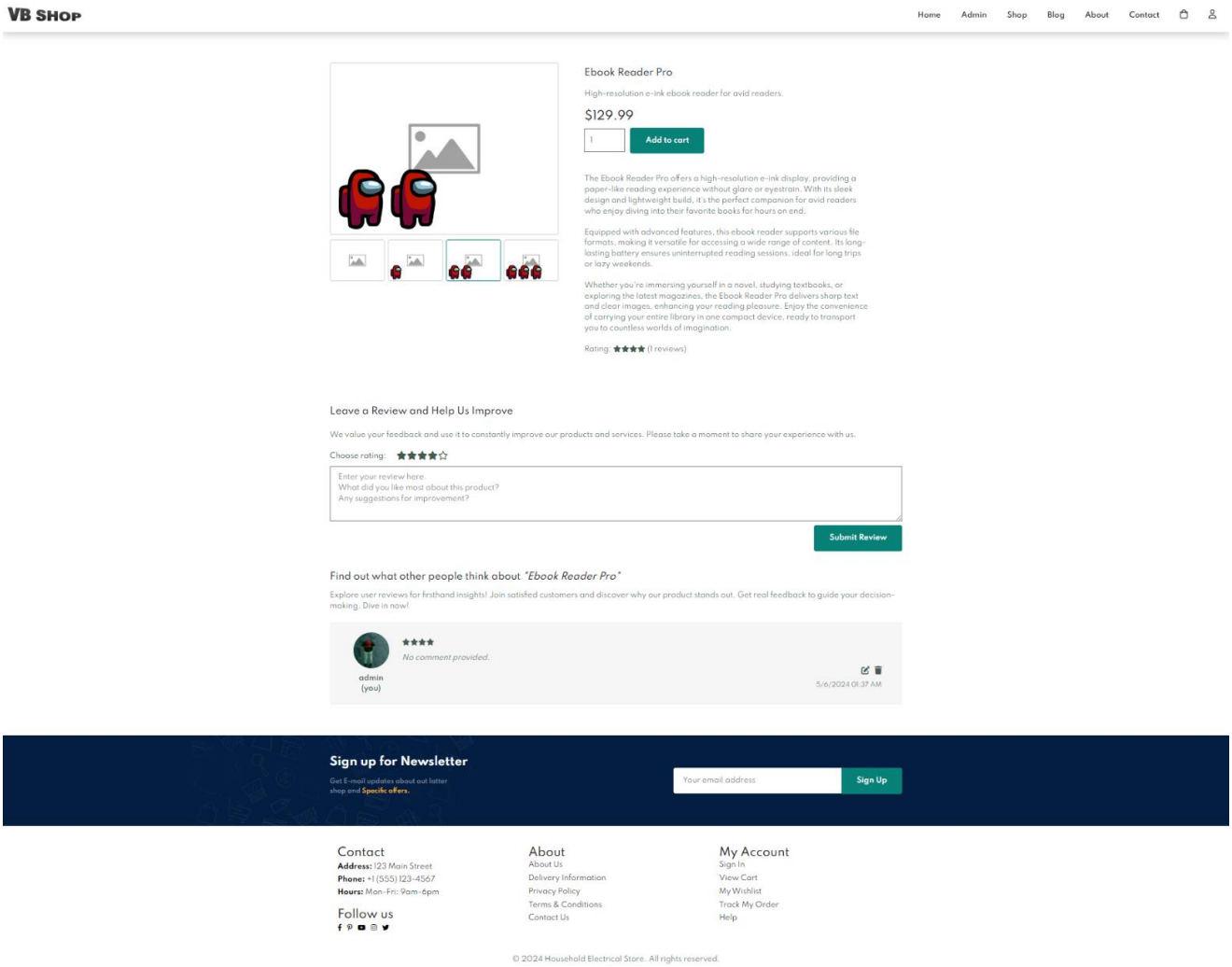

Рисунок 3.14 – Сторінка товару

Авторизовані користувачі мають можливість обирати кількість товару. Поле вводу проходить валідацію, яка включає перевірку на доступну кількість товарів в базі даних і в разі помилки відображає відповідне повідомлення. Якщо товар уже присутній у кошику користувача, система лише оновлює його кількість, що запобігає дублюванню і сприяє покращенню користувацького досвіду.

Кожен авторизований користувач має право створити один відгук на товар, де може вказати оцінку та за бажанням додати текст відгуку. Відгуки містять позначку, що ідентифікує автора, а також надається можливість видалення власного відгуку.

3.3.4 Сторінка кошику

Доступ до сторінки кошику (рис 3.15) обмежений для неавторизованих користувачів. У випадку відсутності авторизації, користувачу відображається сторінка з помилкою, стилізованою та інформативною, яка повідомляє про необхідність входу в систему для доступу до кошику. Сторінка кошику включає таблицю з обраними товарами, де представлено кнопку для видалення товару, зображення, назву, ціну, а також поле для коригування кількості з валідацією за максимально доступною кількістю товару з бази даних. В стовпці таблиці вказується загальна вартість обраних товарів у кошику в залежності від їх кількості. Під таблицею розміщена ще одна, яка показує загальну вартість усіх товарів у кошику, а також кнопку, що веде до сторінки оплати.

У нижній частині сторінки кошику розташована історія замовлень, де відображені придбані товари, груповані та відсортовані за датами. Для кожного дня, коли були зроблені покупки, створено окрему таблицю з переліком замовлених товарів. Кожен елемент таблиці містить зображення товару, назву, короткий опис, точний час замовлення та загальну вартість; якщо кількість товару в замовленні перевищує одиницю, відображається кількість і ціна за одиницю товару.

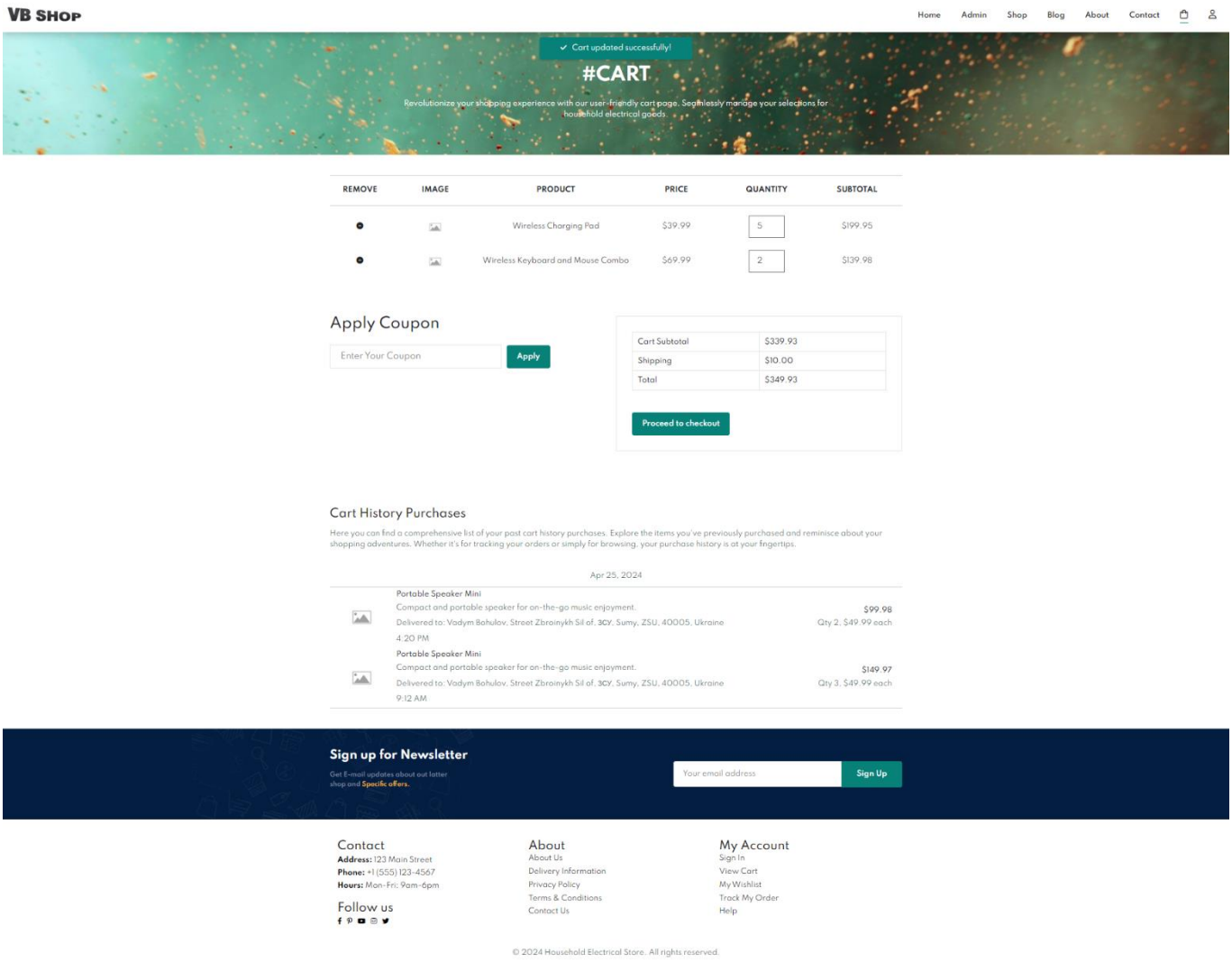

Рисунок 3.15 – Сторінка кошику

3.3.5 Сторінка оплати

Перехід на сторінку оплати можливий лише у випадку, коли в кошику користувача присутній щонайменше один товар у кількості більше нуля та з загальною вартістю понад 50 центів. Після перевірки цих умов відкривається модальне вікно для вказування адреси доставки (рис. 3.16), яке включає поля які є необхідними для обробки доставки. У цьому вікні вимагається вказати повне ім'я одержувача, адресу доставки, місто, штат, поштовий код та країну для всіх товарів у кошику, які підлягають придбанню.

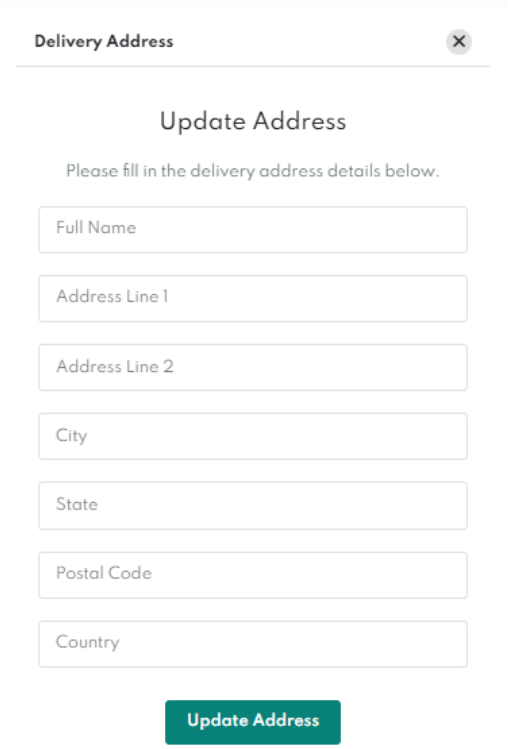

Рисунок 3.16 – Модальне вікно вказання адреси доставки

Оплата здійснюється через сервіс Stripe (рис. 3.17). Після успішної транзакції до кожного товару у кошику додається мітка purchasedDate, що позначає товар як придбаний. Придбані товари відображаються на на сторінці кошику в історії замовлень і не підлягають видаленню чи редагуванню.

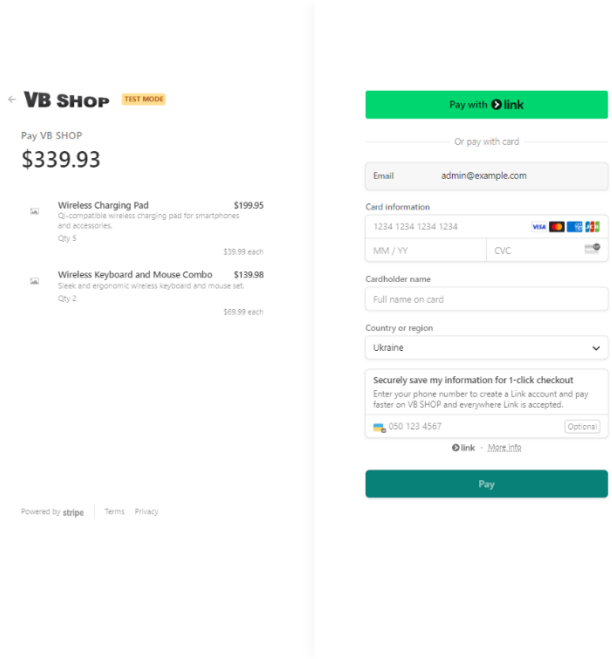

Рисунок 3.17 – Сторінка оплати

3.3.6 Сторінка адміністратора

Сторінка адміністратора (рис. 3.18) є інтерфейсом для управління базою товарів. Вона обладнана таблицею, яка дозволяє вносити зміни до товарів, включаючи назву, ціну, кількість, короткий опис, детальний опис, до чотирьох зображень та категорію. Таблиця також показує сумарну вартість кожного товару, а також дати його створення та останнього оновлення.

Особливості адміністративної панелі включають динамічне оновлення контенту без потреби перезавантаження сторінки, що сприяє оптимізації взаємодії. Додатково, є функціонал для завантаження та перегляду зображень товарів за допомогою зручного слайдера.

Категорії можна швидко додавати, вводячи нову назву прямо на сторінці, або обирати з наявних у випадаючому списку. При оновленні інформації про товар автоматично оновлюється і дата останнього оновлення. Товари відсортовані за часом останнього оновлення. На сторінці присутні кнопки для створення нового товару, які створюють запис про пустий товар за замовчуванням, а також кнопка для видалення товару. Кожне поле включає систему валідації, що гарантує коректність даних.

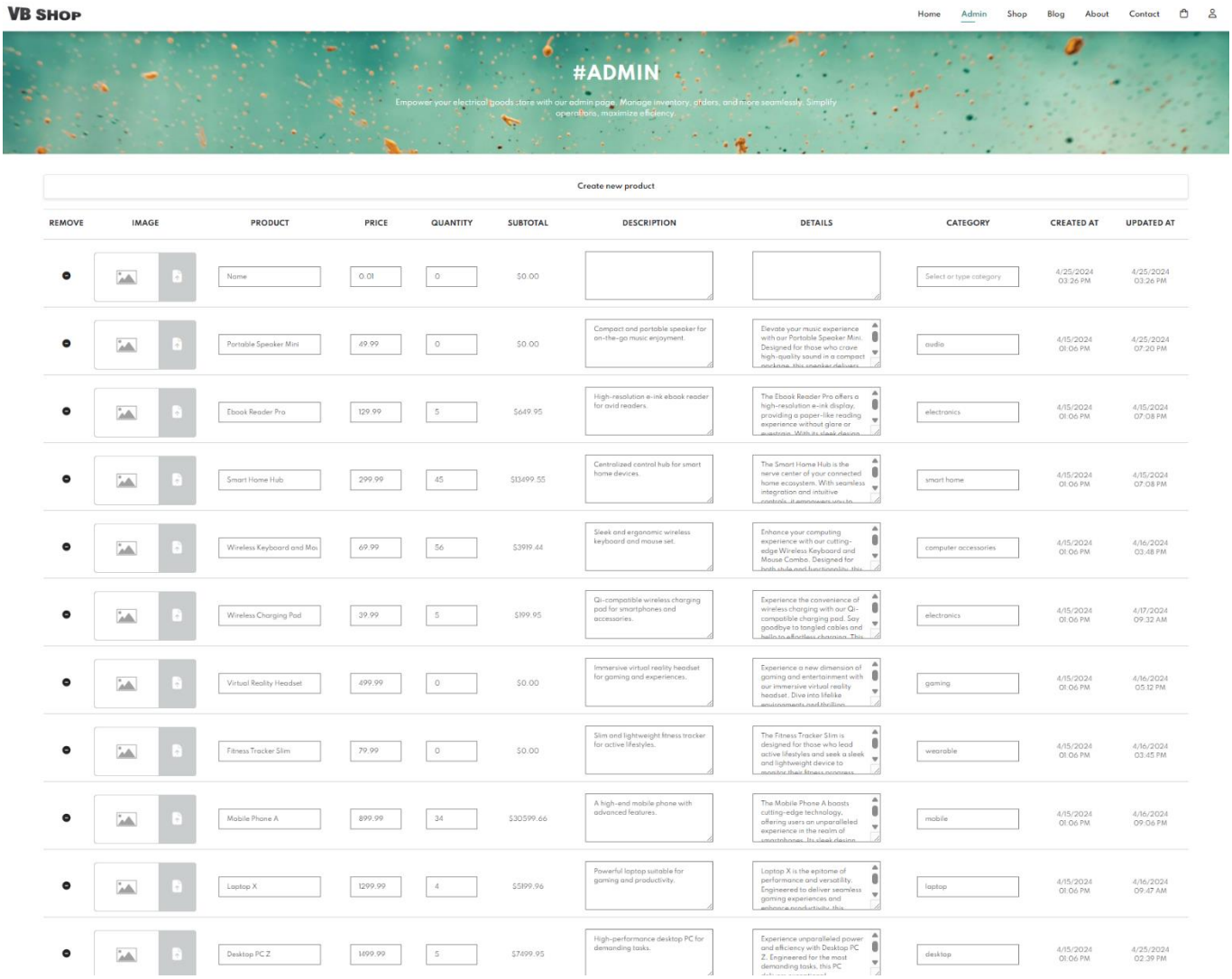

Рисунок 3.18 – Сторінка адміністратора

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Козловський В. Л. Удосконалення системи фінансового контролінгу на підприємстві : master's thesis. 2021. URL: https://essuir.sumdu.edu.ua/handle/123456789/82221 (дата звернення: 21.04.2024).

2. Богулов В. Р. Web-додаток підтримки діяльності магазину електротехнічних товарів побутового використання // ІНФОРМАТИКА, МАТЕМАТИКА, АВТОМАТИКА: матеріали та програма. Суми: Сумський державний університет, 2024. С. 242. URL: https://drive.google.com/file/d/11_r9kqQuZxVKSm8HABWWNUMEJN2T4urZ/view (date of access: 25.05.2024).

3. Нів'євський О. Бізнесу потрібна нова індустріальна політика. Чому владі не варто робити ставки на окремі ринки, ланцюги доданої вартості чи підприємства. Розповідає агроексперт Олег Нів'євський – Forbes.ua. Forbes.ua | Бізнес, мільярдери, новини, фінанси, інвестиції, компанії. URL: https://forbes.ua/money/forbes-blog-chinovniki-ne-providtsi-ta-mantri-pro-stvorennya-dodanoi-vartosti-14022024-19206 (дата звернення: 21.04.2024).

4. НЛП методи впливу на людину. Інформаційний блог за інтересами для красивих чоловіків і жінок. URL: https://velur.kiev.ua/nlp-metodi-vplivu-na-lyudinu/ (дата звернення: 21.04.2024).

5. Новини компаній. Чому покупці надають перевагу українським інтернет-магазинам: Emporium з'ясував причини. ZAXID.NET. URL: https://zaxid.net/chomu_pokuptsi_nadayut_perevagu_ukrayinskim_internet_magazinam_emporium_zyasuvav_prichini_n1580829 (дата звернення: 21.04.2024).

6. Хмілевська В. Обсяги продажів техніки й електроніки в Україні зросли на 17% в порівнянні з 2022 роком. Про причини зростання розповіли експерти Promocodius. Зеркало недели | Дзеркало тижня | Mirror Weekly. URL: https://zn.ua/ukr/novosti-kompaniy/obsjahi-prodazhiv-tekhniki-j-elektroniki-v-

ukrajini-zrosli-na-17-v-porivnjanni-z-2022-rokom-pro-prichini-zrostannja-rozpovili-eksperti-promocodius.html (дата звернення: 21.04.2024).

7. Batkhin Y. Leading the Digital Payments Evolution in Ukraine. Mastercard - Ein globales Technologieunternehmen in der Zahlungsverkehrsbranche. URL: https://www.mastercard.com/news/eemea/en/perspectives/en/2023/leading-the-digital-payments-evolution-in-ukraine/ (date of access: 21.04.2024).

8. eBay Features, Drawbacks, and All You Need to Know - Blog Non-VoIP. Blog Non-VoIP. URL: https://blog.non-voip.com/ebay-features-drawbacks-and-all-you-need-to-know/ (date of access: 21.04.2024).

9. Express - Node.js web application framework. Express - Node.js web application framework. URL: https://expressjs.com/ (date of access: 21.04.2024).

10. Garrofé A., Picca M., Kaplan A. https://actaodontologicalat.com/wp-content/uploads/2022/05/aol_2022_35-1-03.pdf. Acta Odontológica Latinoamericana. 2022. Vol. 35, no. 1. P. 10–15. URL: https://doi.org/10.54589/aol.35/1/10 (date of access: 21.04.2024).

11. How Does the Amazon Recommendation System Work? | Baeldung on Computer Science. Baeldung on Computer Science. URL: https://www.baeldung.com/cs/amazon-recommendation-system (date of access: 21.04.2024).

12. Index | Node.js v21.7.3 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/docs/latest/api/ (date of access: 21.04.2024).

13. Relational Vs. Non-Relational Databases | MongoDB. MongoDB. URL: https://www.mongodb.com/compare/relational-vs-non-relational-databases (date of access: 21.04.2024).

14. Snevy1. 10 Best Web Development Stacks. DEV Community. URL: https://dev.to/snevy1/10-best-web-development-stacks-409i (date of access: 21.04.2024).

15. Перевозчикова Ю. Оптимізація бізнес-процесів як фактор реалізації потенціалу.
URL: https://view.officeapps.live.com/op/view.aspx?src=https://kneu.edu.ua/userfiles

/conf_sep_14/sek3/Perevozchukova/tezu.docx&wdOrigin= (дата звернення: 25.05.2024).

16. Snevy1. 10 Best Web Development Stacks. *DEV Community*. URL: https://dev.to/snevy1/10-best-web-development-stacks-409i (date of access: 21.04.2024).

17. Express - Node.js web application framework. Express - Node.js web application framework. URL: https://expressjs.com/ (date of access: 25.05.2024).

18. Index | Node.js v22.2.0 Documentation. Node.js – Run JavaScript Everywhere. URL: https://nodejs.org/docs/latest/api/ (date of access: 25.05.2024).

19. Relational Vs. Non-Relational Databases | MongoDB. MongoDB. URL: https://www.mongodb.com/resources/compare/relational-vs-non-relational-databases (date of access: 25.05.2024).

ДОДАТОК А

ТЕХНІЧНЕ ЗАВДАННЯ

на розробку інформаційної системи
«Вебдодаток підтримки діяльності магазину
електротехнічних товарів побутового використання»

ПОГОДЖЕНО:

Доцент кафедри інформаційних технологій

_____ Антипенко В.П.

Студент групи ІТ-01

Богулов В. Р.

Суми 2024

1 ПРИЗНАЧЕННЯ Й МЕТА СТВОРЕННЯ WEB-ДОДАТКУ

1.1 Призначення web-додатку

Призначенням вебдодатку є надання клієнтам повної інформації про асортимент товарів та послуг, які пропонує магазин електротехнічних товарів побутового використання. Крім того, вебдодаток також забезпечує зручне адміністрування магазину для його власників.

1.2 Мета створення web-додатку

Збільшення обсягів клієнтів та покращення обслуговування за допомогою онлайн-платформи сприятиме підвищенню доступності продуктів для покупців, забезпечуючи їм зручний спосіб вибору та замовлення товарів.

1.3 Цільова аудиторія

Цільова аудиторія вебдодатку підтримки діяльності магазину електротехнічних товарів побутового використання охоплює широке коло споживачів, що цікавляться покупкою, обслуговуванням або обміном побутової електротехніки.

2 ВИМОГИ ДО WEB-ДОДАТКУ

2.1 Вимоги до web-додатку в цілому

2.1.1 Вимоги до структури й функціонування web-додатку

Вебдодаток має бути доступним в мережі Інтернет на платформі хостингу Heroku. Структура вебдодатку повинна складатися з різних секцій, які взаємодіють між собою та мають чітко визначені функції.

2.1.2 Вимоги до персоналу

Вимоги до персоналу полягають у тому, щоб вони мали загальні навички роботи з персональним комп'ютером і стандартним веб-браузером. Також важливо, щоб вони були ознайомлені з адміністративною панеллю для керування товарами на веб-сайті магазину електротехнічних товарів.

Наявність навичок роботи з API для взаємодії з системою магазину, якщо використання адмін-панелі не є оптимальним способом керування товарами.

2.1.3 Вимоги до збереження інформації

Всі дані, надані у вебдодатку, будуть зберігатися у базі даних, що буде реалізована засобами системи управління базами даних MongoDB. Зокрема, файли зображень, такі як фотографії продуктів і профілі користувачів, будуть зберігатися на сервері.

2.1.4 Вимоги до розмежування доступу

Вебдодаток має бути загальнодоступним для всіх користувачів.

Згідно з правами доступу до інформації у вебдодатку, користувачі можуть бути поділені на три категорії: неавторизовані відвідувачі, авторизовані користувачі і адміністратори.

Неавторизовані відвідувачі можуть переглядати більшість сторінок вебдодатку та ознайомлюватись з асортиментом товарів. Однак вони не мають доступу до сторінки кошика товарів та адміністративної панелі.

Авторизовані користувачі, після входу в систему, можуть скористатися додатковими функціями, такими як додавання товарів до кошика обраного, редагування списку обраних товарів у кошику та залишення відгуків на товари. Однак вони також не мають доступу до адміністративної панелі.

Лише користувачі з роллю адміністратора мають можливість переглядати і користуватися адміністративною панеллю і мати доступ до усіх API функцій.

2.2 Структура web-додатку

2.2.1 Загальна інформація про структуру web-додатку

Структура веб-сайту складається з набору сторінок, які зручно організовані для користувача та доступні через головне меню.

Ось список основних сторінок і модальних вікон:

Головна – основна сторінка.

Адмін – панель управління товарами.

Каталог товарів – список товарів з можливістю фільтрації по категоріям і сортуванням.

Один товар – інформація про конкретний товар та відгуками на нього.

Блог – статті та новини.

Про нас – інформація про вебдодаток.

Контакти – дані для зв'язку.

Кошик – перегляд і редагування обраних товарів та історії замовлень.

Модальне вікно – модальне вікно для авторизації та оновлення даних користувача.

2.2.2 Навігація

Для навігації, у шапці сторінки створена система контент меню. Меню необхідне для швидкого переміщення користувача по усім доступним сторінкам. Меню буде відображатися на всіх сторінках, щоб відвідувач міг в будь-який момент часу перейти на будь-яку сторінку web-додатку.

2.2.3 Наповнення web-додатку (контент)

Для управління контентом вебдодатку використовується адміністративна панель, з якої можна зручно редагувати і додавати продукти. Вся інформація про товари та їх характеристики зберігається у базі даних MongoDB, за винятком зображень, які зберігаються на сервері.

Заповнення та редагування контенту відбувається через адміністративний інтерфейс або беспосередньо через API додатку, дозволяючи оперативно оновлювати інформацію про наявні товари та їх характеристики.

2.2.4 Дизайн та структура додатку

Вебдодаток для підтримки діяльності магазину електротехнічних товарів побутового використання втілює концепцію мінімалістичного дизайну зі світлою темою, що робить його зручним і приємним у використанні. Кожна сторінка додатку виконана в єдиному стилі, забезпечуючи єдність візуального досвіду для користувачів.

Розміщення елементів на сторінках ретельно продумане, залишаючи достатньо вільного місця для зручного сприйняття інформації. На кожній сторінці присутній достатній обсяг контенту, який відповідає потребам користувачів, не перевантажуючи їх зайвою інформацією.

Використання якісних зображень з Unsplash - сервісу з безкоштовними фотографіями високої якості та відкритою ліцензією для комерційних проектів - додає професійності та привабливості додатку, створюючи позитивне враження у користувачів.

На рисунку А.1. зображена загальна схема сторінки для кожного сайту, де розташування елементів показано схематично. Кожен сайт має власний контент, що розширює макет загальної сторінки.

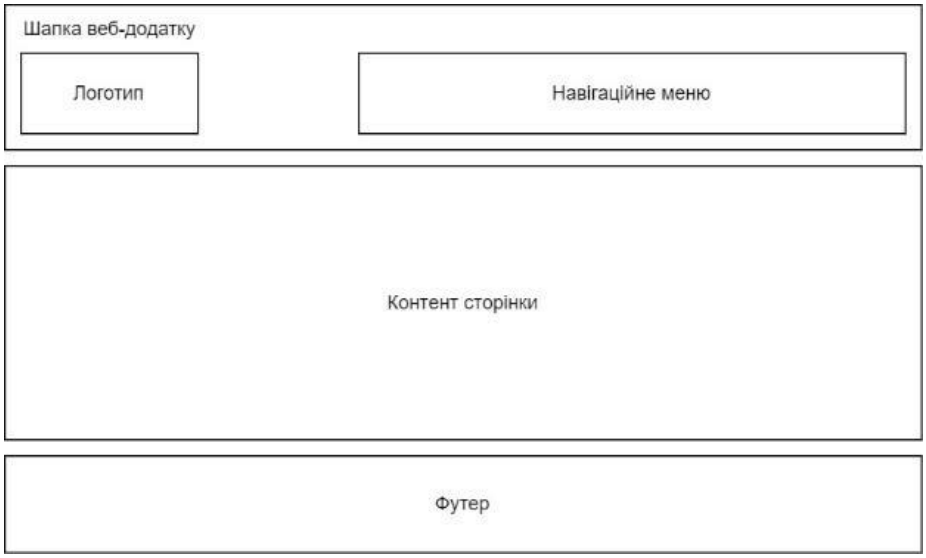

Рисунок А.1 – Загальна схема сторінки

Додаток включає наступні сторінки (див. рис. А.2–А.6): Головна, Адмін панель, Блог, Каталог товарів, Про нас та Контакти.

Рисунок А.2. – Макет сторінки "Головна"

Рисунок А.3. – Макет сторінки "Адмін панель"

Рисунок А.4. – Макет сторінки "Блог"

Рисунок А.5. – Макет сторінки "Каталог товарів"

Рисунок А.6. – Макет сторінки "Про нас"

Рисунок А.7. – Макет сторінки "Контакти"

Рисунок А.8. – Макет сторінки "Кошик"

Натискання на відповідний пункт меню в навігаційній панелі відкриє модальне вікно. Залежно від статусу користувача, це вікно може містити форму авторизації (логін або реєстрація) або профіль, який дозволяє користувачу оновлювати свої дані та зображення профілю (див. рис. А.8 та А.9).

Рисунок А.8. – Макет модального вікна авторизації

Рисунок А.9. – Макет модального вікна профілю

Якщо в додатку виникає неоперабельна помилка або посилання вказує на неіснуючу сторінку, або доступ до контенту потребує авторизації чи спеціальних прав доступу, то на сайті буде відображений текст, що повідомляє про помилку. Такий підхід забезпечить користувачеві зручність і розуміння ситуації, а також позитивно вплине на загальне враження від взаємодії з веб-ресурсом.

2.2.5 Система навігації (карта web-додатку)

Карта web-додатку зображена на рисунку А.10.

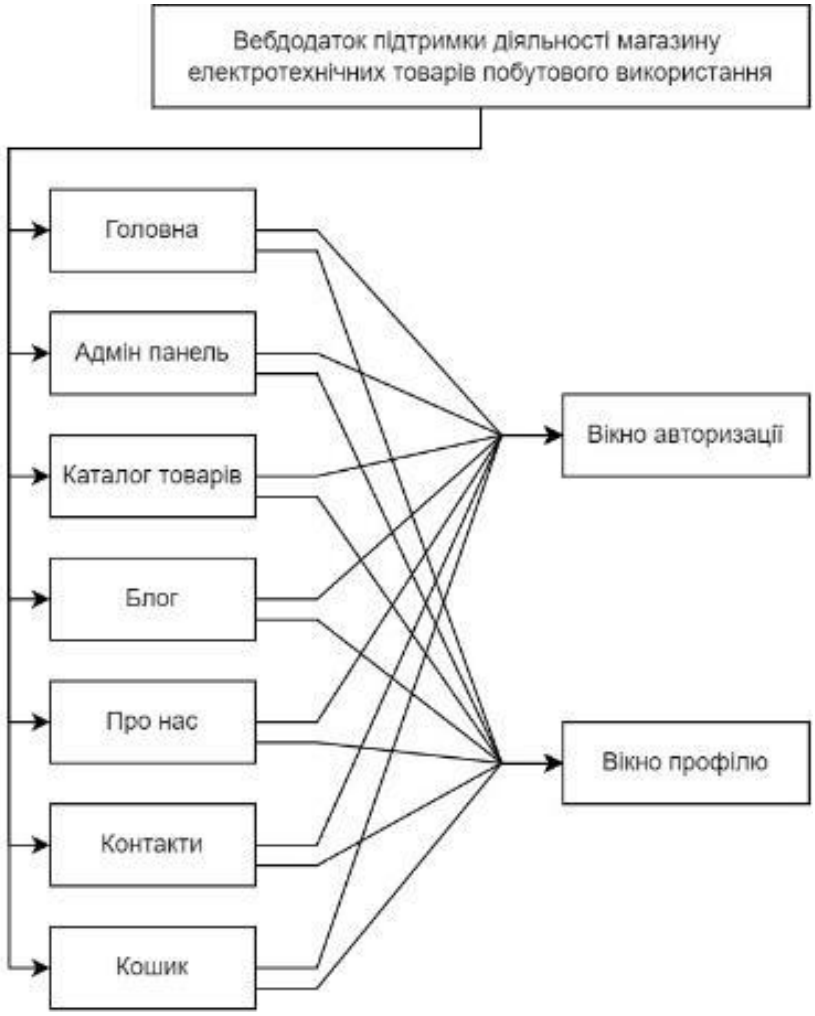

Рисунок А.2 – Карта web-додатку

2.3 Вимоги до функціонування системи

2.3.1 Потреби користувача

Нижче наведені потреби користувача, що були визначені для врахування у системі, і представлені у таблиці А.1.

Таблиця А.1 — Потреби користувача

ID	Потреби користувача	Джерело

UN-01	Перегляд сторінок: головна, блог, про додаток, контакти	Відвідувач
UN-02	Пошук товарів по категоріям і сортуванням	Відвідувач
UN-03	Перегляд детальної інформації про товар з відгуками	Відвідувач
UN-04	Логін та реєстрація користувача	Відвідувач
UN-05	Оновлення даних користувача, включаючи зображення профіля	Авторизований користувач
UN-06	Створення та видалення відгуків на товари	Авторизований користувач
UN-07	Додавання товарів в особистий кошик	Авторизований користувач
UN-08	Редагування списку товарів в кошику	Авторизований користувач
UN-09	Перегляд вартості замовлення	Авторизований користувач
UN-10	Перегляд адміністративної панелі та управління товарами в адмін панелі	Адміністратор
UN-11	Оплата обраних товарів	Авторизований користувач
UN-12	Перегляд історії замовлень	Авторизований користувач

2.3.2 Системні вимоги

Даний розділ визначає, розподіляє та вказує на системні вимоги, визначені розробником. Їх перелік наведений в таблиці А.2.

Таблиця А.2 — Системні вимоги

ID	Системні вимоги	Пріоритет	Опис
SR-01	Наявність модуля пошуку товарів	M	Забезпечує можливість виконання пошуку товарів за категоріями та сортуванням

Продовження таблиці А. 2

ID	Системні вимоги	Пріоритет	Опис
SR-02	База даних товарів	M	Надає можливість відображення детальної інформації про кожен товар
SR-03	Можливість реєстрації та авторизації користувачів	M	Забезпечує контроль доступу до особистого кабінету та історії замовлень
SR-04	Наявність особистого кабінету для користувачів	M	Дозволяє користувачам зберігати, оновлювати та видаляти свої дані
SR-05	Модуль управління адміністративною панеллю	M	Надає адміністраторам можливість керувати товарами та замовленнями
SR-06	База даних з історією замовлень	S	Забезпечує відображення історії замовлень для користувачів та адміністраторів
SR-07	Модуль оплати та замовлення	S	Надає можливість здійснення покупок та оплати замовлень через додаток
SR-08	Можливість додавання акцій та пропозицій	C	Забезпечує відображення акцій та спеціальних пропозицій для користувачів

Умовні позначення в таблиці А.2:

Must have (M) – вимоги, які повинні бути реалізовані в системі;

Should have (S) – вимоги, які мають бути виконані, але вони можуть почекати своєї черги;

Could have (C) – вимоги, які можуть бути реалізовані, але вони не є центральною ціллю проекту.

2.4 Вимоги до видів забезпечення

2.4.1 Вимоги до інформаційного забезпечення

Реалізація web-додатку відбувається з використанням наступних технологій та інструментів:

1 Node.js для серверної частини додатку.

2 Express.js для створення веб-сервера та маршрутизації.

3 MongoDB для зберігання даних у NoSQL базі даних.

4 Pug для шаблонізації та відображення сторінок.

5 CSS для стилізації веб-сторінок та покращення їх зовнішнього вигляду.

6 Heroku для розгортання та хостингу вебдодатку у хмарному середовищі.

7 Stripe для оплати, що надає засоби для обробки платежів.

8 GitHub для керування версіями та спільної розробки програмного забезпечення.

Додаток також використовує різні модулі Node.js, такі як Express, MongoDB, Pug, Stripe, Axios, Bcrypt / Bcryptjs, Dotenv, Jsonwebtoken, Mongoose, Nodemailer, Sharp та Validator.

Ці технології дозволяють створити потужний та ефективний web-додаток зі зручним інтерфейсом користувача та надійною базою даних.

2.4.2 Вимоги до лінгвістичного забезпечення

Web-додаток має бути виконаний англійською мовою.

2.4.3 Вимоги до програмного забезпечення

Програмне забезпечення клієнтської частини повинне задовольняти наступним вимогам:

1 Забезпечення сумісності з різними веб-браузерами, включаючи Chrome, Firefox, Safari та Edge.

2 Ефективне використання ресурсів пристрою користувача для швидкого завантаження та відображення сторінок.

3 Підтримка адаптивного дизайну для коректного відображення на різних пристроях, включаючи комп'ютери, планшети та смартфони.

3 СКЛАД І ЗМІСТ РОБІТ ЗІ СТВОРЕННЯ WEB-ДОДАТКУ

Докладний опис етапів роботи зі створення web-додатку наведено в таблиці А.3.

Таблиця А.3 — Етапи створення web-додатку

№	Склад і зміст робіт	Строк розробки
1	Планування та аналіз вимог	5 днів
2	Підготовка технічного завдання та специфікації	3 дні
3	Розробка архітектури додатку	7 днів
4	Налаштування середовища розробки та робота з версійним контролем (GitHub)	4 дні
5	Розробка моделей даних та бази даних MongoDB	6 днів
6	Створення серверної частини додатку на Node.js та Express.js	8 днів
7	Розробка контролерів та маршрутизаторів	4 дні
8	Впровадження аутентифікації та авторизації користувачів	5 днів
9	Створення API для взаємодії з клієнтською частиною	6 днів
10	Розробка клієнтської частини додатку з використанням Pug та CSS	8 днів
11	Інтеграція клієнтської та серверної частин додатку	3 дні
12	Валідація та тестування всіх функцій додатку	4 дні
13	Виправлення помилок та оптимізація коду	3 дні
14	Налаштування хостингу на платформі Heroku	2 дні
15	Розробка інтеграції платіжної системи Stripe	5 днів
16	Підготовка документації та інструкцій для користувачів	3 дні
17	Тестування безпеки та відповідності стандартам	4 дні
18	Підготовка до релізу	4 дні
19	Реліз та розгортання додатку	2 дні

Загальна тривалість робіт: 88 днів.

4 ВИМОГИ ДО СКЛАДУ Й ЗМІСТУ РОБІТ ІЗ ВВЕДЕННЯ WEB-ДОДАТКУ В ЕКСПЛУАТАЦІЮ

Для введення web-додатку в експлуатацію необхідно провести ряд робіт. Це включає тестування та валідацію функціоналу, підготовку документації та інструкцій для користувачів. Потім додаток підготовляється до релізу та розгортання на виробничому середовищі. Після введення в експлуатацію проводиться моніторинг та підтримка додатку, його оптимізація та подальший розвиток.

ДОДАТОК Б

Планування робіт

Деталізація мети проекту методом SMART. Розробка вебдодатка для підтримки діяльності магазину електротехнічних товарів побутового використання є ключовим завданням з метою поліпшення взаємодії з клієнтами та оптимізації процесів покупок. Для досягнення успіху в цьому проекті необхідно чітко сформулювати цілі та завдання за допомогою методу SMART.

SMART-мета: Створити ефективний та зручний вебдодаток для покупок електротехнічних товарів побутового використання з інтуїтивним і зручним інтерфейсом, який забезпечить збільшення кількості покупок та задоволеність клієнтів, і буде завершений у встановлені терміни згідно з календарним планом реалізації.

Результати деталізації методом SMART розміщені у таблиці. Б.1.

Таблиця Б.1 — Деталізація мети методом SMART

Критерій	Опис
Specific	Створити ефективний та зручний вебдодаток для покупок електротехнічних товарів побутового використання з інтуїтивним і зручним інтерфейсом.
Measurable	Оцінка успішності проекту буде здійснена через збільшення кількості покупок та задоволеність клієнтів від використання вебдодатка.
Achievable	Реалізація вебдодатка можлива за допомогою сучасних веб-технологій та програмного забезпечення, доступних для розробників.
Relevant	Розробники мають достатній рівень кваліфікації для створення зазначеного вебдодатка, а необхідні ресурси та технології є наявними.
Time-framed	Проект повинен бути завершений у встановлені терміни, згідно з календарним планом реалізації.

Планування структури проекту:

Для керування проектом створення вебдодатка підтримки діяльності магазину електротехнічних товарів побутового використання використовується WBS діаграма. Цей інструмент дозволяє систематизувати роботу, розподілити завдання та визначити ієрархію виконавчих пакетів. Детальна декомпозиція робіт надає можливість чітко описати кожний етап проекту.

Структура організації для реалізації проекту:

Після побудови WBS розробляється організаційна структура виконавців OBS. Це дозволяє краще організувати внутрішню роботу команди та забезпечити ефективне виконання завдань. Організаційна структура проекту визначає відповідальність та розподіл функцій між учасниками команди, щоб досягти успішного завершення проекту.

Таблиця Б.2 — Виконавці проекту

Роль	Ім’я	Проектна роль
Розробник	Богулов В. Р.	Розроблення вебдодатка та його функціоналу
Дизайнер	Богулов В. Р.	Створення дизайну інтерфейсу користувача
Тестувальник	Богулов В. Р., Чибіряк Я. І.	Проведення тестування функціоналу додатка
Консультант	Чибіряк Я. І.	Надання консультаційних послуг з технічних питань

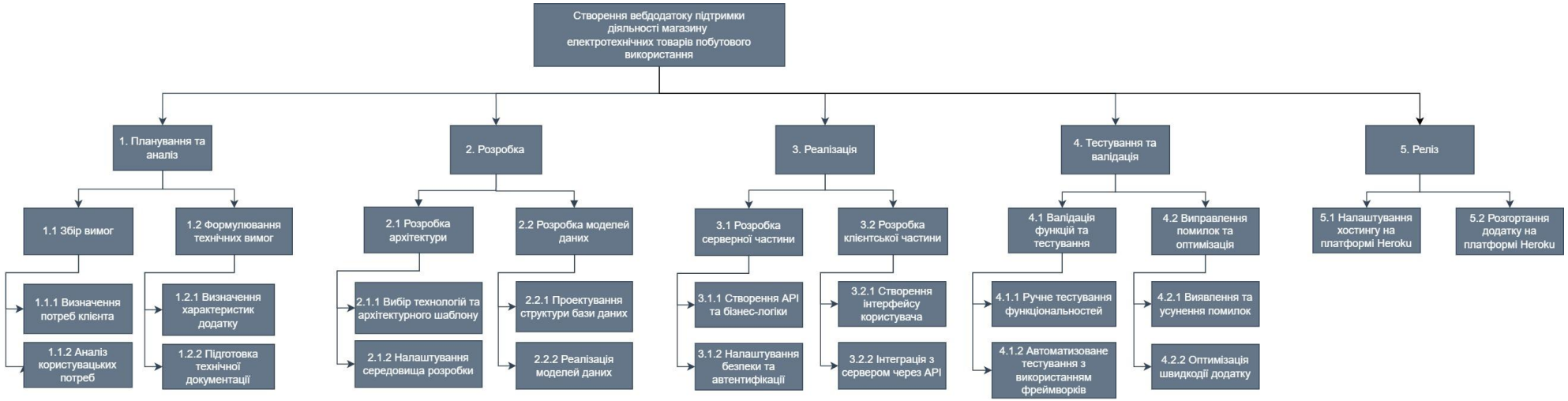

Рисунок Б.1 − WBS. Структура робіт проекту

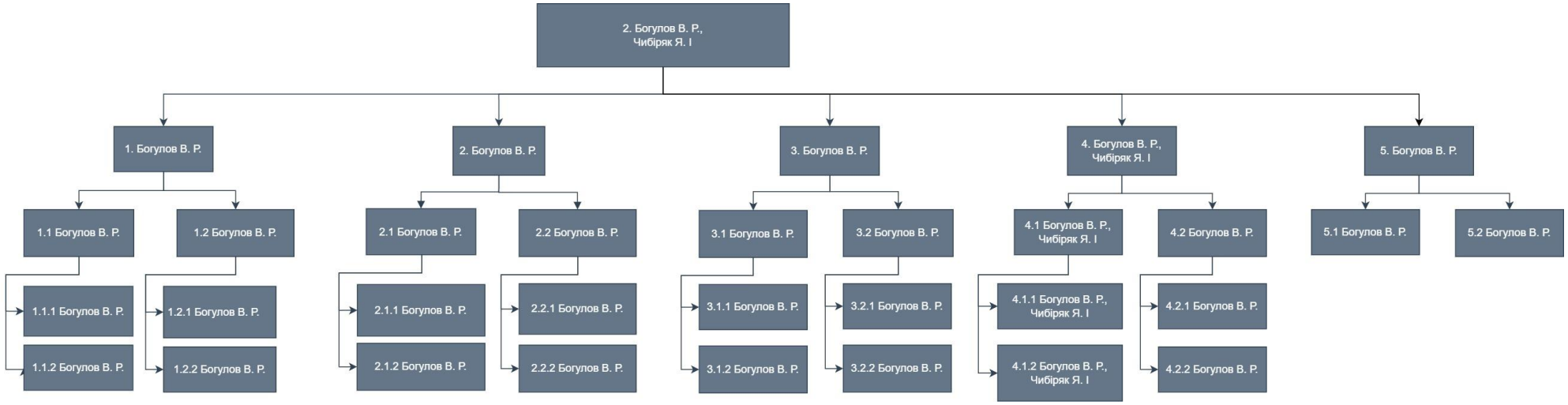

Рисунок Б.2 − Організаційна структура проекту (OBS)

Діаграма Ганта. Розробка вебдодатку для покупок електротехнічних товарів вимагає систематичного підходу та ефективного керування. Використання діаграми Ганта в цьому процесі має кілька вагомих переваг. Діаграма Ганта дозволяє чітко визначити послідовність робіт та встановити критичні шляхи, що сприяє уникненню затримок і оптимізації часу. Цей інструмент допомагає ефективно розподілити ресурси та визначити відповідальних за кожен етап проекту, зменшуючи ризик конфліктів і підвищуючи продуктивність роботи команди. Крім того, він надає можливість відстежувати прогрес виконання завдань та вчасно виявляти можливі відхилення від графіку. Використання діаграми Ганта забезпечує ефективне керування проектом та досягнення його мети вчасно і з мінімальними ризиками. Діаграма зображена на рисунку Б.3-Б.4.

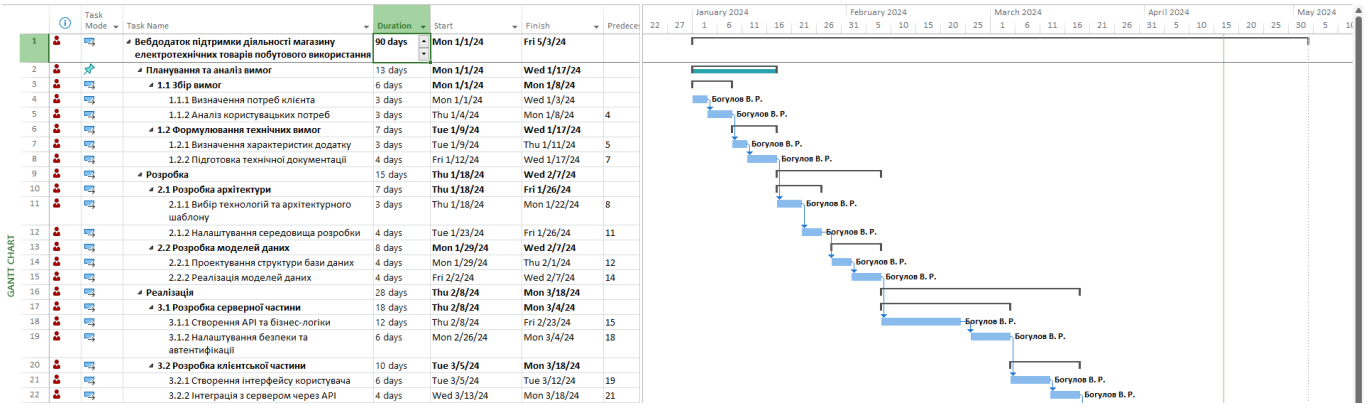

Рисунок Б.3 – Діаграма Ганта

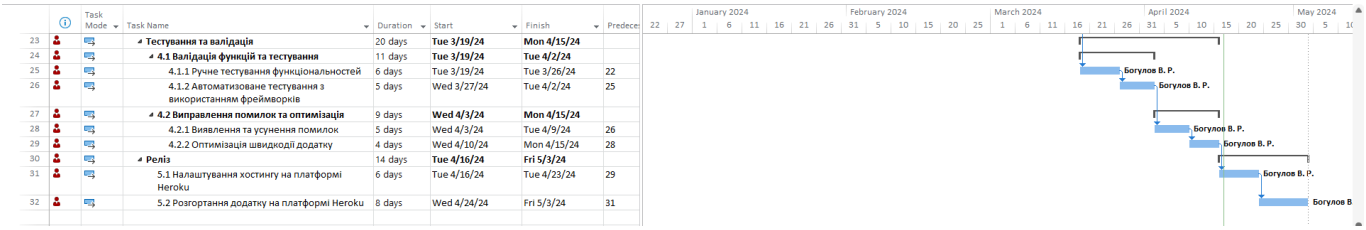

Рисунок Б.4 – Продовження діаграми Ганта

Аналіз ризиків. Аналіз ризиків є критичним етапом у успішному виконанні будь-якого проекту. Під час якісної оцінки ідентифікуються потенційні ризики, які

вимагають негайної уваги. Для кращого розуміння ймовірності виникнення та впливу ризиків застосовується шкала оцінювання (Таблиця Б.3). Крім цього, виявляються менш критичні аспекти ризику, такі як можливі технічні проблеми, які можуть виникнути на пізніших етапах проекту.

Таблиця Б.3 — Шкала оцінювання ймовірності виникнення та впливу ризику на виконання проекту

Оцінка	Ймовірність виникнення	Вплив ризику
1	Низька	Низький
2	Середня	Середній
3	Висока	Високий

Кількісна оцінка ризиків дозволяє більш точно визначити вплив кожного з них на проект. Планування реагування на ризики включає розробку конкретних стратегій та дій з метою мінімізації негативного впливу ризиків на проект. Матриця ймовірності виникнення ризиків та їх впливу є корисним інструментом для цього процесу (таблиця Б.4).

Оцінка ефективності запланованих заходів допомагає зрозуміти, наскільки успішно можна управляти ризиками та як це може вплинути на успішне завершення проекту. Шкала оцінювання за рівнем ризику і Оцінка ймовірності виникнення, величини витрат та індексу ризику надають зручні інструменти для визначення цього ефекту (Таблиці Б.5 і Б.6).

Таблиця Б.4 — Матриця ймовірності виникнення ризиків та впливу ризику

Ймовірність виникнення	3	RS_1, RS_3, RS_6		
	2	RS_4, RS_8	RS_2, RS_11, RS_12, RS_14	
	1	RS_15	RS_9, RS_13	RS_5, RS_7, RS_10
		1	2	3
		Вплив ризику		

– зелений колір – прийнятні ризики;

– жовтий колір – виправданні ризики;

– червоний колір – недопустимі ризики.

На підставі отриманого значення індексу ризики класифікують: за рівнем ризику, що знаходиться в табл. А.4.

Таблиця Б.5 — Шкала оцінювання за рівнем ризику

№	Назва	Межі	Ризики, які входять(номера)
1	Прийнятні	1≤R≤2	4, 8, 9, 13, 15
2	Виправдані	3≤R≤4	1, 2, 3, 5, 6, 7, 10,11, 12, 14
3	Недопустимі	6≤R≤9	

Таблиця Б.6 — Оцінка ймовірності виникнення, величини витрат та індексу ризику

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_1	Відкритий	Невірна оцінка труднощів під час розробки архітектури	Високий	Низький	3	Ретельніше проаналізувати вимоги перед початком розробки	Прийняття	Проведення додаткових оцінок і підготовка резервних планів
RS_2	Відкритий	Проблеми з інтеграцією API з серверною частиною	Середній	Середній	4	Перевірити документацію API та провести тестування на ранніх етапах	Дослідження	Використання альтернативних API або розробка власного
RS_3	Відкритий	Відсутність досвіду в роботі з обраною технологією	Високий	Низький	3	Провести додаткові навчання та тренування перед початком розробки	Перенесення	Використання готових шаблонів та ресурсів для навчання

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_4	Відкритий	Недостатнє тестування під час реалізації клієнтської частини	Середній	Низький	2	Використання автоматизованих тестів та ручних перевірок перед випуском	Згладжування	Розширення тестового покриття та використання додаткових тестових інструментів
RS_5	Відкритий	Відсутність реалізації механізмів безпеки	Низький	Високий	3	Включення механізмів автентифікації та авторизації на ранніх етапах розробки	Перенесення	Застосування готових бібліотек та рішень для забезпечення безпеки
RS_6	Відкритий	Потреба у додатковому функціоналі під час розробки	Високий	Низький	3	Провести оцінку впливу на графік та бюджет проекту	Прийняття	Додаткове оцінювання потреб і можливостей

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_7	Відкритий	Проблеми з оптимізацією швидкодії додатку	Низький	Високий	3	Провести профіліювання та аналіз швидкодії на ранніх етапах	Дослідження	Планування розробки додаткових оптимізацій
RS_8	Відкритий	Проблеми з розподілом робіт та відслідковуванням прогресу	Середній	Низький	2	Впровадження системи управління проектом та регулярні звіти	Згладжування	Використання інструментів для відслідковування прогресу
RS_9	Відкритий	Проблеми з використанням обраної платформи або інструментів	Низький	Середній	2	Провести додатковий аналіз можливостей та обмежень обраної платформи	Згладжування	Міграція на іншу платформу

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_10	Відкритий	Неочікувані зміни вимог під час розробки	Низький	Високий	3	Оцінити вплив на проект та виконати необхідні зміни	Прийняття	Реорганізація процесу збору вимог
RS_11	Відкритий	Неправильна оцінка труднощів під час збору вимог	Середній	Середній	4	Ретельніше проаналізувати вимоги перед початком проекту	Дослідження	Проведення додаткових консультацій з клієнтом
RS_12	Відкритий	Недостатній час для тестування перед випуском	Середній	Середній	4	Планування достатнього часу для тестування на кожному етапі розробки	Згладжування	Використання автоматизованих тестів та планування більш ефективного використання ресурсів

Продовження таблиці Б.6

ID	Статус ризику	Опис ризику	Ймовірність виникнення	Вплив ризику	Ранг ризику	План А	Тип стратегії реагування	План Б
RS_13	Відкритий	Втрата інформації через технічні неполадки	Низький	Середній	2	Розробка плану резервного копіювання та відновлення	Перенесення	Проведення регулярних технічних аудитів та удосконалення системи безпеки
RS_14	Відкритий	Проблеми з інтеграцією сторонніх сервісів	Середній	Середній	4	Ретельне планування та тестування інтеграції на ранніх етапах	Дослідження	Використання альтернативних сервісів або розробка власного рішення
RS_15	Відкритий	Проблеми з доступом до необхідних ресурсів	Низький	Низький	1	Забезпечення резервних джерел та встановлення механізмів контролю доступу	Уникнення	Знаходження альтернативних ресурсів та розробка плану відновлення

ДОДАТОК В

ЛІСТИНГ ПРОГРАМНОГО КОДУ ОСНОВНИХ МОДУЛІВ WEB-ДОДАТКУ

1 JAVASCRIPT FILES

app.js

```
const path = require('path');
const express = require('express');
const cookieParser = require('cookie-parser');
const compression = require('compression');
const cors = require('cors');
const rateLimit = require('express-rate-limit');
const mongoSanitize = require('express-mongo-sanitize');
const xss = require('xss-clean');
const hpp = require('hpp');
const bodyParser = require('body-parser');

const AppError = require('./utils/appError');
const globalErrorHandler = require('./controllers/errorController');

const app = express();

// Configuration
app.set('views', path.join(__dirname, 'views'));
app.set('view engine', 'pug');
app.use(express.static(path.join(__dirname, 'public')));

// Route for webhook-checkout
const transactionController = require('./controllers/transactionController');
app.post(
    '/webhook-checkout',
    bodyParser.raw({ type: 'application/json' }),
    transactionController.webhookCheckout
);

// Middleware
app.use(cookieParser());
app.use(compression());

app.use(bodyParser.json({ limit: '100kb' }));
```

```
app.use(bodyParser.urlencoded({ extended: true, limit: '100kb' }));

app.use(cors());
// Enable CORS for preflight requests (OPTIONS)
app.options('*', cors());

app.use(
    rateLimit({
        max: 1000,
        windowMs: 60 * 1000,
        message:
            'Too many requests from this IP, please try again in a minute!',
    })
);

app.use(mongoSanitize());
app.use(xss());

// Routes
const userRouter = require('./routes/userRoutes');
const productRouter = require('./routes/productRoutes');
const viewRouter = require('./routes/viewRoutes');
const transactionRouter = require('./routes/transactionRoutes');
app.use('/', viewRouter);
app.use('/api/v1/users', userRouter);
app.use('/api/v1/products', productRouter);
app.use('/api/v1/transaction', transactionRouter);

// Error handling
app.all('*', (req, res, next) => {
    next(new AppError(`Can't find ${req.originalUrl} on this server!`, 404));
});
app.use(globalErrorHandler);

module.exports = app;
```

server.js

```
const mongoose = require('mongoose');
require('dotenv').config();

// Import the app instance
const app = require('./app.js');

// Set up error handling for uncaught exceptions and unhandled rejections
process.on('uncaughtException', handleError);
process.on('unhandledRejection', handleError);

// Handle graceful shutdown on SIGTERM signal
```

```
process.on('SIGTERM', () => gracefulShutdown());

async function main() {
    try {
        // Connect to the database
        await connectToDatabase();

        // Start the server with error handling and server object assignment
        const server = await startServer();
    } catch (err) {
        console.error(' ERROR:', err.name, err.message);
        await gracefulShutdown();
        process.exit(1); // Ensure exit after shutdown
    }
}

async function connectToDatabase() {
    try {
        const DB = process.env.DATABASE.replace(
            '<PASSWORD>',
            process.env.DATABASE_PASSWORD
        );
        await mongoose.connect(DB, {});
        console.log('DB connection successful!');
    } catch (err) {
        console.error('Error connecting to DB:', err);
        throw err; // Re-throw for handling in main
    }
}

async function startServer() {
    try {
        const port = process.env.PORT || 3000;
        server = await app.listen(port); // Assuming `app` creates a server object
        console.log(`App running on port ${port}...`);
        return server; // Return the server object for graceful shutdown
    } catch (err) {
        console.error('Error starting server:', err);
        throw err; // Re-throw for handling in main
    }
}

async function gracefulShutdown() {
    console.log(' SIGTERM RECEIVED. Shutting down gracefully');

    // Close the server if it exists
    if (server) {
        await server.close();
        console.log(' Server closed successfully.');
    }
```

```
    console.log(' Process terminated!');
}

async function handleError(err) {
    console.error(' ERROR:', err.name, err.message);
    await gracefulShutdown();
    process.exit(1); // Ensure exit after shutdown
}

// Start the application
(async () => {
    await main();
})();
```

public/js/adminPanel.js

```
import axios from 'axios';
import { showAlert } from './showAlert';

//

//

if (window.location.pathname === '/admin') {
    //

    // API FUNCTIONS

    const table = document.getElementById('admin-panel');

    const deleteProduct = async (productId) => {
        try {
            const res = await axios.delete(`/api/v1/products/${productId}`);
        } catch (error) {
            console.error('Error deleting product:', error);
            showAlert('error', 'Failed to delete product.');
        }
    };

    async function updateProduct(formData) {
        try {
            const productId = formData.get('productId'); // Retrieve product ID from formData
            const res = await axios.patch(
                `/api/v1/products/${productId}`,
                formData
            );
            return res.data.data.product; // Return the updated product object
        } catch (error) {
            console.error('Error updating product data and images:', error);
```

```
        showAlert('error', 'Error updating product.'); // Display error message
        throw error; // Rethrow the error to handle it elsewhere if needed
    }
}

const getAllProducts = async () => {
    try {
        const res = await axios.get(`/api/v1/products/`, {
            params: {
                sort: '-createdAt',
            },
        });

        return res.data.data; // array of products
    } catch (error) {
        console.error('Error fetching products:', error);
        return []; // Return an empty array in case of error
    }
};

const getCategories = async () => {
    try {
        const res = await axios.get(`/api/v1/products/categories`);
        return res.data.data; // array of categories
    } catch (error) {
        console.error('Error fetching categories:', error);
        return []; // Return an empty array in case of error
    }
};

async function createProduct() {
    try {
        const product = {
            name: 'Name',
            price: 0.01,
            quantity: 0,
        };

        const res = await axios.post(`/api/v1/products`, product);
        return res.data.data; // Return the newly created product
    } catch (error) {
        console.error('Error creating product:', error);
        return null;
    }
}

//

//

// RENDERING PRODUCTS TABLE
```

```
            // Create and render a product row
            function renderProductRow(product, categories) {
                const rowHtml = `
            <td><a href="#" class="removeBtn"><i class="fa-solid fa-circle-minus"></i></a></td>
            <td>
              <div class="image-td">
                <div class="image-slider">
                  <div class="slider-content">
                    ${renderProductImages(product.images)}
                  </div>

                  <div class="slider-buttons">
                    ${renderSliderButtons(product.images)}
                  </div>
                </div>
                <div class="upload-btn">
                  <i class="fa-solid fa-file-arrow-up"></i>
                  <input type="file" class="file-upload" style="display:none;" multiple>
              </div>

              </div>
            </td>
            <td><input type="text" name="product-name" value="${
                product.name || ''
            }"></td>
            <td><input type="number" name="product-price" value="${
                product.price !== undefined ? product.price : '0.01'
            }" min="0"></td>
            <td><input type="number" name="product-quantity" value="${
                product.quantity !== undefined ? product.quantity : '0'
            }" min="0"></td>
            <td>$${((product.price || 0) * (product.quantity || 0)).toFixed(2)}</td>
            <td><textarea name="product-description" rows="4">${
                product.description || ''
            }</textarea></td>
            <td><textarea name="product-details" rows="4">${
                product.details || ''
            }</textarea></td>
            <td>
              <div class="dropdown">
                <input type="text" name="product-category" list="category-list" placeholder="Select
or type category" value="${
                    product.category || ''
                }">
                <datalist id="category-list">
                  ${categories
                      .map((category) => `<option value="${category}"></option>`)
                      .join('')}
                </datalist>
              </div>
```

```
            </td>
            <td>${formatDateTime(product.createdAt)}</td>
            <td class="updated-at">${formatDateTime(product.updatedAt)}</td>
        `;

            return rowHtml;
        }

        function formatDateTime(dateTime) {
            return (
                new Date(dateTime).toLocaleDateString() +
                '<br>' +
                new Date(dateTime).toLocaleTimeString([], {
                    hour: '2-digit',
                    minute: '2-digit',
                })
            );
        }

        function renderProductImages(images) {
            if (images.length > 0) {
                return images
                    .map(
                        (image, index) =>
                            `<img src="../img/products/${image}" alt="Product Image ${
                                index + 1
                            }" class="slider-image">`
                    )
                    .join('');
            } else {
                return `<img src="img/placeholder.jpg" alt="Product Image Unavailable"
class="slider-image">`;
            }
        }

        function renderSliderButtons(images) {
            if (images.length > 1) {
                return `
              <div class="slider-btn prev"><i class="fa-solid fa-arrow-left"></i></div>
              <div class="slider-btn next"><i class="fa-solid fa-arrow-right"></i></div>
            `;
            } else {
                return '';
            }
        }

        function populateTable(products, categories) {
            const tbody = document.querySelector('#admin-panel tbody');
            if (!tbody) return;
            tbody.innerHTML = ''; // Clear existing table rows
```

```
    products.forEach((product) => {
        const newRow = document.createElement('tr');
        newRow.setAttribute('data-product-id', product._id);
        newRow.innerHTML = renderProductRow(product, categories);
        tbody.appendChild(newRow);

        handleImageSlider(newRow);
        handleImageUploadClick(newRow);
    });
}

function handleImageSlider(row) {
    // Add event listeners for slider buttons after updating HTML content
    const slider = row.querySelector('.image-slider');
    const sliderContent = slider.querySelector('.slider-content');
    const prevBtn = slider.querySelector('.slider-btn.prev');
    const nextBtn = slider.querySelector('.slider-btn.next');
    const sliderImages = slider.querySelectorAll('.slider-image');
    const imageWidth = 125; // Assuming a fixed image width
    let currentIndex = 0;

    if (prevBtn && nextBtn) {
        const slide = (direction) => {
            if (direction === 'left') {
                currentIndex =
                    currentIndex === 0
                        ? sliderImages.length - 1
                        : currentIndex - 1;
            } else {
                currentIndex =
                    currentIndex === sliderImages.length - 1
                        ? 0
                        : currentIndex + 1;
            }
            const offset = -currentIndex * imageWidth;
            sliderContent.style.transform = `translateX(${offset}px)`;
        };

        prevBtn.addEventListener('click', () => {
            slide('left');
        });

        nextBtn.addEventListener('click', () => {
            slide('right');
        });
    }
}

function handleImageUploadClick(row) {
    const uploadBtn = row.querySelector('.upload-btn i');
    const fileInput = row.querySelector('.file-upload');
```

```
    if (uploadBtn && fileInput) {
        if (!uploadBtn.hasEventListener) {
            uploadBtn.addEventListener('click', () => fileInput.click());
            uploadBtn.hasEventListener = true;
        }
    } else {
        console.error('File upload input not found');
    }
}

//

//

// REMOVING PRODUCTS

async function removeRow(row) {
    const productId = row.dataset.productId;
    try {
        await deleteProduct(productId);
        row.remove();
        showAlert('success', 'Product deleted successfully.');
    } catch (error) {
        console.error('Error removing product:', error);
        showAlert('error', 'Failed to delete the product.');
    }
}

if (table) {
    // Add event listener for remove button click
    table.addEventListener('click', (event) => {
        const removeBtn = event.target.closest('.removeBtn');
        if (removeBtn) {
            event.preventDefault();
            const row = removeBtn.closest('tr');
            removeRow(row);
        }
    });
}

//

//

// UPDATING PRODUCTS

async function updateRowImages(row, product) {
    try {
        const formData = createFormData(product);
        const imageInput = row.querySelector('.file-upload');
```

```
            const files = imageInput.files;
            for (const file of files) {
                formData.append('images', file);
            }
            const updatedProduct = await updateProduct(formData);

            // Update the images in the existing row
            const imageContainer = row.querySelector('.slider-content');
            imageContainer.innerHTML = renderProductImages(
                updatedProduct.images
            );

            const sliderButtons = row.querySelector('.slider-buttons');
            sliderButtons.innerHTML = renderSliderButtons(
                updatedProduct.images
            );

            // Reset the file input
            imageInput.value = '';

            // Reapply event listeners
            handleImageSlider(row);
            handleImageUploadClick(row);
        } catch (error) {
            showAlert('error', 'Failed to update product images');
        }
    }

    function updateRowDate(row) {
        const updatedAtCell = row.querySelector('.updated-at');
        if (updatedAtCell) {
            const currentDate = new Date();
            updatedAtCell.innerHTML = formatDateTime(currentDate);
        }
    }

    async function updateRow(row, eventType) {
        const getInputValue = (selector, type = 'text') =>
            row.querySelector(selector).value;

        const product = {
            productId: row.getAttribute('data-product-id'),
            name: getInputValue('input[name="product-name"]').trim(),
            price: parseFloat(getInputValue('input[name="product-price"]')),
            quantity: parseInt(getInputValue('input[name="product-quantity"]')),
            description: getInputValue(
                'textarea[name="product-description"]'
            ).trim(),
            details: getInputValue('textarea[name="product-details"]').trim(),
            category: getInputValue('input[name="product-category"]').trim(),
        };
```

```
    updateSubTotal(row, product.price, product.quantity);

    if (eventType === 'change') {
        rewriteInvalidFields(row, product);
        updateRowDate(row); // Call the function to update updatedAt row

        const imageInput = row.querySelector('.file-upload');
        const files = imageInput.files;
        if (files.length > 0) {
            await updateRowImages(row, product);
            return;
        } else {
            try {
                const formData = createFormData(product);
                await updateProduct(formData);
            } catch (error) {
                showAlert('error', 'Failed to update product');
            }
        }
    }
}

function rewriteInvalidFields(row, product) {
    const { name, price, quantity } = product;
    const errorMessages = [];

    if (!name || !name.trim()) {
        row.querySelector('input[name="product-name"]').value = 'Name';
        product.name = 'Name';
        errorMessages.push('Product name is required.');
    } else {
        product.name = name.trim();
    }

    if (isNaN(price) || price <= 0) {
        row.querySelector('input[name="product-price"]').value = 0.01;
        product.price = 0.01;
        errorMessages.push('Price must be a positive number.');
    }

    if (quantity < 0 || !Number.isInteger(quantity)) {
        row.querySelector('input[name="product-quantity"]').value = 0;
        product.quantity = 0;
        errorMessages.push('Quantity must be a positive whole number.');
    }

    if (errorMessages.length !== 0) {
        showAlert('error', errorMessages.join('\n'));
    }
}
```

```
function createFormData(product) {
    const formData = new FormData();
    for (const key in product) {
        formData.append(key, product[key]);
    }
    return formData;
}

function updateSubTotal(row, price, quantity) {
    const subTotalCell = row.querySelector('td:nth-child(6)');
    const subtotal =
        !price || !quantity || price <= 0 || quantity <= 0
            ? 0
            : price * quantity;
    subTotalCell.textContent = `$${subtotal.toFixed(2)}`;
}

// initialize inputs for text and areaText fields
if (table) {
    table.addEventListener('input', (event) => {
        const row = event.target.closest('tr');
        if (row) {
            updateRow(row, 'input');
        }
    });

    // Add event listener for change (sending request to server)
    table.addEventListener('change', (event) => {
        const row = event.target.closest('tr');
        if (row) {
            updateRow(row, 'change');
        }
    });
}

//

//

// CREATING PRODUCTS

async function handleProductCreation() {
    try {
        const product = await createProduct();
        const categories = await getCategories();

        const tbody = document.querySelector('#admin-panel tbody');

        // Create a new table row
        const newRow = document.createElement('tr');
```

```
            newRow.setAttribute('data-product-id', product._id);

            newRow.innerHTML = renderProductRow(product, categories);
            tbody.insertAdjacentHTML('afterbegin', newRow.outerHTML);

            // Select the newly created row
            const row = document.querySelector(
                '#admin-panel tbody tr[data-product-id="' + product._id + '"]'
            );

            handleImageSlider(row);
            handleImageUploadClick(row);

            // show allert
            showAlert('success', 'Product created successfully!', true);
        } catch (error) {
            showAlert('error', 'Failed to create product.');
        }
    }

    // Event listener for the "Create Product" button
    const addButton = document.getElementById('createProdBtn');
    if (addButton) addButton.addEventListener('click', handleProductCreation);

    //

    //

    // INITIALIZE

    window.addEventListener('DOMContentLoaded', async () => {
        try {
            const products = await getAllProducts();
            const categories = await getCategories();

            populateTable(products, categories);
        } catch (error) {
            showAlert('error', 'Page initialization error');
        }
    });
}
```

public/js/authModal.js

```
import axios from 'axios';
import { showAlert } from './showAlert';
import { closeAllModals } from './modal-utils.js';
//
```

```
//

// MODAL AUTH

const openAuthModal = async () => {
    const modalAuthEL = document.getElementById('modal-auth');
    modalAuthEL.style.display = 'block'; // Immediately change display to block
    requestAnimationFrame(() => {
        requestAnimationFrame(() => {
            modalAuthEL.classList.add('active');
        });
    });
};
const closeAuthModal = () => {
    const modalAuthEL = document.getElementById('modal-auth');
    modalAuthEL.classList.remove('active');

    setTimeout(() => {
        modalAuthEL.style.display = 'none';
    }, 200); // Match the delay to the CSS transition duration
};

const navAuthEL = document.querySelectorAll('#nav-auth');
navAuthEL.forEach((element) => {
    element.addEventListener('click', (e) => {
        e.preventDefault();

        closeAllModals();
        openAuthModal();
    });
});
const closeAuthModalEL = document.getElementById('close-modal-auth');
if (closeAuthModalEL) {
    closeAuthModalEL.addEventListener('click', (e) => {
        e.preventDefault();
        closeAuthModal();
    });
}
document.addEventListener('keydown', (e) => {
    if (e.key === 'Escape') {
        closeAuthModal();
    }
});

//

//

// Toggle authorization form (login - signup)
const toggleAuthEL = document.querySelectorAll('.toggle-auth');
Array.from(toggleAuthEL).forEach((element) => {
```

```
    element.addEventListener('click', (e) => {
        e.preventDefault();

        const formSignup = document.getElementById('form-signup');
        const formLogin = document.getElementById('form-login');
        formSignup.classList.toggle('hidden');
        formLogin.classList.toggle('hidden');
    });
});

//

//

// LOGIN

const login = async (email, password) => {
    try {
        const res = await axios.post(`/api/v1/users/login`, {
            email,
            password,
        });

        if (res.data.status === 'success') {
            showAlert('success', 'Logged in successfully!', true);

            window.setTimeout(() => {
                window.location.href = window.location.href; // Relocate to the current page
            }, 500);
        }
    } catch (error) {
        showAlert(
            'error',
            error.response?.data?.error?.message || 'An unknown error occurred.'
        );
    }
};

const loginForm = document.getElementById('form-login');
if (loginForm) {
    loginForm.addEventListener('submit', function (e) {
        e.preventDefault();
        const email = document.querySelector('#form-login .email').value;
        const password = document.querySelector('#form-login .password').value;
        login(email, password);
    });
}

//

//
```

```
// FORGET PASSWORD

const forgetPassword = async (email) => {
    try {
        const res = await axios.post(`/api/v1/users/forgetPassword`, {});

        if (res.data.status === 'success') {
            showAlert('success', 'Signup successfully!', true);
            window.setTimeout(() => {
                window.location.href = window.location.href; // Relocate to the current page
            }, 500);
        }
    } catch (error) {
        showAlert(
            'error',
            error.response?.data?.message || 'An unknown error occurred.'
        );
    }
};

//

//

// SIGNUP

const signup = async (name, email, password, passwordConfirm) => {
    try {
        const res = await axios.post(`/api/v1/users/signup`, {
            name,
            email,
            password,
            passwordConfirm,
        });

        if (res.data.status === 'success') {
            showAlert('success', 'Signup successfully!', true);
            window.setTimeout(() => {
                window.location.href = window.location.href; // Relocate to the current page
            }, 500);
        }
    } catch (error) {
        if (error.response?.data?.error?.errors) {
            const errorMessage = Object.values(error.response.data.error.errors)
                .map((error) => error.message)
                .join(', ');
            showAlert('error', errorMessage);
        } else if (error.response?.data?.message) {
            showAlert('error', error.response.data.message);
        } else {
```

```
            showAlert('error', 'An error occurred');
        }
    }
};

const signupForm = document.getElementById('form-signup');
if (signupForm) {
    signupForm.addEventListener('submit', function (e) {
        e.preventDefault();
        const name = document.querySelector('#form-signup .name').value.trim();
        const email = document
            .querySelector('#form-signup .email')
            .value.trim();
        const password = document
            .querySelector('#form-signup .password')
            .value.trim();
        const passwordConfirm = document
            .querySelector('#form-signup .password-confirm')
            .value.trim();
        signup(name, email, password, passwordConfirm);
    });
}

//

//

// LOGOUT

const logout = async () => {
    try {
        const res = await axios.get(`/api/v1/users/logout`, {});

        if (res.data.status === 'success') {
            showAlert('success', 'Logged out successfully!');
            window.setTimeout(() => {
                location.assign('/');
            }, 500);
        }
    } catch (error) {
        showAlert(
            'error',
            error.response?.data?.message || 'An unknown error occurred.'
        );
    }
};

const logoutBtn = document.getElementById('logout');
if (logoutBtn) {
    logoutBtn.addEventListener('click', function (e) {
        e.preventDefault();
```

```
            logout();
        });
    }

    //

    //

    // UPDATE USER

    async function getUser() {
        const res = await axios.get(`/api/v1/users/me`);
        return res.data.data;
    }

    const updateUser = async (data, type = 'data') => {
        const url =
            type === 'password'
                ? '/api/v1/users/updateMyPassword'
                : '/api/v1/users/updateMe';

        try {
            const res = await axios.patch(`${url}`, data);

            if (res.data.status === 'success') {
                showAlert('success', `User data updated successfully!`, true);
            }
        } catch (error) {
            showAlert(
                'error',
                error.response?.data?.message || 'An unknown error occurred.'
            );
        }
    };

    const renderAvatar = (user) => {
        const avatarContainer = document.querySelector(
            '#modal-auth.modal-overlay .modal-container .avatar'
        );
        avatarContainer.innerHTML = `
        ${
            !user.photo.includes('default')
                ? `<img src="./../img/users/${user.photo}" alt="User photo">`
                : `<img src="./../img/users/default.jpg" alt="Default user photo">`
        }
        <div class="menu">
          <input id="photo" type="file" accept="image/*" name="photo" hidden>
          <label for="photo">
            <p id="upload-avatar">Choose a new profile picture <i class="fa-solid fa-rotate"></i>
</p>
          </label>
```

```
      ${
        !user.photo.includes('default')
          ? `
        <p id="delete-avatar">Or revert to default <i class="fa-solid fa-trash"></i> </p>
        `
          : ''
      }
    </div>
  `;

  const deleteAvatarBtn = document.getElementById('delete-avatar');
  if (deleteAvatarBtn) {
    deleteAvatarBtn.addEventListener('click', handleDeleteAvatar);
  }

  const fileInput = document.querySelector(
    '#modal-auth .modal-container .avatar #photo'
  );
  if (fileInput) {
    fileInput.addEventListener('change', handleFileSelection);
  }
};

const handleDeleteAvatar = async () => {
  try {
    const res = await axios.delete(`/api/v1/users/deleteAvatar`);

    if (res.data.status === 'success') {
      showAlert('success', 'Avatar photo deleted successfully!', true);
      renderAvatar(await getUser());
    }
  } catch (err) {
    showAlert('error', 'Failed to delete avatar photo');
  }
};

async function handleFileSelection(event) {
  const selectedFile = event.target.files[0];

  if (selectedFile) {
    const form = new FormData();
    const photoFile = document.querySelector('#photo').files[0];
    if (photoFile) {
      form.append('photo', photoFile);
    }

    await updateUser(form);
    renderAvatar(await getUser());
  }
}
```

```
const initFormSubmission = async () => {
    const form = new FormData();
    form.append(
        'name',
        document.querySelector('#form-update .name').value.trim()
    );
    form.append(
        'email',
        document.querySelector('#form-update .email').value.trim()
    );

    await updateUser(form);
    renderAvatar(await getUser()); // Render avatar after updating user
};

const initUpdateUserForm = async () => {
    const updateUserForm = document.getElementById('form-update');
    if (updateUserForm) {
        updateUserForm.addEventListener('submit', async (e) => {
            e.preventDefault();
            await initFormSubmission();
        });

        renderAvatar(await getUser()); // Render avatar on form initialization
    }
};

window.addEventListener('load', async () => {
    await initUpdateUserForm();
});
```

public/js/cartPage.js

```
import axios from 'axios';
import { showAlert } from './showAlert';
import { closeAllModals } from './modal-utils.js';

if (window.location.pathname === '/cart') {
    // Update product quantity
    const updateProductQuantity = async (itemId, quantity) => {
        try {
            const res = await axios.patch(`/api/v1/users/cart/${itemId}`, {
                quantity,
            });
            if (res.status === 200) {
                showAlert('success', 'Cart updated successfully!');
            } else {
                showAlert('error', 'Failed to update cart.');
            }
```

```
    } catch (error) {
        showAlert('error', 'Failed to update cart.');
    }
};

// Remove item from cart
async function remove(itemId) {
    try {
        const res = await axios.delete(`/api/v1/users/cart/${itemId}`);
        if (res.status === 200) {
            const item = document.querySelector(`
      #cart table tbody tr[data-item-id="${itemId}"]`);
            if (item) {
                item.remove();
                showAlert('success', 'Product removed!');
                updateSubtotal();
            }
        } else {
            showAlert('error', 'Failed to remove product.');
        }
    } catch (error) {
        showAlert('error', 'Failed to remove product.');
    }
}

const table = document.querySelector('#cart table tbody');
const cartSubtotalEl = document.querySelector(
    '#subtotal table tr:first-child td:last-child'
);
const shippingCostEl = document.querySelector(
    '#subtotal table tr:nth-child(2) td:last-child'
);
const totalEl = document.querySelector(
    '#subtotal table tr:last-child td:last-child'
);

let quantity = null;

function validateQty(input, maxQty) {
    quantity = parseInt(input.value);
    if (isNaN(quantity)) {
        quantity = 0;
        showAlert('error', 'Please enter a valid quantity.');
    } else if (quantity < 0) {
        input.value = 0;
        quantity = 0;
        showAlert('error', 'Quantity cannot be negative.');
    } else if (quantity > maxQty) {
        input.value = maxQty;
        quantity = maxQty;
        showAlert('error', 'Quantity exceeds the maximum allowed.');
```

```
    }
    return quantity;
}

async function updateRowDetails(row, eventType) {
    const qtyInput = row.querySelector('input[type="number"]');
    if (qtyInput) {
        const priceEl = row.querySelector('td:nth-child(4)');
        const subtotalEl = row.querySelector('td:last-child');
        const itemId = row.getAttribute('data-item-id');
        const maxQty = parseInt(qtyInput.getAttribute('max'));
        const qty = validateQty(qtyInput, maxQty);
        const price = parseFloat(priceEl.textContent.slice(1));
        const subtotal = qty * price;
        subtotalEl.textContent = `$${subtotal.toFixed(2)}`;
        updateSubtotal();
        if (eventType === 'change') {
            try {
                await updateProductQuantity(itemId, quantity);
                qtyInput.value = quantity;
            } catch (error) {
                showAlert('error', 'Failed to update product quantity.');
            }
        }
    }
}

async function updateSubtotal() {
    let cartSubtotal = 0;
    for (const row of table.querySelectorAll('tr')) {
        const subtotalEl = row.querySelector('td:last-child');
        cartSubtotal += parseFloat(subtotalEl.textContent.slice(1));
    }
    const shippingCost = cartSubtotal === 0 ? 0 : 10;
    cartSubtotalEl.textContent = `$${cartSubtotal.toFixed(2)}`;
    shippingCostEl.textContent = `$${shippingCost.toFixed(2)}`;
    totalEl.textContent = `$${(cartSubtotal + shippingCost).toFixed(2)}`;
}
if (table) {
    window.addEventListener('load', updateSubtotal);
    table.addEventListener('input', (event) => {
        const row = event.target.closest('tr');
        if (row) {
            updateRowDetails(row, 'input');
        }
    });
    table.addEventListener('change', (event) => {
        const row = event.target.closest('tr');
        if (row) {
            updateRowDetails(row, 'change');
        }
```

```
    });

    table.addEventListener('click', (e) => {
        e.preventDefault();
        const removeBtn = e.target.closest('a[href="#"]');
        if (removeBtn) {
            const row = removeBtn.closest('tr');
            const itemId = row.getAttribute('data-item-id');
            remove(itemId);
        }
    });
}

//

//

// CHECKOUT

async function getUser() {
    try {
        const res = await axios.get(`/api/v1/users/me`);
        return res.data.data;
    } catch (error) {
        console.error(
            'Error getting user:',
            error.response ? error.response.data : error.message
        );
        throw error;
    }
}

async function createCheckoutSession(userId) {
    try {
        const response = await axios.get(
            `/api/v1/transaction/checkout-session/${userId}`
        );
        return response.data.session.id;
    } catch (error) {
        showAlert(
            'error',
            'Error creating checkout session:',
            error.response ? error.response.data : error.message
        );
        throw error;
    }
}

document.getElementById('checkout').addEventListener('click', (e) => {
    e.preventDefault();
```

```
    closeAllModals();
    openAddressModal();
});

//

//

// ADDRESS

function validateAddressInputs() {
    const fullName = document.querySelector('input.full-name').value.trim();
    const addressLine1 = document
        .querySelector('input.address-line1')
        .value.trim();
    const addressLine2 = document
        .querySelector('input.address-line2')
        .value.trim();
    const city = document.querySelector('input.city').value.trim();
    const state = document.querySelector('input.state').value.trim();
    const postalCode = document
        .querySelector('input.postal-code')
        .value.trim();
    const country = document.querySelector('input.country').value.trim();

    if (
        fullName &&
        addressLine1 &&
        city &&
        state &&
        postalCode &&
        country
    ) {
        return {
            fullName,
            addressLine1,
            addressLine2,
            city,
            state,
            postalCode,
            country,
        };
    } else {
        showAlert('error', 'Please fill in all required fields.');
        return false;
    }
}

async function updateCartItemAddress(userId, itemId, address) {
    try {
        await axios.patch(`/api/v1/users/cart/${itemId}`, {
```

```
                userId,
                address,
            });
            return true;
        } catch (error) {
            console.error(
                'Error updating address for item:',
                error.response ? error.response.data : error.message
            );
            return false;
        }
    }

    async function updateAddressesForCheckout(userId, cartItems) {
        const address = validateAddressInputs();
        if (!address) return false;

        for (const item of cartItems) {
            if (item.quantity > 0 && !item.purchaseDate) {
                const response = await updateCartItemAddress(
                    userId,
                    item._id,
                    address
                );
                if (!response) {
                    showAlert(
                        'error',
                        'Failed to update address for one or more items.'
                    );
                    return false;
                }
            }
        }
        return true;
    }

    async function proceedToCheckout(userId) {
        const sessionId = await createCheckoutSession(userId);
        if (sessionId) {
            const stripe = Stripe(

'pk_test_51OQAJeQhM6j92pI0XynL7tkiuXOZ5LJKCULZ9bBtM3nZ43xomYlfowhHwflfizosFgI0E
AmEqyL7wHznCxx5b1KA00cCf1bsW7'
            );
            await stripe.redirectToCheckout({ sessionId });
        } else {
            showAlert('error', 'Failed to create checkout session.');
        }
    }

    document
```

```
    .getElementById('form-address')
    .addEventListener('submit', async (e) => {
        e.preventDefault();
        try {
            const user = await getUser();
            if (await updateAddressesForCheckout(user._id, user.cart)) {
                proceedToCheckout(user._id);
            }
        } catch (error) {
            showAlert(
                'error',
                'Failed to update addresses or checkout: ' +
                    (error.response ? error.response.data : error.message)
            );
        }
    });

//

//

// MODAL ADDRESS

const calculateTotals = () => {
    let totalQuantity = 0,
        subtotal = 0;
    document.querySelectorAll('#cart table tbody tr').forEach((row) => {
        const qtyInput = row.querySelector('input[type="number"]');
        const price =
            parseFloat(
                row.querySelector('td:nth-child(4)').textContent.slice(1)
            ) || 0;
        if (qtyInput) {
            totalQuantity += parseInt(qtyInput.value) || 0;
            subtotal += price * (parseInt(qtyInput.value) || 0);
        }
    });
    return { totalQuantity, subtotal };
};

const openAddressModal = () => {
    const modalAddressEL = document.getElementById('modal-address');
    const { totalQuantity, subtotal } = calculateTotals();

    if (totalQuantity > 0 && subtotal > 0.5) {
        modalAddressEL.style.display = 'block'; // Immediately change display to block
        requestAnimationFrame(() => {
            requestAnimationFrame(() => {
                modalAddressEL.classList.add('active');
            });
        });
```

```
        } else {
            if (totalQuantity === 0) {
                showAlert('error', 'No items in the cart to proceed.');
            } else {
                showAlert('error', 'Cart subtotal needs to be more than $0.5.');
            }
        }
    };
    const closeAddressModal = () => {
        const modalAddressEL = document.getElementById('modal-address');
        modalAddressEL.classList.remove('active');

        setTimeout(() => {
            modalAddressEL.style.display = 'none';
        }, 200); // Match the delay to the CSS transition duration
    };

    document
        .getElementById('close-modal-address')
        .addEventListener('click', (e) => {
            e.preventDefault();
            closeAddressModal();
        });

    document.addEventListener('keydown', (e) => {
        if (e.key === 'Escape') {
            closeAddressModal();
        }
    });
}
```

public/js/index.js

```
// toogle the side menu in small screen
const bar = document.getElementById('bar');
const close = document.getElementById('close');
const nav = document.getElementById('navbar');

//

//

// navbar for small screen
if (bar) {
    bar.addEventListener('click', (e) => {
        e.preventDefault();
        nav.classList.add('active');
    });
}
```

```
if (close) {
    close.addEventListener('click', (e) => {
        e.preventDefault();
        nav.classList.remove('active');
    });
}

//

//

//

//

// SHOP PAGE
import './shopPage';

//

//

// AUTORIZATION
import './authModal';

//

//

// CART PAGE
import './cartPage';

//

//

// SINGLE PRODUCT PAGE
import './prodSingle';

//

//

// ADMIN PANEL
import './adminPanel';
```

public/js/modal-utils.js

```
export const closeAllModals = () => {
```

```
    const activeModals = document.querySelectorAll('.modal-overlay.active');

    activeModals.forEach((modal) => {
        modal.classList.remove('active');
        setTimeout(() => {
            modal.style.display = 'none';
        }, 200); // Ensure the display property changes after the transition
    });
};
```

public/js/prodSingle.js

```
import axios from 'axios';
import { showAlert } from './showAlert';

if (window.location.pathname.startsWith('/prod-single')) {
    //

    // TOGGLE IMAGE VIEW
    const mainImg = document.getElementById('main-image');
    const smallImgGroup = document.querySelector('.small-img-groop');

    if (smallImgGroup) {
        const smallImgs = smallImgGroup.querySelectorAll('.small-img');

        if (smallImgs.length > 0) {
            smallImgs[0].classList.add('active');
            mainImg.src = smallImgs[0].src;
        }

        smallImgs.forEach((element) => {
            element.addEventListener('click', () => {
                smallImgs.forEach((img) => img.classList.remove('active'));
                element.classList.add('active');
                mainImg.src = element.src;
            });

            element.addEventListener('mouseover', () => {
                mainImg.src = element.src;
            });

            element.addEventListener('mouseout', () => {
                const lastHoveredImg =
                    smallImgGroup.querySelector('.small-img:hover');
                if (lastHoveredImg) {
                    mainImg.src = lastHoveredImg.src;
                }
            });
        });
```

```
    smallImgGroup.addEventListener('mouseleave', () => {
        const activeImg = smallImgGroup.querySelector('.small-img.active');
        if (activeImg) {
            mainImg.src = activeImg.src;
        }
    });
}

//

//

async function getUser() {
    try {
        const res = await axios.get(`/api/v1/users/me`);
        return res.data.data;
    } catch (error) {
        console.error('Error getting user:', error.message);
        throw error;
    }
}

function getProductId() {
    return document
        .querySelector('#prodetails')
        .getAttribute('data-product-id');
}

//

//

// INPUT QUANTITY AND ADDING TO CART

const addToCart = async (productId, quantity, user) => {
    try {
        const res = await axios.get(`/api/v1/products/${productId}`);

        if (res.data.status === 'success') {
            const product = res.data.data;
            if (product && product.quantity >= quantity && quantity > 0) {
                const resAddProd = await axios.post(`/api/v1/users/cart`, {
                    item: {
                        product_id: product._id,
                        quantity,
                    },
                });

                if (resAddProd.data.status === 'success') {
                    showAlert('success', 'Product added to cart!', true);
```

```
                window.setTimeout(() => {
                    location.assign('/cart');
                }, 500);
            } else {
                showAlert(
                    'error',
                    'Failed to add product to the cart.'
                );
            }
        } else if (quantity <= 0 || isNaN(quantity)) {
            showAlert('error', 'Please enter a valid quantity.');
        }
    }
} catch (error) {
    showAlert('error', error.response.data.message);
}
};

async function handleButtonClick(event) {
    event.preventDefault();

    const quantity = parseInt(quantityInput.value);

    try {
        const user = await getUser();
        await addToCart(getProductId(), quantity, user);
    } catch (error) {
        showAlert('error', 'Please log in to access.');
    }
}

const addToCartBtn = document.querySelector('#prodetails #addToCartBtn');
const quantityInput = document.querySelector(
    '.single-pro-details input[name="quantity"]'
);

function handleInputChange(eventType) {
    let quantity = parseInt(quantityInput.value);
    const maxQuantity = parseInt(quantityInput.getAttribute('max'));

    if (isNaN(quantity)) {
        quantity = 1;
        showAlert('error', 'Please enter a valid quantity.');
    } else if (quantity < 0) {
        quantity = 0;
        showAlert('error', 'Quantity cannot be negative.');
    } else if (quantity > maxQuantity) {
        quantityInput.value = maxQuantity;
        quantity = maxQuantity;
        showAlert('error', 'Quantity exceeds the maximum allowed.');
```

```
    }

    if (eventType === 'change') {
        quantityInput.value = quantity;
    }
}

if (addToCartBtn && quantityInput) {
    addToCartBtn.addEventListener('click', handleButtonClick);
    quantityInput.addEventListener('input', () =>
        handleInputChange('input')
    );
    quantityInput.addEventListener('change', () =>
        handleInputChange('change')
    );
}

//

//

// GET ALL REVIEWS

async function getReviews(productId) {
    try {
        const res = await axios.get(
            `/api/v1/products/${productId}/reviews`
        );
        if (res.data.status === 'success') {
            return res.data.data;
        } else {
            throw new Error('Failed to fetch reviews');
        }
    } catch (error) {
        console.error('Error fetching reviews:', error.message);
        throw error;
    }
}

function formatReviewDate(date) {
    const formattedDate = new Date(date);
    return `${formattedDate.toLocaleDateString()} ${formattedDate.toLocaleTimeString(
        [],
        { hour: '2-digit', minute: '2-digit' }
    )}`;
}

function renderReview(review, currentUserID = null) {
    const isCurrentUser = review.user._id === currentUserID;
    return `
  <div class="review" data-review="${review._id}">
```

```
      <div class="user-info">
        <img class="user-photo" src="/img/users/${
          review.user.photo
        }" alt="User Photo">
        <p>${review.user.name}${isCurrentUser ? ' (you)' : ''}</p>
      </div>
      <div class="review-content">
        <div>
          <p class="stars">${Array(review.rating)
            .fill('<i class="fa-solid fa-star"></i>')
            .join('')}</p>
          <p class="comment">${
            review.comment || 'No comment provided.'
          }</p>
        </div>
        <div class="specifics">
          ${
            isCurrentUser
              ? `
            <div class="options">
              <i class="fa-solid fa-pen-to-square edit"></i>
              <i class="fa-solid fa-trash delete"></i>
            </div>`
              : ''
          }
          <p class="date">${formatReviewDate(
            review.updatedAt > review.createdAt
              ? review.updatedAt
              : review.createdAt
          )}</p>
        </div>
      </div>
    </div>`;
}

function renderAverageRating(reviews) {
    const targetElement = document.querySelector(
        '#prodetails .single-pro-details'
    );
    if (!targetElement) return;

    let avgRatingElement = targetElement.querySelector('.avg-rating');
    if (!avgRatingElement) {
        avgRatingElement = document.createElement('div');
        avgRatingElement.classList.add('avg-rating');
        targetElement.appendChild(avgRatingElement);
    }

    let avgRatingHTML = '';
    if (reviews.length > 0) {
        const avgRating = Math.round(
```

```
                reviews.reduce((acc, cur) => acc + cur.rating, 0) /
                    reviews.length
            );
            for (let i = 0; i < avgRating; i++) {
                avgRatingHTML += '<i class="fa-solid fa-star"></i>';
            }
            avgRatingHTML = `
        <p class="stars">Rating:
          ${avgRatingHTML} (${reviews.length} reviews)
        </p>
      `;
        } else {
            avgRatingHTML = '<p class="stars">No ratings yet</p>';
        }

        avgRatingElement.innerHTML = avgRatingHTML;
    }

    async function renderReviews() {
        try {
            const productId = await getProductId();
            const reviews = await getReviews(productId);
            let currentUserID;
            try {
                currentUserID = (await getUser())._id;
            } catch (getUserError) {
                console.error('Error getting user:', getUserError.message);
            }

            renderAverageRating(reviews);

            const reviewsContainer =
                document.getElementById('reviews-container');
            if (reviews.length > 0) {
                reviewsContainer.innerHTML = reviews
                    .map((review) => renderReview(review, currentUserID))
                    .join('');
            } else {
                reviewsContainer.innerHTML = `<p id="no-reviews-message">Be the first to share your thoughts! Write a review and help others make informed decisions about our product. Your feedback matters!</p>`;
            }
        } catch (error) {
            console.error('Error rendering reviews:', error.message);
        }
    }

    window.addEventListener('load', renderReviews);

    //
```

```
//

// CREATE NEW REVIEW

// Function to toggle star ratings
function toggleStars(clickedRating) {
    let highestRating = 0;
    document.querySelectorAll('[data-rating]').forEach((star) => {
        const starRating = parseInt(star.getAttribute('data-rating'));
        const isActive = starRating <= clickedRating;
        star.classList.toggle('fa-solid', isActive);
        star.classList.toggle('fa-regular', !isActive);
        if (isActive && starRating > highestRating) {
            highestRating = starRating;
        }
    });
    document.querySelectorAll('[data-rating]').forEach((star) => {
        const starRating = parseInt(star.getAttribute('data-rating'));
        star.classList.toggle('active', starRating === highestRating);
    });
}

function getReviewData() {
    const rating = document
        .querySelector('#reviewForm .active')
        .getAttribute('data-rating');
    const comment = document
        .querySelector('#reviewForm .comment')
        .value.trim();
    return { rating, comment };
}

async function createReview(productId, reviewData) {
    try {
        const response = await axios.post(
            `/api/v1/products/${productId}/review`,
            reviewData
        );
        if (response.data.status === 'success') {
            await renderReviews();
            return response.data.message;
        } else {
            throw new Error('Failed to create review');
        }
    } catch (error) {
        showAlert('error', error.response.data.message);
    }
}

// Event listener for star rating
document.querySelectorAll('[data-rating]').forEach((star) => {
```

```
    star.addEventListener('click', function () {
        const clickedRating = parseInt(this.getAttribute('data-rating'));

        toggleStars(clickedRating);
    });
});

// Event listener for form submission
const reviewForm = document.querySelector('#reviewForm');
reviewForm.addEventListener('submit', async (event) => {
    event.preventDefault();
    const reviewData = getReviewData();
    try {
        const productId = getProductId();
        const currentUserID = (await getUser())._id;
        const reviews = await getReviews(productId);

        // Check if the current user has already submitted a review for this product
        const hasUserReviewed = reviews.some(
            (review) => review.user._id === currentUserID
        );
        if (hasUserReviewed) {
            showAlert('error', 'Review already exists for this product.');

            return;
        }

        // If the user hasn't already reviewed, proceed to create the review
        await createReview(productId, reviewData);
        showAlert('success', 'Review created successfully.', true);
    } catch (error) {
        showAlert('error', 'Please log in to access.');
    }
});

//

//

//  DELETE & EDIT REVIEW

// Function to delete a review
async function deleteReview(reviewId) {
    try {
        const response = await axios.delete(
            `/api/v1/products/${getProductId()}/review/${reviewId}`
        );
        if (response.data.status === 'success') {
            showAlert('success', 'Review deleted.');
        } else {
            showAlert('error', 'Failed to delete review');
```

```
            throw new Error('Failed to delete review.');
        }
    } catch (error) {
        console.error('Error deleting review:', error.message);
        throw error; // Rethrow the error for handling in the calling function
    }
}

// Event listener for handling edit and delete actions on reviews
const reviewsContainer = document.getElementById('reviews-container');
reviewsContainer.addEventListener('click', async (event) => {
    const target = event.target;
    const reviewId = target.closest('.review').dataset.review;

    if (target.classList.contains('edit')) {
        handleEdit(reviewId);
    } else if (target.classList.contains('delete')) {
        try {
            await handleDelete(reviewId);
            await renderReviews();
        } catch (error) {
            showAlert('error', error.response.data.message);
        }
    }
});

async function handleEdit(reviewId) {
    // Implement edit functionality here
}

async function handleDelete(reviewId) {
    await deleteReview(reviewId);
}
}
```

public/js/shopPage.js

```
import axios from 'axios';
import { showAlert } from './showAlert';

//

//

if (window.location.pathname === '/shop') {
    //

    // CATEGORIES
```

```
const categoryBtns = document.querySelectorAll('.cat-btn');
categoryBtns.forEach((button) => {
    button.addEventListener('click', toggleCategory);
});

// function resetCategories() {
//   setActiveCategories([]);
//   localStorage.removeItem('categories'); // Clear local storage for categories
// }

function toggleCategory() {
    this.classList.toggle('active');

    currentPage = 1;
    renderProducts();
}

function getActiveCategories() {
    return Array.from(categoryBtns)
        .filter((button) => button.classList.contains('active'))
        .map((button) => button.dataset.category);
}

function setActiveCategories(categories) {
    categoryBtns.forEach((button) => {
        const category = button.dataset.category;
        if (categories.includes(category)) {
            button.classList.add('active');
        } else {
            button.classList.remove('active');
        }
    });
}

function setActiveCategoriesOnLoad() {
    const storedCategories = localStorage.getItem('categories');
    const categories = storedCategories ? storedCategories.split(',') : [];
    setActiveCategories(categories);
}

//

//

// SEARCH BY NAME

let searchName = null;

const searchButton = document.querySelector('#prod-filter .search button');
const clearButton = document.querySelector('#prod-filter .clear i');
const searchInput = document.querySelector(
```

```
    '#prod-filter .search input[type="search"]'
);

searchButton.addEventListener('click', async () => {
    try {
        searchName = searchInput.value.trim() || null;

        currentPage = 1;
        renderProducts();
    } catch (error) {
        showAlert('Failed to search for product by name.');
        console.error(error);
    }
});
clearButton.addEventListener('click', async () => {
    try {
        searchInput.value = null;
        searchName = null;

        currentPage = 1;
        renderProducts();
    } catch (error) {
        showAlert('Failed to clear search.');

        console.error(error);
    }
});

//

//

// SORT ORDER BUTTONS

// Define default sort options
const defaultSortOptions = {
    name: 'none',
    price: 'asc',
    date: 'none',
};

// Retrieve sort options from local storage or use default options
const sortOptions =
    JSON.parse(localStorage.getItem('sortOptions')) || defaultSortOptions;

function toggleSortType(button) {
    const sortType = button.getAttribute('sort-type');
    if (sortType === 'none') {
        return 'asc';
    } else if (sortType === 'asc') {
        return 'desc';
```

```
    } else {
        return 'none';
    }
}

// Function to update sorting options in local storage and UI
function updateSortOptions(button) {
    const sortType = toggleSortType(button);
    button.setAttribute('sort-type', sortType);
    sortOptions[button.getAttribute('name')] = sortType;

    // Hide/show arrow icons based on sort type
    const iconUp = button.querySelector('.asc');
    const iconDown = button.querySelector('.desc');
    iconUp.classList.toggle('hidden', sortType !== 'asc');
    iconDown.classList.toggle('hidden', sortType !== 'desc');

    // Adjust logic to set price option to ascending if all sorting options are none
    const allNone = Object.values(sortOptions).every(
        (value) => value === 'none'
    );
    if (allNone) {
        const priceButton = document.querySelector('[name="price"]');
        priceButton.setAttribute('sort-type', 'asc');
        sortOptions['price'] = 'asc';
        const priceIconUp = priceButton.querySelector('.asc');
        const priceIconDown = priceButton.querySelector('.desc');
        priceIconUp.classList.remove('hidden');
        priceIconDown.classList.add('hidden');
    }

    // Save sort options to local storage
    localStorage.setItem('sortOptions', JSON.stringify(sortOptions));
}

// reset sort options
function resetSortOptions(buttons, currentButton) {
    buttons.forEach((otherButton) => {
        if (otherButton !== currentButton) {
            otherButton.setAttribute('sort-type', 'none');
            const otherIconUp = otherButton.querySelector('.asc');
            const otherIconDown = otherButton.querySelector('.desc');
            otherIconUp.classList.add('hidden');
            otherIconDown.classList.add('hidden');
            sortOptions[otherButton.getAttribute('name')] = 'none';
        }
    });
}

// initialize sorting buttons
function initializeSortingButtons() {
```

```
    const sortingButtons = document.querySelectorAll('.sorting > *');
    sortingButtons.forEach((button) => {
        button.addEventListener('click', () => {
            resetSortOptions(sortingButtons, button);
            updateSortOptions(button);

            currentPage = 1;
            renderProducts(); // Update product list render
        });
    });
}

// update HTML code according to sorting options
function updateSortingButtons() {
    const sortingButtons = document.querySelectorAll('.sorting > *');
    sortingButtons.forEach((button) => {
        const sortType = sortOptions[button.getAttribute('name')];
        button.setAttribute('sort-type', sortType);

        // Hide/show arrow icons based on sort type
        const iconUp = button.querySelector('.asc');
        const iconDown = button.querySelector('.desc');
        iconUp.classList.toggle('hidden', sortType !== 'asc');
        iconDown.classList.toggle('hidden', sortType !== 'desc');
    });
}

function getSortString() {
    let sortField = null;
    for (const key in sortOptions) {
        if (sortOptions[key] !== 'none') {
            sortField = sortOptions[key] === 'asc' ? key : `-${key}`;
            break;
        }
    }
    return sortField;
}

//

//

// PAGINATION

let currentPage = 1;
const intemsPerPage = 5;

// Get filtered products count from the server
const fetchProductsCount = async () => {
    try {
        const fields = 'id';
```

```
        const categories = getActiveCategories().join(',');

        const res = await axios.get(`/api/v1/products`, {
            params: {
                category: categories,
                name: searchName,
                fields,
            },
        });
        return res.data.results;
    } catch (error) {
        showAlert('Failed to fetch product count.');

        console.error(error);
    }
};

// Update pagination UI
const updatePagination = async () => {
    try {
        const productsCount = await fetchProductsCount();
        const totalPages = Math.ceil(productsCount / intemsPerPage);
        const pages = Array.from({ length: totalPages }, (_, i) => i + 1);

        renderPagination(pages);
    } catch (error) {
        if (error) {
            showAlert('Failed to update pagination.');
        }

        console.error(error);
    }
};

// Render pagination buttons
const renderPagination = (pages) => {
    const paginationContainer = document.querySelector('#prod-pagi');
    paginationContainer.innerHTML = ''; // Clear existing pagination buttons

    pages.forEach((page) => {
        const button = createPaginationButton(page);
        paginationContainer.appendChild(button);
    });
};

// Create a pagination button
const createPaginationButton = (page) => {
    const button = document.createElement('p');
    button.textContent = page;
    button.setAttribute('data-page', page);
    button.classList.add('paginate-btn');
```

```
    if (page === currentPage) {
        button.classList.add('active');
    }
    button.addEventListener('click', () => handlePaginationClick(page));
    return button;
};

// Handle pagination button click
const handlePaginationClick = (page) => {
    currentPage = page;
    updateActivePaginationButton(page);
    renderProducts();
};

// Update active pagination button
const updateActivePaginationButton = (page) => {
    const paginationContainer = document.querySelector('#prod-pagi');
    const allButtons =
        paginationContainer.querySelectorAll('.paginate-btn');
    allButtons.forEach((btn) => btn.classList.remove('active'));

    const activeButton = paginationContainer.querySelector(
        `[data-page="${page}"]`
    );
    activeButton.classList.add('active');
};

//

//

// UPDATING PRODUCTS LIST

// Get filtered products from the server
const fetchProducts = async () => {
    try {
        const fields =
            'name, description, price, updatedAt, images, slug, reviews';
        const sort = getSortString().replace('date', 'updatedAt');
        const categories = getActiveCategories().join(',');
        const page = currentPage;

        localStorage.setItem('categories', categories);

        const res = await axios.get(`/api/v1/products`, {
            params: {
                page,
                category: categories,
                sort,
                name: searchName,
                fields,
```

```
                limit: intemsPerPage,
            },
        });
        return res.data.data;
    } catch (error) {
        showAlert('Failed to fetch products. ');

        console.error(error);
    }
};

const renderProducts = async () => {
    try {
        const products = await fetchProducts();
        const productContainer = document.querySelector('.pro-container');

        await updatePagination();

        if (!products || !products.length) {
            productContainer.innerHTML =
                '<div class="no-products"><h4>No products found</h4><p>Our inventory is currently depleted. We\'re working on restocking. Thank you for your patience.</p></div>';
            return;
        }

        productContainer.innerHTML = products
            .map(
                (product) => `
    <div class="pro">
    <img src="${
        product.images?.[0]
            ? `img/products/${product.images[0]}`
            : './img/placeholder.jpg'
    }" alt="${product.name}" width="100%" id="main-image">
    <div class="des">
        <span>${product.name}</span>
        ${
            product.description
                ? `<h5>${product.description}</h5>`
                : '<h5>No description available for this product.</h5>'
        }
        ${
            product.reviews && product.reviews.length
                ? `<p class="stars">${'<i class="fa-solid fa-star"></i>'.repeat(
                      Math.round(
                          product.reviews.reduce(
                              (acc, cur) => acc + cur.rating,
                              0
                          ) / product.reviews.length
                      )
                  )} (${product.reviews.length})</p>`
```

```
                : ''
            }
            ${product.price ? `<h4>$${product.price}</h4>` : ''}
            <a href="/prod-single/${
                product.slug
            }"><i class="fas fa-cart-shopping cart"></i></a>
        </div>
    </div>

    `
            )
            .join('');
    } catch (error) {
        showAlert('Failed to render products.');

        console.error(error);
    }
};

//

//

// INITIALIZE

document.addEventListener('DOMContentLoaded', async () => {
    try {
        initializeSortingButtons();
        updateSortingButtons();

        setActiveCategoriesOnLoad();
        renderProducts();
    } catch (error) {
        showAlert('Failed to initialize page.');

        console.error(error);
    }
});
}
```

public/js/showAlert.js

```
import confetti from 'canvas-confetti';

//

//

const confettiOptions = {
```

```
            particleCount: 30,
            startVelocity: 60,
            ticks: 200,
            zIndex: 1000,
            gravity: 0.5,
            spread: 15 * 2,
        };

        // Left side confetti with angle 120 degrees
        const leftConfettiOptions = {
            ...confettiOptions,
            angle: 90 + 12,
            origin: { x: 1, y: 1 },
        };

        // Right side confetti with angle 240 degrees (symmetrical angle)
        const rightConfettiOptions = {
            ...confettiOptions,
            angle: 90 - 12,
            origin: { x: 0, y: 1 },
        };

        //

        //

        let confettiInterval;

        const fadeAndRemoveAlert = async (id) => {
            try {
                const el = document.getElementById(id);
                if (el) {
                    await new Promise((resolve) => setTimeout(resolve, 2000)); // Delay fade-out
                    el.style.opacity = '0';
                    await new Promise((resolve) => setTimeout(resolve, 1000)); // Delay removal by
additional time after fade-out
                    el.parentElement.removeChild(el);
                }
            } catch (error) {
                console.error('Error in fadeAndRemoveAlert:', error);
            }
        };

        const clearConfetti = () => {
            clearInterval(confettiInterval);
            confetti.reset();
        };

        export const showAlert = async (type, message, shouldThrowConfetti = false) => {
            try {
                const id = Math.random().toString(36).substring(2, 15); // Generate unique ID
```

```
        let icon = '';
        if (type === 'success') {
            icon = '<i class="fa-solid fa-check"></i>';
            if (shouldThrowConfetti) {
                confetti(leftConfettiOptions); // Left side confetti
                confetti(rightConfettiOptions); // Right side confetti
            }
        } else if (type === 'error') {
            icon = '<i class="fa-solid fa-triangle-exclamation"></i>';
        }

        const markup = `<div class="alert alert--${type}" id="${id}">${icon}${
            Array.isArray(message) ? message.join('<br>') : message
        }</div>`;

        document.querySelector('body').insertAdjacentHTML('beforeend', markup);

        await fadeAndRemoveAlert(id);
    } catch (error) {
        console.error('Error in showAlert:', error);
    }
};

// Listen for visibility change
document.addEventListener('visibilitychange', () => {
    if (document.hidden) {
        clearConfetti();
    }
});
```

2 CONTROLLER SCRIPTS

controllers/authController.js

```
const crypto = require('crypto');
const { promisify } = require('util');
const jwt = require('jsonwebtoken');
const User = require('./../models/userModel');
const catchAsync = require('./../utils/catchAsync');
const AppError = require('./../utils/appError');
const sendEmail = require('./../utils/email');

//

//

// TOKEN
```

```
const signToken = (id) => {
    return jwt.sign({ id }, process.env.JWT_SECRET, {
        expiresIn: process.env.JWT_EXPIRES_IN,
    });
};

const createSendToken = (user, statusCode, req, res) => {
    const token = signToken(user._id);
    const expiresIn =
        Date.now() + parseInt(process.env.JWT_COOKIE_EXPIRES_IN, 10);

    res.cookie('jwt', token, {
        expires: new Date(expiresIn),
        httpOnly: true,
        secure: req.secure || req.headers['x-forwarded-proto'] === 'https',
    });
    user.password = undefined;

    res.status(statusCode).json({
        status: 'success',
        token,
        data: {
            user,
        },
    });
};

//

//

// AUTH FUCNTIONS

exports.signup = catchAsync(async (req, res, next) => {
    const newUser = await User.create({
        name: req.body.name,
        email: req.body.email,
        password: req.body.password,
        passwordConfirm: req.body.passwordConfirm,
    }); // cant change the role

    try {
        await sendEmail({
            email: req.body.email,
            subject: 'Welcome to VB SITE!',
            message: `Hello ${req.body.name},\n\nWelcome to VB SITE! We are thrilled to have you on board.\n\nBest regards,\nThe VB SITE Team`,
        });
    } catch (error) {
        console.error('Error sending greeting email:', error);
    }
```

```
    createSendToken(newUser, 201, req, res);
});

exports.login = catchAsync(async (req, res, next) => {
    const { email, password } = req.body;
    if (!email || !password) {
        return next(new AppError('Please provide email and password!', 400));
    }
    const user = await User.findOne({ email }).select('+password');
    if (!user || !(await user.correctPassword(password, user.password))) {
        return next(new AppError('Incorrect email or password', 401));
    }
    createSendToken(user, 200, req, res);
});

exports.logout = (req, res) => {
    res.cookie('jwt', 'loggedout', {
        expires: new Date(Date.now() + 90 * 24 * 60 * 60 * 1000), //90d
        httpOnly: true,
    });
    res.status(200).json({ status: 'success' });
};

//

//

// PROTECTION

exports.protect = catchAsync(async (req, res, next) => {
    // getting token and check of it's there
    let token;
    if (
        req.headers.authorization &&
        req.headers.authorization.startsWith('Bearer')
    ) {
        token = req.headers.authorization.split(' ')[1];
    } else if (req.cookies.jwt) {
        // get token from cookies
        token = req.cookies.jwt;
    }

    if (!token || token === 'loggedout') {
        return next(
            new AppError(
                'You are not logged in! Please log in to get access.',
                401
            )
        );
    }
```

```
    // verification token
    const decoded = await promisify(jwt.verify)(token, process.env.JWT_SECRET);

    // check if user still exists
    const currentUser = await User.findById(decoded.id);
    if (!currentUser) {
        return next(
            new AppError(
                'The user belonging to this token does no longer exist.',
                401
            )
        );
    }

    // check if user changed password after the token was issued
    if (currentUser.changedPasswordAfter(decoded.iat)) {
        return next(
            new AppError(
                'User recently changed password! Please log in again.',
                401
            )
        );
    }

    // GRANT ACCESS TO PROTECTED ROUTE
    req.user = currentUser;
    next();
});

exports.isLoggedIn = async (req, res, next) => {
    if (req.cookies.jwt) {
        try {
            //  verify token
            const decoded = await promisify(jwt.verify)(
                req.cookies.jwt,
                process.env.JWT_SECRET
            );

            //  check if user still exists
            const currentUser = await User.findById(decoded.id);
            if (!currentUser) {
                return next();
            }

            //  check if user changed password after the token was issued
            if (currentUser.changedPasswordAfter(decoded.iat)) {
                return next();
            }

            // THERE IS A LOGGED IN USER
```

```
            req.user = currentUser;
            res.locals.user = currentUser;
            return next();
        } catch (err) {
            return next();
        }
    }
    next();
};

exports.restrictTo = (...allowedRoles) => {
    return (req, res, next) => {
        if (!allowedRoles.includes(req.user.role)) {
            return next(
                new AppError(
                    'You do not have permission to perform this action.',
                    403
                )
            );
        }

        next();
    };
};

//

//

// PASSWORD

exports.forgotPassword = catchAsync(async (req, res, next) => {
    // 1) Get user based on POSTed email
    const user = await User.findOne({ email: req.body.email });
    if (!user) {
        return next(new AppError('There is no user with email address.', 404));
    }

    // 2) Generate the random reset token
    const resetToken = user.createPasswordResetToken();
    await user.save({ validateBeforeSave: false });

    // 3) Send it to user's email
    const resetURL = `${req.protocol}://${req.get(
        'host'
    )}/api/v1/users/resetPassword/${resetToken}`;

    const message = `Forgot your password?\n\nSubmit a patch request with your new password and passwordConfirm to:\n${resetURL}.\n\nYour password reset token is:\n${resetToken}\n\nIf you didn't forget your password, please ignore this email!`;
```

```
    try {
        await sendEmail({
            email: user.email,
            subject: 'Your password reset token (valid for 10 min)',
            message,
        });

        res.status(200).json({
            status: 'success',
            message: 'Token sent to email!',
        });
    } catch (err) {
        user.passwordResetToken = undefined;
        user.passwordResetExpires = undefined;
        await user.save({ validateBeforeSave: false });

        return next(
            new AppError(
                'There was an error sending the email. Try again later!'
            ),
            500
        );
    }
});

exports.resetPassword = catchAsync(async (req, res, next) => {
    // 1) Get user based on the token
    const hashedToken = crypto
        .createHash('sha256')
        .update(req.params.token)
        .digest('hex');

    const user = await User.findOne({
        passwordResetToken: hashedToken,
        passwordResetExpires: { $gt: Date.now() },
    });

    // 2) If token has not expired, and there is user, set the new password
    if (!user) {
        return next(new AppError('Token is invalid or has expired', 400));
    }
    user.password = req.body.password;
    user.passwordConfirm = req.body.passwordConfirm;
    user.passwordResetToken = undefined;
    user.passwordResetExpires = undefined;
    await user.save();

    // 3) Log the user in, send JWT
    createSendToken(user, 200, req, res);
});
```

controllers/cartController.js

```
const multer = require('multer');
const sharp = require('sharp');
const User = require('../models/userModel');
const Product = require('../models/productModel');
const catchAsync = require('../utils/catchAsync');
const AppError = require('../utils/appError');

//

//

exports.getAllItems = catchAsync(async (req, res, next) => {
    const userId = req.params.user_id;
    const user = await User.findById(userId)
        .select('cart') // Retrieve only the 'cart' field
        .populate('cart.product'); // Populate products within the cart

    if (!userId) {
        return next(new AppError('User not found.', 404));
    }

    return res.status(200).json({
        status: 'success',
        data: {
            cart: user.cart,
        },
    });
});

//

//

exports.createItem = catchAsync(async (req, res, next) => {
    const userId = req.user._id;
    const productId = req.body.item.product_id;
    const quantity = req.body.item.quantity;

    const user = await User.findById(userId);
    if (!user) {
        return next(new AppError('User not found.', 404));
    }

    // Check if there are other items with the same product ID that have falsy purchaseDate
value
    const sameProductItems = user.cart.filter(
        (item) => item.product.equals(productId) && !item.purchaseDate
    );
```

```
    if (sameProductItems.length === 0) {
        // If no non-purchased items with the same product ID, add new item
        const newItem = {
            product: productId,
            quantity: quantity,
        };
        user.cart.push(newItem);
    } else {
        // Find the first non-purchased item with the same product ID and update its quantity
        const itemToUpdate = sameProductItems[0];
        itemToUpdate.quantity = quantity;
    }

    await user.save({ validateBeforeSave: false });

    return res.status(200).json({
        status: 'success',
        message: 'Product added to cart successfully.',
        data: {
            cart: user.cart,
        },
    });
});

//

//

exports.deleteAllItems = catchAsync(async (req, res, next) => {
    const userId = req.user._id;

    const user = await User.findById(userId);
    if (!user) {
        return next(new AppError('User not found.', 404));
    }

    user.cart = [];

    await user.save({ validateBeforeSave: false });

    return res.status(200).json({
        status: 'success',
        message: 'All items deleted from cart successfully.',
    });
});

//

//
```

```
exports.getItem = catchAsync(async (req, res, next) => {
    const userId = req.user._id;
    const itemId = req.params.item_id;

    const user = await User.findById(userId);
    if (!user) {
        return next(new AppError('User not found.', 404));
    }

    const item = user.cart.find((item) => item._id == itemId);

    if (!item) {
        return next(new AppError('Item not found in the cart.', 404));
    }

    return res.status(200).json({
        status: 'success',
        data: {
            item,
        },
    });
});

//

//

exports.updateItem = catchAsync(async (req, res, next) => {
    const userId = req.user._id;
    const itemId = req.params.item_id;
    const { quantity, address } = req.body;

    // Find the user by ID
    const user = await User.findById(userId);
    if (!user) {
        return next(new AppError('User not found.', 404));
    }

    // Find the item in the user's cart
    const itemToUpdate = user.cart.find(
        (item) => item._id.toString() === itemId
    );

    if (!itemToUpdate) {
        return next(new AppError('Item not found in the cart.', 404));
    }

    // Update quantity if provided
    if (quantity !== undefined) {
        // Check for undefined to allow setting quantity to 0
        itemToUpdate.quantity = quantity;
```

```
    }

    // Update delivery address if provided
    if (address) {
        const requiredFields = [
            'fullName',
            'addressLine1',
            'city',
            'state',
            'postalCode',
            'country',
        ];
        const isAddressValid = requiredFields.every((field) => address[field]);

        if (!isAddressValid) {
            return next(
                new AppError('Incomplete delivery address provided.', 400)
            );
        }

        itemToUpdate.address = {
            ...itemToUpdate.address, // Preserve existing fields
            ...address,
            addressLine2: address.addressLine2 || '', // Ensure addressLine2 is not undefined
        };
    }

    // Save the updated user document
    await user.save({ validateBeforeSave: false });

    res.status(200).json({
        status: 'success',
        message: 'Cart item updated successfully.',
        data: {
            cart: user.cart,
        },
    });
});

//

//

exports.deleteItem = catchAsync(async (req, res, next) => {
    const userId = req.user._id;
    const itemId = req.params.item_id;

    const user = await User.findById(userId);
    if (!user) {
        return next(new AppError('User not found.', 404));
    }
```

```
    const itemIndex = user.cart.findIndex((item) => item._id == itemId);

    if (itemIndex === -1) {
        return next(new AppError('Item not found in the cart.', 404));
    }

    user.cart.splice(itemIndex, 1);

    await user.save({ validateBeforeSave: false });

    return res.status(200).json({
        status: 'success',
        message: 'Item deleted from cart successfully.',
        data: {
            cart: user.cart,
        },
    });
});
```

controllers/errorController.js

```
const AppError = require('./../utils/appError');

const handleCastErrorDB = (err) => {
    const message = `Invalid ${err.path}: ${err.value}.`;
    return new AppError(message, 400);
};

const handleDuplicateFieldsDB = () => {
    const message = `Duplicate field value. Please use another value!`;
    return new AppError(message, 400);
};

const handleValidationErrorDB = (err) => {
    const errors = Object.values(err.errors).map((el) => el.message);
    const message = `Invalid input data. ${errors.join('. ')}`;
    return new AppError(message, 400);
};

const handleJWTError = () =>
    new AppError('Invalid token. Please log in again!', 401);

const handleJWTExpiredError = () =>
    new AppError('Your token has expired! Please log in again.', 401);

const sendError = (err, req, res) => {
    const statusCode = err.statusCode || 500;
    const status = err.status || 'error';
```

```
    if (req.originalUrl.startsWith('/api')) {
        res.status(statusCode).json({
            status,
            error: err,
            message: err.message,
            stack:
                process.env.NODE_ENV === 'development' ? err.stack : undefined,
        });
    } else {
        console.error('ERROR', err);
        res.status(statusCode).render('error', {
            title: 'Something went wrong!',
            msg: err.isOperational ? err.message : 'Please try again later.',
        });
    }
};

module.exports = (err, req, res, next) => {
    err.statusCode = err.statusCode || 500;
    err.status = err.status || 'error';

    let error = { ...err };
    error.message = err.message;

    if (error.name === 'CastError') error = handleCastErrorDB(error);
    if (error.code === 11000) error = handleDuplicateFieldsDB();
    if (error.name === 'ValidationError')
        error = handleValidationErrorDB(error);
    if (error.name === 'JsonWebTokenError') error = handleJWTError();
    if (error.name === 'TokenExpiredError') error = handleJWTExpiredError();

    sendError(error, req, res);
};
```

controllers/handlerFactory.js

```
const catchAsync = require('./../utils/catchAsync');
const AppError = require('./../utils/appError');
const APIFeatures = require('./../utils/apiFeatures');

exports.deleteOne = (Model) =>
    catchAsync(async (req, res, next) => {
        const doc = await Model.findByIdAndDelete(req.params.id);

        if (!doc) {
            return next(new AppError('No document found with that ID', 404));
        }
```

```
        res.status(200).json({
            status: 'success',
            message: 'Document deleted successfully',
            data: doc,
        });
    });

exports.updateOne = (Model) =>
    catchAsync(async (req, res, next) => {
        let updateData = req.body;
        if (Model.schema.obj.updatedAt) {
            updateData.updatedAt = Date.now();
        }

        const doc = await Model.findByIdAndUpdate(req.params.id, updateData, {
            new: true,
            runValidators: true,
        });

        if (!doc) {
            return next(new AppError('No document found with that ID', 404));
        }

        res.status(200).json({
            status: 'success',
            message: 'Document updated successfully',
            data: doc,
        });
    });

exports.createOne = (Model) =>
    catchAsync(async (req, res, next) => {
        const doc = await Model.create(req.body);

        res.status(201).json({
            status: 'success',
            message: 'Document created successfully',
            data: doc,
        });
    });

exports.getOne = (Model, popOptions) =>
    catchAsync(async (req, res, next) => {
        let query = Model.findById(req.params.id);
        if (popOptions) query = query.populate(popOptions);
        const doc = await query;

        if (!doc) {
            return next(new AppError('No document found with that ID', 404));
        }
```

```
        res.status(200).json({
            status: 'success',
            message: 'Document found',
            data: doc,
        });
    });

exports.getAll = (Model) =>
    catchAsync(async (req, res, next) => {
        let filter = {};
        if (req.params.productId) filter = { product: req.params.productId };

        const features = new APIFeatures(Model.find(filter), req.query)
            .filter()
            .sort()
            .limitFields()
            .paginate();
        const doc = await features.query;

        res.status(200).json({
            status: 'success',
            message: 'Documents retrieved',
            results: doc.length,
            data: doc,
        });
    });

exports.deleteAll = (Model) =>
    catchAsync(async (req, res, next) => {
        const features = new APIFeatures(Model.find(), req.query)
            .filter()
            .sort()
            .limitFields()
            .paginate();

        const doc = await features.query;
        const deletedDocs = await Model.deleteMany();

        res.status(200).json({
            status: 'success',
            message: 'Documents deleted',
            results: deletedDocs.deletedCount,
            data: doc,
        });
    });
```

controllers/productController.js

```
const multer = require('multer');
```

```
const sharp = require('sharp');
const Product = require('../models/productModel');
const catchAsync = require('../utils/catchAsync');
const AppError = require('../utils/appError');
const factory = require('./handlerFactory');
const mongoose = require('mongoose');
const fs = require('fs');

//

//

// UPDATE IMAGES

const multerStorage = multer.memoryStorage();
const multerFilter = (req, file, cb) => {
    if (file.mimetype.startsWith('image')) {
        cb(null, true);
    } else {
        cb(
            new AppError('Not an image! Please upload only images.', 400),
            false
        );
    }
};

const upload = multer({
    storage: multerStorage,
    fileFilter: multerFilter,
    limits: {
        files: 4,
    },
});

exports.uploadProductImages = upload.array('images', 4);

// Middleware to resize product images
exports.resizeProductImages = async (req, res, next) => {
    if (!req.files || req.files.length === 0) {
        return next();
    }

    try {
        // Initialize req.body.images as an empty array if it's not already defined
        req.body.images = req.body.images || [];
        // Resize and save the new uploaded photos
        await Promise.all(
            req.files.map(async (file, index) => {
                const imagePath = `public/img/products/product-${
                    req.params.id
                }-${Date.now()}-${index}.jpeg`;
```

```
            await sharp(file.buffer)
                .resize(800, 600)
                .toFormat('jpeg')
                .jpeg({ quality: 90 })
                .toFile(imagePath);
            // Push the filename to req.body.images array for update
            req.body.images.push(imagePath.split('/').pop());
        })
    );
    next();
  } catch (err) {
    return next(new AppError('Error resizing product images.', 500));
  }
};

exports.updateProduct = async (req, res, next) => {
  try {
    const productId = req.params.id;
    const updateData = req.body;

    // Update the 'updatedAt' field with the current date
    updateData.updatedAt = Date.now();

    // Check if there are new photos uploaded
    if (req.files && req.files.length > 0) {
      // Find the product by ID to get the old images
      const product = await Product.findById(productId);

      // Delete old images if they exist
      if (product.images && product.images.length > 0) {
        await Promise.all(
          product.images.map(async (image) => {
            const imagePath = `public/img/products/${image}`;
            if (fs.existsSync(imagePath)) {
              await fs.promises.unlink(imagePath);
            }
          })
        );
      }
    }

    const updatedProduct = await Product.findByIdAndUpdate(
      productId,
      updateData,
      {
        new: true, // Return the updated document
        runValidators: true,
      }
    );

    if (!updatedProduct) {
```

```
            return next(new AppError('No product found with that ID', 404));
        }

        return res.status(200).json({
            status: 'success',
            data: {
                product: updatedProduct,
            },
        });
    } catch (err) {
        return next(new AppError('Error updating product.', 500));
    }
};

//

//

// GET ALL UNIQUE CATEGORIES

exports.getAllUniqueCategories = catchAsync(async (req, res, next) => {
    try {
        const categories = await Product.aggregate([
            { $group: { _id: '$category' } },
            { $match: { _id: { $ne: null } } }, // Filter out null categories
            { $project: { _id: 0, category: '$_id' } }, // Reshape the output
            { $sort: { category: 1 } }, // Sort by the category field in ascending order
        ]);

        return res.status(200).json({
            status: 'success',
            data: categories.map((category) => category.category),
        });
    } catch (error) {
        return res.status(500).json({
            status: 'fail',
            message: 'Internal server error.',
        });
    }
});

//

//

// REVIEWS

exports.getReviews = catchAsync(async (req, res, next) => {
    const productId = req.params.id;

    const reviews = await Product.aggregate([
```

```
        { $match: { _id: new mongoose.Types.ObjectId(productId) } },
        { $unwind: '$reviews' },
        {
            $lookup: {
                from: 'users',
                localField: 'reviews.user',
                foreignField: '_id',
                as: 'reviews.user',
            },
        },
        { $unwind: '$reviews.user' },
        {
            $project: {
                _id: '$reviews._id', // Include the review's _id
                'user._id': '$reviews.user._id', // Include the user's _id
                'user.name': '$reviews.user.name',
                'user.photo': '$reviews.user.photo',
                rating: '$reviews.rating',
                comment: '$reviews.comment',
                createdAt: '$reviews.createdAt',
                updatedAt: '$reviews.updatedAt',
            },
        },
    ]);

    return res.status(200).json({
        status: 'success',
        data: reviews,
    });
});

//

//

exports.createReview = catchAsync(async (req, res, next) => {
    const productId = req.params.id;
    const { rating, comment } = req.body;
    const userId = req.user._id;

    const product = await Product.findById(productId);

    if (!product) {
        return next(new AppError('Product not found.', 404));
    }

    product.reviews.push({ user: userId, rating, comment });
    await product.save();

    return res.status(201).json({
        status: 'success',
```

```
        message: 'Review added successfully.',
    });
});

//

//

exports.deleteReview = catchAsync(async (req, res, next) => {
    const productId = req.params.id;
    const reviewId = req.params.review_id;
    const userId = req.user._id;

    const product = await Product.findById(productId);

    if (!product) {
        return next(new AppError('Product not found.', 404));
    }

    const review = product.reviews.find((review) =>
        review._id.equals(reviewId)
    );

    if (!review) {
        return next(new AppError('Review not found for this product.', 404));
    }

    product.reviews.pull(reviewId); // Remove the review by its ID
    await product.save();

    return res.status(200).json({
        status: 'success',
        message: 'Review deleted successfully.',
    });
});

//

//

// OTHER GENERIC API FUNCTIONS

exports.getProduct = factory.getOne(Product);
exports.getAllProducts = factory.getAll(Product);
exports.createProduct = factory.createOne(Product);
exports.deleteProduct = factory.deleteOne(Product);
exports.deleteAllProducts = factory.deleteAll(Product);
```

controllers/transactionController.js

```
const stripe = require('stripe')(process.env.STRIPE_SECRET_KEY);
const User = require('../models/userModel');
const Product = require('../models/productModel');
const catchAsync = require('../utils/catchAsync');
const AppError = require('../utils/appError');

//

//

exports.getCheckoutSession = catchAsync(async (req, res, next) => {
    // 1) Get the user
    const user = await User.findById(req.user.id).populate('cart.product');

    // 2) Check if user and cart exist and have items
    if (!user || !user.cart || user.cart.length === 0) {
        return next(new AppError('Shopping cart is empty.', 400));
    }

    // Filter out purchased items and items with quantity less than 1
    const filteredItems = user.cart.filter(
        (item) => !item.purchaseDate && item.quantity > 0
    );

    // Check if any filtered product in cart has a quantity less than 1
    if (filteredItems.some((item) => item.product.quantity < 1)) {
        return next(new AppError('Some products are out of stock', 400));
    }

    // Check if no products are left after filtering
    if (filteredItems.length === 0) {
        return next(new AppError('Shopping cart is empty.', 400));
    }

    // Ensure proper data formatting for line items
    const lineItems = filteredItems.map((item) => ({
        price_data: {
            currency: 'usd',
            product_data: {
                name: item.product.name,
                description:
                    item.product.description ||
                    'No description available for this product.', // Set string if not defined
                images: item.product.images.map(
                    (image) =>
                        `${req.protocol}://${req.get(
                            'host'
                        )}/img/products/${image}`
                ),
            },
```

```
                unit_amount: Math.round(item.product.price * 100), // Convert to integer
            },
            quantity: item.quantity,
        }));

        //

        // Create checkout session with `mode` parameter
        const session = await stripe.checkout.sessions.create({
            payment_method_types: ['card'],
            success_url: `${req.protocol}://${req.get('host')}/cart`, // Add successfullyPurchased query parameter
            cancel_url: `${req.protocol}://${req.get('host')}/cart`,
            customer_email: req.user.email,
            line_items: lineItems,
            mode: 'payment', // Replace with 'subscription' for subscriptions
            metadata: {
                cartItems: JSON.stringify(filteredItems.map((item) => item._id)), // Pass cart item IDs as metadata
            },
        });

        // Create session as response
        res.status(200).json({
            status: 'success',
            session,
        });
    });

    //

    //

    exports.webhookCheckout = async (req, res, next) => {
        const signature = req.headers['stripe-signature'];

        let event;
        try {
            event = stripe.webhooks.constructEvent(
                req.body,
                signature,
                process.env.STRIPE_WEBHOOK_SECRET
            );
        } catch (err) {
            console.error('Error in webhook signature verification:', err.message);
            return res.status(400).send(`Webhook error: ${err.message}`);
        }

        if (event.type === 'checkout.session.completed') {
            try {
                await handleCheckoutCompletion(event.data.object);
```

```
            res.status(200).json({ received: true });
        } catch (error) {
            console.error('Error handling checkout completion:', error.message);
            res.status(error.statusCode || 500).json({ error: error.message });
        }
    } else {
        res.status(200).json({ received: false });
    }
};

const handleCheckoutCompletion = async (sessionData) => {
    const { customer_email, metadata } = sessionData;

    if (!customer_email || !metadata || !metadata.cartItems) {
        throw new AppError('Invalid request data', 400);
    }

    const itemIds = JSON.parse(metadata.cartItems);

    try {
        const user = await User.findOne({ email: customer_email });
        if (!user) {
            throw new AppError('User not found', 404);
        }

        await markItemsPurchased(user, itemIds);
    } catch (error) {
        console.error(
            'Error occurred while handling checkout completion:',
            error
        );
        throw error;
    }
};

const markItemsPurchased = async (user, itemIds) => {
    await Promise.all(
        itemIds.map(async (itemId) => {
            const cartItem = user.cart.find(
                (cartItem) => cartItem._id.toString() === itemId
            );
            if (cartItem) {
                // Update purchase date
                cartItem.purchaseDate = new Date();
                await updateUserCart(user, cartItem);
                await updateProductStock(cartItem.product, cartItem.quantity);
            }
        })
    );

    await user.save({ validateBeforeSave: false });
```

```
    console.log('User cart saved successfully:', user);
};

const updateUserCart = async (user, cartItem) => {
    // Update purchase date in user's cart
    const index = user.cart.findIndex(
        (item) => item._id.toString() === cartItem._id.toString()
    );
    user.cart[index].purchaseDate = cartItem.purchaseDate;
};

const updateProductStock = async (productId, quantity) => {
    // Decrement product stock
    const product = await Product.findById(productId);
    if (!product) {
        throw new AppError('Product not found', 404);
    }
    console.log('\n\nProduct found:', product);
    console.log('\n\nquantity minus:', quantity);
    product.quantity -= quantity;
    await product.save();
    console.log('\n\nProduct stock updated:', product);
};
```

controllers/userController.js

```
const multer = require('multer');
const sharp = require('sharp');
const User = require('./../models/userModel');
const catchAsync = require('./../utils/catchAsync');
const AppError = require('./../utils/appError');
const factory = require('./handlerFactory');
const fs = require('fs');

//

//

// UPLOAD USER AVATAR PHOTO

const multerStorage = multer.memoryStorage();

const multerFilter = (req, file, cb) => {
    if (file.mimetype.startsWith('image')) {
        cb(null, true);
    } else {
        cb(
            new AppError('Not an image! Please upload only images.', 400),
            false
```

```
        );
    }
};

const upload = multer({
    storage: multerStorage,
    fileFilter: multerFilter,
});

exports.uploadUserPhoto = upload.single('photo');

exports.resizeUserPhoto = catchAsync(async (req, res, next) => {
    if (!req.file) {
        return next();
    }

    req.file.filename = `user-${req.user.id}-${Date.now()}.jpeg`;

    await sharp(req.file.buffer)
        .resize(500, 500)
        .toFormat('jpeg')
        .jpeg({ quality: 90 })
        .toFile(`public/img/users/${req.file.filename}`);

    next();
});

//

//

const filterObj = (obj, ...allowedFields) => {
    const newObj = {};
    Object.keys(obj).forEach((el) => {
        if (allowedFields.includes(el)) newObj[el] = obj[el];
    });
    return newObj;
};

exports.getMe = (req, res, next) => {
    req.params.id = req.user._id;
    next();
};

exports.updateMe = catchAsync(async (req, res, next) => {
    // 1) Create error if user POSTs password data
    if (req.body.password || req.body.passwordConfirm) {
        return next(
            new AppError(
                'This route is not for password updates. Please use /updateMyPassword.',
                400
```

```
            )
        );
    }

    // 2) Filtered out unwanted fields names that are not allowed to be updated
    const filteredBody = filterObj(req.body, 'name', 'email');
    if (req.file) {
        filteredBody.photo = req.file.filename;

        // Check if there's an old photo to delete
        if (req.user.photo !== 'default.jpg') {
            const filePath = `./public/img/users/${req.user.photo}`;
            fs.unlink(filePath, (err) => {
                if (err) {
                    console.error('Error deleting old photo:', err);
                }
            });
        }
    }

    // 3) Update user document
    const updatedUser = await User.findByIdAndUpdate(
        req.user.id,
        filteredBody,
        {
            new: true,
            runValidators: true,
        }
    );

    return res.status(200).json({
        status: 'success',
        data: {
            user: updatedUser,
        },
    });
});

exports.deleteAvatar = catchAsync(async (req, res, next) => {
    const user = req.user;
    const fileToDelete = req.user.photo;
    const filePath = `./public/img/users/${fileToDelete}`;

    // Delete the file from the server
    fs.unlink(filePath, async (err) => {
        if (err) {
            console.error('Error deleting file:', err);
        } else {
            // Update user's photo to default
            req.user.photo = 'default.jpg';
            await user.save({ validateBeforeSave: false });
```

```
            return res.status(200).json({
                status: 'success',
                data: {
                    user: req.user,
                    fileToDelete,
                },
            });
        }
    });
});

exports.deleteMe = catchAsync(async (req, res, next) => {
    const user = await User.findByIdAndUpdate(req.user.id, { active: false });

    return res.status(200).json({
        status: 'success',
        data: user,
    });
});

exports.createUser = (req, res) => {
    return res.status(500).json({
        status: 'error',
        message: 'This route is not defined! Please use /signup instead',
    });
};

//

//

exports.getUser = factory.getOne(User);
exports.getAllUsers = factory.getAll(User);
exports.deleteUser = factory.deleteOne(User);
exports.deleteAllUsers = factory.deleteAll(User);

// Do NOT update passwords with this!
exports.updateUser = factory.updateOne(User);
```

controllers/viewsController.js

```
const axios = require('axios');
const path = require('path');

const Product = require('../models/productModel');
const AppError = require('../utils/appError');
const catchAsync = require('../utils/catchAsync');

//
```

```
//

exports.getHome = catchAsync(async (req, res, next) => {
    res.status(200).render('home', {
        title: 'Home',
    });
});

exports.getShop = catchAsync(async (req, res, next) => {
    const baseURL = req.protocol + '://' + req.get('host');
    const categories = await axios.get(baseURL + '/api/v1/products/categories');

    res.status(200).render('shop', {
        title: 'Shop',
        categories: categories.data.data,
    });
});

exports.getSProduct = catchAsync(async (req, res, next) => {
    const product = await Product.findOne({
        slug: req.params.slug,
    });

    if (!product) {
        return next(new AppError('There is no product with that name.', 404));
    }

    res.status(200).render('prod-single', {
        title: product.name,
        product,
    });
});

exports.getBlog = catchAsync(async (req, res, next) => {
    res.status(200).render('blog', {
        title: 'Blog',
    });
});

exports.getAbout = catchAsync(async (req, res, next) => {
    res.status(200).render('about', {
        title: 'About',
    });
});

exports.getContact = catchAsync(async (req, res, next) => {
    res.status(200).render('contact', {
        title: 'Contact',
    });
});
```

```
exports.getCart = catchAsync(async (req, res, next) => {
    const baseURL = req.protocol + '://' + req.get('host');
    const user = await axios.get(
        baseURL + `/api/v1/users/cart/${req.user._id}`
    );

    const cart = user.data.data.cart.reduce((acc, item) => {
        if (item.purchaseDate) {
            const date = new Date(item.purchaseDate).toLocaleDateString(
                'en-US',
                {
                    month: 'short',
                    day: 'numeric',
                    year: 'numeric',
                }
            );
            acc[date] = acc[date] || { date, items: [] };
            acc[date].items.push(item);
        }
        return acc;
    }, {});

    const groupedPurchases = Object.values(cart)
        .sort((b, a) => new Date(a.date) - new Date(b.date))
        .map((group) => {
            group.items.sort(
                (b, a) => new Date(a.purchaseDate) - new Date(b.purchaseDate)
            );
            return group;
        });

    res.status(200).render('cart', {
        title: 'Cart',
        cart: user.data.data.cart,
        groupedPurchases,
    });
});

exports.getAdmin = catchAsync(async (req, res, next) => {
    res.status(200).render('admin', {
        title: 'Admin',
    });
});
```

3 MODEL SCRIPTS

models/productModel.js

```
const mongoose = require('mongoose');
```

```
const slugify = require('slugify');
const AppError = require('./../utils/appError');
const User = require('./userModel');
const fs = require('fs').promises;
const path = require('path');

const productSchema = new mongoose.Schema({
    reviews: [
        {
            user: {
                type: mongoose.Schema.Types.ObjectId,
                ref: 'User',
                required: [true, 'User reference error'],
            },
            rating: {
                type: Number,
                required: [true, 'Please enter rating stars'],
                min: 1,
                max: 5,
            },
            comment: {
                type: String,
                default: '',
                maxlength: 300,
            },
            updatedAt: {
                type: Date,
                default: Date.now(),
            },
            createdAt: {
                type: Date,
                default: Date.now(),
            },
        },
    ],
    name: {
        type: String,
        require: [true, 'Please enter product name'],
    },
    description: {
        type: String,
        required: false,
    },
    details: {
        type: String,
        required: false,
    },
    quantity: {
        type: Number,
        require: [true, 'Please enter product quantity'],
        default: 1,
```

```
        min: 0,
    },
    price: {
        type: Number,
        require: [true, 'Please enter product price'],
        min: 0,
    },
    discount: {
        type: Number,
        require: false,
        min: 0,
        max: 1,
    },
    category: {
        type: String,
        require: false,
    },
    updatedAt: {
        type: Date,
        default: undefined,
    },
    createdAt: {
        type: Date,
        default: undefined,
    },
    images: {
        type: [String],
        required: false,
    },
    slug: {
        type: String,
        required: false,
    },
});

productSchema.index({ slug: 1 });

//

//

// runs before .save() and .create()
productSchema.pre('save', async function (next) {
    const currentDate = new Date();
    this.updatedAt = currentDate;
    if (!this.createdAt) {
        this.createdAt = currentDate;
    }

    if (this.isModified('name')) {
        this.slug = slugify(this.name, { lower: true });
```

```
        // Check if the slug already exists
        const slugRegex = new RegExp(`^${this.slug}(-[0-9]+)?$`, 'i');
        const productsWithSimilarSlug = await this.constructor.find({
            slug: slugRegex,
        });

        if (productsWithSimilarSlug.length > 0) {
            // If similar slugs exist, append a unique identifier
            this.slug = `${this.slug}-${productsWithSimilarSlug.length + 1}`;
        }
    }
    next();
});

//

//

productSchema.pre(/update/i, async function (next) {
    const newQuantity = this._update.quantity;
    // Find the product being updated
    const product = await this.model.findOne(this.getQuery());

    // Iterate through each user's cart
    await User.find().then(async (users) => {
        for (const user of users) {
            // Check if the product exists in the user's cart and has not been purchased
            const item = user.cart.find(
                (item) =>
                    item.product.toString() === product._id.toString() &&
                    !item.purchaseDate
            );
            if (item && item.quantity > newQuantity) {
                // If the quantity in the user's cart exceeds the new quantity of the product, update
it
                item.quantity = newQuantity;
                await user.save({ validateBeforeSave: false });
            }
        }
    });

    next();
});

//

//

productSchema.pre(/delete/i, async function (next) {
    const productId = this.getQuery()._id;
```

```
    try {
        // Find the product
        const product = await this.model.findOne({ _id: productId });

        // Delete old images if they exist
        if (product && product.images && product.images.length > 0) {
            await Promise.all(
                product.images.map(async (image) => {
                    const imagePath = path.join(
                        'public',
                        'img',
                        'products',
                        image
                    );
                    try {
                        await fs.access(imagePath, fs.constants.F_OK); // Check if file exists
                        await fs.unlink(imagePath); // Delete file
                    } catch (error) {
                        if (error.code === 'ENOENT') {
                        } else {
                            throw error; // Propagate other errors
                        }
                    }
                })
            );
        }

        // Find all users who have this product in their cart
        const users = await User.find({ 'cart.product': productId });

        // Remove this product from each user's cart
        await Promise.all(
            users.map(async (user) => {
                user.cart = user.cart.filter(
                    (item) => item.product.toString() !== productId.toString()
                );
                await user.save({ validateBeforeSave: false });
            })
        );

        next();
    } catch (error) {
        next(error);
    }
});

//

//
```

```
const Product = mongoose.model('Product', productSchema);
module.exports = Product;
```

models/userModel.js

```
const crypto = require('crypto');
const mongoose = require('mongoose');
const validator = require('validator');
const bcrypt = require('bcryptjs');

const userSchema = new mongoose.Schema({
    name: {
        type: String,
        required: [true, 'Please tell us your name!'],
    },
    email: {
        type: String,
        required: [true, 'Please provide your email'],
        unique: true,
        lowercase: true,
        validate: [validator.isEmail, 'Please provide a valid email'],
    },
    photo: {
        type: String,
        default: 'default.jpg',
    },
    role: {
        type: String,
        enum: ['user', 'admin'],
        default: 'user',
    },

    cart: [
        {
            product: {
                type: mongoose.Schema.Types.ObjectId,
                required: [true, 'Product reference error'],
                ref: 'Product',
            },
            quantity: {
                type: Number,
                required: [true, 'Please input product quantity'],
                min: 0,
            },
            purchaseDate: {
                type: Date,
                default: null,
            },
```

```
        address: {
            fullName: {
                type: String,
                required: [true, 'Please input full name for delivery'],
            },
            addressLine1: {
                type: String,
                required: [
                    true,
                    'Please input address line 1 for delivery',
                ],
            },
            addressLine2: {
                type: String,
            },
            city: {
                type: String,
                required: [true, 'Please input city for delivery'],
            },
            state: {
                type: String,
                required: [true, 'Please input state for delivery'],
            },
            postalCode: {
                type: String,
                required: [true, 'Please input postal code for delivery'],
            },
            country: {
                type: String,
                required: [true, 'Please input country for delivery'],
            },
        },
    },
],

password: {
    type: String,
    required: [true, 'Please provide a password'],
    minlength: 3,
    select: false,
},
passwordConfirm: {
    type: String,
    required: [true, 'Please confirm your password'],
    validate: {
        // This only works on CREATE and SAVE
        validator: function (el) {
            return el === this.password;
        },
        message: 'Passwords are not the same!',
    },
```

```
    },
    passwordChangedAt: Date,
    passwordResetToken: String,
    passwordResetExpires: Date,
    active: {
        type: Boolean,
        default: true,
        select: false,
    },
});

//

//

// PASSWORD

userSchema.pre('save', async function (next) {
    // Only run this function if password was actually modified
    if (!this.isModified('password')) return next();

    this.password = await bcrypt.hash(this.password, 12);

    this.passwordConfirm = undefined;
    next();
});

userSchema.pre('save', function (next) {
    if (!this.isModified('password') || this.isNew) return next();

    this.passwordChangedAt = Date.now() - 1000;
    next();
});

userSchema.methods.correctPassword = async function (
    candidatePassword,
    userPassword
) {
    return await bcrypt.compare(candidatePassword, userPassword);
};

userSchema.methods.changedPasswordAfter = function (JWTTimestamp) {
    if (this.passwordChangedAt) {
        const changedTimestamp = parseInt(
            this.passwordChangedAt.getTime() / 1000,
            10
        );

        return JWTTimestamp < changedTimestamp;
    }
```

```
    // False means NOT changed
    return false;
};

userSchema.methods.createPasswordResetToken = function () {
    const resetToken = crypto.randomBytes(32).toString('hex');

    this.passwordResetToken = crypto
        .createHash('sha256')
        .update(resetToken)
        .digest('hex');

    this.passwordResetExpires = Date.now() + 10 * 60 * 1000;

    return resetToken;
};

//

//

// Call mongoose.model() after methods and hooks declaration
const User = mongoose.model('User', userSchema);
module.exports = User;
```

4 ROUTE HANDLERS

routes/productRoutes.js

```
const express = require('express');
const productController = require('../controllers/productController');
const authController = require('../controllers/authController');

const router = express.Router();

//

//

router.route('/categories').get(productController.getAllUniqueCategories);

router.route('/').get(productController.getAllProducts);
router.route('/:id').get(productController.getProduct);

router.route('/:id/reviews').get(productController.getReviews);

//

//
```

```
router.use(authController.protect);
router.route('/:id/review').post(productController.createReview); // create review
router.route('/:id/review/:review_id').delete(productController.deleteReview);     //     create
review

//

//

router.use(authController.restrictTo('admin'));

router.route('/').post(productController.createProduct);
router.route('/').delete(productController.deleteAllProducts);
router
    .route('/:id')
    .patch(
        productController.uploadProductImages,
        productController.resizeProductImages,
        productController.updateProduct
    )
    .delete(productController.deleteProduct);

module.exports = router;
```

routes/transactionRoutes.js

```
const express = require('express');
const transactionController = require('./../controllers/transactionController');
const authController = require('../controllers/authController');

const router = express.Router();

//

//

router.use(authController.protect);

router.get(
    '/checkout-session/:userId',
    transactionController.getCheckoutSession
);

// route '/webhook-checkout' in app.js because it's need to be in row

module.exports = router;
```

routes/userRoutes.js

```
const express = require('express');
const authController = require('./../controllers/authController');
const userController = require('./../controllers/userController');
const cartController = require('./../controllers/cartController');

const router = express.Router();

//

//

router.post('/signup', authController.signup);
router.post('/login', authController.login);
router.get('/logout', authController.logout);

router.post('/forgotPassword', authController.forgotPassword);
router.patch('/resetPassword/:token', authController.resetPassword);

//

//

router.route('/cart/:user_id').get(cartController.getAllItems);

//

//

// Protect all routes after this middleware
router.use(authController.protect);

router.get('/me', userController.getMe, userController.getUser);
router.patch(
    '/updateMe',
    userController.uploadUserPhoto,
    userController.resizeUserPhoto,
    userController.updateMe
);
router.delete('/deleteAvatar', userController.deleteAvatar);
router.delete('/deleteMe', userController.deleteMe);

//

//

// Routes for all items in user cart
router
    .route('/cart')
```

```
    .post(cartController.createItem)
    .delete(cartController.deleteAllItems);

// Routes for one item by ID in user cart
router
    .route('/cart/:item_id')
    .get(cartController.getItem)
    .patch(cartController.updateItem)
    .delete(cartController.deleteItem);

//

//

// Restrict all routes after this middleware
router.use(authController.restrictTo('admin'));

router
    .route('/')
    .get(userController.getAllUsers)
    .post(userController.createUser)
    .delete(userController.deleteAllUsers);

router
    .route('/:id')
    .get(userController.getUser)
    .patch(userController.updateUser)
    .delete(userController.deleteUser);

module.exports = router;
```

routes/viewRoutes.js

```
const express = require('express');
const authController = require('../controllers/authController');
const viewsContoller = require('../controllers/viewsController');
const transactionController = require('../controllers/transactionController');

const router = express.Router();

router.get('/', authController.isLoggedIn, viewsContoller.getHome);

router.get('/shop', authController.isLoggedIn, viewsContoller.getShop);

router.get(
    '/prod-single/:slug',
    authController.isLoggedIn,
    viewsContoller.getSProduct
);
```

```
router.get('/blog', authController.isLoggedIn, viewsContoller.getBlog);

router.get('/about', authController.isLoggedIn, viewsContoller.getAbout);

router.get('/contact', authController.isLoggedIn, viewsContoller.getContact);

router.get(
    '/cart',
    authController.isLoggedIn,
    authController.protect,
    viewsContoller.getCart
);

router.get(
    '/admin',
    authController.isLoggedIn,
    authController.protect,
    authController.restrictTo('admin'),
    viewsContoller.getAdmin
);

module.exports = router;
```

5 UTILITIES

utils/apiFeatures.js

```
class APIFeatures {
    constructor(query, queryString) {
        this.query = query;
        this.queryString = queryString;
    }

    filter() {
        const queryObj = { ...this.queryString };
        const excludedFields = [
            'page',
            'sort',
            'limit',
            'fields',

            'category',
            'name',
        ];
        excludedFields.forEach((el) => delete queryObj[el]);

        // Add category filtering
        if (this.queryString.category) {
```

```
        const categories = this.queryString.category
            .split(',')
            .map((category) => category.trim());
        queryObj.category = { $in: categories };
    }

    // Add name filtering
    if (this.queryString.name) {
        queryObj.name = this.queryString.name;
    }

    // Advanced filtering
    let queryStr = JSON.stringify(queryObj);
    queryStr = queryStr.replace(
        /\b(gte|gt|lte|lt)\b/g,
        (match) => `$${match}`
    );

    this.query = this.query.find(JSON.parse(queryStr));

    return this;
}

sort() {
    if (this.queryString.sort) {
        const sortBy = this.queryString.sort.split(',').join(' ');
        this.query = this.query.sort(sortBy);
    } else {
        this.query = this.query.sort('-createdAt');
    }

    return this;
}

limitFields() {
    if (this.queryString.fields) {
        const fields = this.queryString.fields.split(',').join(' ');
        this.query = this.query.select(fields);
    } else {
        this.query = this.query.select('-__v');
    }

    return this;
}

paginate() {
    const page = this.queryString.page * 1 || 1;
    const limit = this.queryString.limit * 1 || 99999;
    const skip = (page - 1) * limit;

    this.query = this.query.skip(skip).limit(limit);
```

```
        return this;
    }
}
module.exports = APIFeatures;
```

utils/appError.js

```
class AppError extends Error {
    constructor(message, statusCode) {
        super(message);

        this.statusCode = statusCode;
        this.status = `${statusCode}`.startsWith('4') ? 'fail' : 'error';
        this.isOperational = true;

        this.stack = Error.captureStackTrace(this, this.constructor);
    }
}

module.exports = AppError;
```

utils/catchAsync.js

```
module.exports = (fn) => {
    return (req, res, next) => {
        fn(req, res, next).catch((err) => next(err));
    };
};
```

utils/email.js

```
const nodemailer = require('nodemailer');
require('dotenv').config(); // Load environment variables from .env file

const sendEmail = async (options, res) => {
    // create a transporter
    const transporter = nodemailer.createTransport({
        service: 'gmail',
        auth: {
            user: process.env.EMAIL_FROM,
            pass: process.env.GMAIL_PASSWORD,
        },
    });
```

```
    // Define the email options
    const mailOptions = {
        from: `VB SITE <${process.env.EMAIL_FROM}>`,
        to: options.email,
        subject: options.subject,
        text: options.message,
    };

    // actually send the email
    const info = await transporter.sendMail(mailOptions);
};

module.exports = sendEmail;
```

6 STYLESHEETS (CSS)

public/css/style.css

```
@import
url('https://fonts.googleapis.com/css2?family=Spartan:wght@100;200;300;400;500;600;700;800
;900&display=swap');
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: 'Spartan', sans-serif;
}

:root {
    --primary-color: #088178;
    --secondary-color: #465b52;
    --accent-color: #cce7d0;
    --white: #ffffff;
    --black: #000000;

    --font-size-h1: 46px;
    --font-size-h2: 36px;
    --font-size-h3: 28px;
    --font-size-h4: 22px;
    --font-size-h5: 18px;
    --font-size-large: 16px;
    --font-size-normal: 14px;
    --font-size-small: 12px;
    --font-size-caption: 10px;
}

html {
    scroll-behavior: smooth;
}
```

```
/* Global Styles */

h1 {
    font-size: var(--font-size-h1);
    line-height: 64px;
    color: #222;
}

h2 {
    font-size: var(--font-size-h2);
    line-height: 54px;
    color: #222;
}

h4 {
    font-size: var(--font-size-h4);
    color: #222;
}

h6 {
    font-weight: 700;
    font-size: var(--font-size-normal);
}

p {
    font-size: var(--font-size-normal);
    color: var(--secondary-color);
    margin: 15px 0 20px 0;
}

.section-p1 {
    padding: 40px 80px;

    max-width: 1280px;
    margin: auto;
}

.section-m1 {
    margin: 40px 0;

    /* max-width: 1280px;
  margin: auto; */
}

body {
    width: 100%;
}

button.normal {
    font-size: var(--font-size-normal);
```

```
    font-weight: 600;
    padding: 15px 30px;
    color: var(--black);
    background-color: white;
    cursor: pointer;
    border-radius: 4px;
    border: none;
    outline: none;
    transition: 0.2s;
}
button.white {
    font-size: var(--font-size-normal);
    font-weight: 600;
    padding: 11px 18px;
    color: rgb(255, 255, 255);
    background-color: transparent;
    cursor: pointer;
    border: 1px solid white;
    outline: none;
    transition: 0.2s;
}

/* PAGE HEADER */
#page-header h2 {
    font-weight: 700;
}

#page-header p {
    margin: 10px auto;
    font-size: var(--font-size-normal);
    max-width: 900px;
}

/* HEADER */
#header {
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 5px;
    background-color: var(--white);
    box-shadow: 0 5px 15px rgb(0, 0, 0, 0.2);

    z-index: 999;
    position: sticky;
    top: 0;
    left: 0;
}

#header img.logo {
    height: 30px;
    width: auto;
```

```
}

/* NAVBAR */

#navbar {
    display: flex;
    align-items: center;
    justify-content: center;
    margin-top: 20px;
}

#navbar.active {
    justify-content: start;
    right: 0px;
}
#navbar li {
    list-style: none;
    padding: 0 20px;

    position: relative;
}
#navbar li a {
    text-decoration: none;
    font-size: var(--font-size-normal);
    font-weight: 600;
    color: #1a1a1a;
    cursor: pointer;

    transition: 0.3s ease;
}

#navbar li a:hover,
#navbar li a.active {
    color: var(--primary-color);
}
#navbar li a.active::after,
#navbar li a:hover::after {
    content: '';
    width: 30%;
    height: 2px;
    background-color: var(--primary-color);

    position: absolute;
    bottom: -4px;
    left: 20px;
}
#navbar i {
    color: #222;
    font-size: var(--font-size-h5);
}
```

```
#mobile {
    display: none;
    align-items: center;
}
#mobile ul {
    list-style: none;
    display: flex;
    margin: 0;
    padding: 0;
}
#mobile li {
    margin-right: 30px; /* Adjust as needed for spacing between icons */
}
#mobile a {
    cursor: pointer;
    font-size: var(--font-size-h5);
    text-decoration: none;
    color: #222; /* Adjust as needed for icon color */
}

#close {
    display: none;
}
#bar a {
    cursor: pointer;
}

/* HERO */

#hero {
    background-image: url('../img/header7.jpg');
    min-height: 70vh;
    max-height: fit-content;
    width: 100%;
    background-size: cover;
    background-position: center; /* horizontally and vertically centered */

    display: flex;
    flex-direction: column;
    align-items: start;
    justify-content: center;
    text-align: start;
}
#hero .hero-container {
    padding: 80px 40px;
    margin: auto;
    max-width: 1280px;
}

#hero .hero-container h3 {
    line-height: normal;
```

```
}
#hero .hero-container h1 {
    font-weight: 700;
    line-height: normal;
    color: var(--primary-color);
}
#hero .hero-container p {
    color: black;
    max-width: 80%;
}
#hero .hero-container a {
    color: var(--primary-color);
    border: 0;
    padding: 15px 30px;
    border-radius: 4px;
    background-color: white;
    background-repeat: no-repeat;
    cursor: pointer;
    font-weight: 700;
    font-size: var(--font-size-normal);
    text-decoration: none;
    transition: color 0.3s ease, background-color 0.3s ease;
}

#hero .hero-container a:hover {
    color: white;
    background-color: var(--primary-color);
}

/* FEARURE */
#feature {
    display: flex;
    align-items: center;
    justify-content: space-between;
    flex-wrap: wrap;
}
#feature .fe-box {
    width: 180px;
    text-align: center;
    padding: 25px 15px;
    box-shadow: 20px 20px 34px rgb(0, 0, 0, 0.1);

    border: 1px solid var(--accent-colorr);
    border-radius: 4px;
    margin: 15px 0;

    transition: 0.2s ease;
}
#feature .fe-box:hover {
    box-shadow: 10px 10px 54px rgb(0, 0, 0, 0.2);
}
```

```
#feature .fe-box img {
    width: 100%;
    margin-bottom: 10px;
}
#feature .fe-box h6 {
    display: inline-block;

    padding: 9px 8px 6px 8px;
    line-height: 1;
    border-radius: 4px;
    color: var(--primary-color);
    background-color: #fddde4;
}

#feature .fe-box:nth-child(2) h6 {
    background-color: #cdebbc;
}
#feature .fe-box:nth-child(3) h6 {
    background-color: #d1e8f2;
}
#feature .fe-box:nth-child(4) h6 {
    background-color: #cdd4f8;
}
#feature .fe-box:nth-child(5) h6 {
    background-color: #f6dbf6;
}
#feature .fe-box:nth-child(6) h6 {
    background-color: #fff2e5;
}

/* PRODUCT */
#product1 {
    padding: 20px 0;
    text-align: center;
}
#product1 .pro-container {
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
}
#product1 .pro-container .no-products {
    display: flex;
    flex-direction: column;
    justify-content: center;
    text-align: center;
    margin: 60px 0;
}
#product1 .pro-container a {
    text-decoration: none;
}
#product1 .pro-header {
```

```
    display: flex;
    flex-direction: column;
    align-self: center;
}
#product1 .pro-header.hiden {
    display: none;
}

/* PRODUCT IN SHOP LIST */

#product1 .pro {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    width: 220px;
    padding: 10px 12px;
    border: 1px solid var(--accent-colorr); /* Corrected typo in variable name */
    border-radius: 25px;
    cursor: pointer;
    box-shadow: 0 20px 30px rgba(0, 0, 0, 0.05);
    margin: 15px 10px;
    transition: 0.2s ease;
    position: relative;
}

#product1 .pro:hover {
    box-shadow: 0 20px 30px rgba(0, 0, 0, 0.1);
}

#product1 .pro img {
    width: 100%;
    margin-bottom: 10px;
    border-radius: 4px;
}

#product1 .pro .des {
    display: flex;
    flex-direction: column;
    justify-content: space-between;
    align-items: flex-start;
    text-align: start;
    height: 100%; /* Fill the available space */
    box-sizing: border-box;
}

#product1 .pro .des span {
    color: #606063;
    font-size: var(--font-size-normal);
}

#product1 .pro .des h5 {
```

```
    padding-top: 7px; /* Corrected typo in padding */
    color: #1a1a1a;
    font-size: var(--font-size-normal);
}

#product1 .pro .des .stars {
    color: var(--primary-color);
    font-size: var(--font-size-small);
    margin: 0;
}

#product1 .pro .des h4 {
    margin-top: auto;
    padding-top: 20px;
    font-size: var(--font-size-small);
    font-weight: 700;
    color: var(--primary-color);
}

#product1 .pro .cart {
    display: flex;
    flex-direction: column;
    align-items: center;

    width: 40px;
    height: 40px;
    line-height: 40px;
    border-radius: 50px;
    background-color: #e8f6ea;
    /* font-weight: 500; */
    color: var(--primary-color);
    border: 1px solid var(--accent-colorr);

    position: absolute;
    bottom: 20px;
    right: 20px;
}

/* BANNER */
#banner {
    max-width: 100%;
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: center;
    text-align: center;

    background-image: url('../img/lg-banner3.jpg');
    background-color: #606063;
    width: 100%;
    background-size: cover;
```

```
    min-height: 260px;
}
#banner h4 {
    color: white;
    font-size: var(--font-size-large);
}
#banner h2 {
    color: white;
    font-size: var(--font-size-h3);
    padding: 10px 0;
}
#banner h2 span {
    color: red;
}
#banner button:hover {
    color: white;
    background-color: var(--primary-color);
}

/* SM BANNER   */
#sm-banner {
    width: auto;
    /* display: flex; */
    justify-content: space-between;
    flex-wrap: wrap;
}
#sm-banner .banner-box {
    display: flex;
    flex-direction: column;
    justify-content: center;
    align-items: flex-start;

    background-image: url('../img/sm-banner1.jpg');
    background-position: center;
    background-size: cover;

    padding: 15px 20px;
    margin-bottom: 10px;
}
#sm-banner .banner-box2 {
    background-image: url('../img/sm-banner2.jpg');
    background-position: center;
}
#sm-banner .banner-box2,
#sm-banner h4 {
    color: white;
    font-size: var(--font-size-h5);
    font-weight: 300;
}
#sm-banner h2 {
```

```
    color: white;
    font-size: var(--font-size-h5);
    font-weight: 800;
}
#sm-banner span {
    color: white;
    font-size: var(--font-size-normal);
    font-weight: 500;
    padding-bottom: 15px;
}
#sm-banner .banner-box:hover button {
    background-color: var(--primary-color);
    border: 1px solid var(--primary-color);
}

/* NEWSLETTER */
#newsletter {
    max-width: 100%;
    margin: 0;

    background-image: url('../img/newsletter.png');
    background-repeat: no-repeat;
    background-position: 20% 30%;
    background-color: #041e42;
}
#newsletter .newslatter-container {
    margin: auto;
    max-width: 1120px;
    display: flex;
    justify-content: space-between;
    flex-wrap: wrap;
    align-items: center;
}

#newsletter h4 {
    font-size: var(--font-size-h4);
    font-weight: 700;
    color: white;
}
#newsletter p {
    font-size: var(--font-size-small);
    font-weight: 600;
    color: #818ea0;
}
#newsletter p span {
    color: #ffbd42;
}
#newsletter .form {
    display: flex;
    width: 40%;
}
```

```
#newsletter input {
    height: 3.125rem;
    padding: 0 1.25em;
    font-size: var(--font-size-normal);
    width: 100%;
    border: 1px solid transparent;
    border-radius: 4px;
    outline: none;

    border-top-right-radius: 0;
    border-bottom-right-radius: 0;
}
#newsletter button {
    background-color: var(--primary-color);
    color: white;
    white-space: nowrap;

    border-top-left-radius: 0;
    border-bottom-left-radius: 0;
}

/* FOOTER */

#footer .columns .col {
    width: 33.33%;
    float: left;
    margin-bottom: 20px;
}

#footer .columns .col h4,
#footer .columns .col .follow h4 {
    margin: 0;
}

#footer .columns .col p {
    color: black;
    font-size: var(--font-size-normal);
    margin: 4px 0;
}
#footer .columns .col p strong {
    font-weight: bold;
}

#footer .columns .col a {
    display: block;
    margin-bottom: 5px;
    text-decoration: none;

    font-size: var(--font-size-normal);
    color: #000;
    transition: color 0.2s ease;
```

```
}

#footer .columns .col a:hover,
#footer .columns .col .follow .icon i:hover {
    color: var(--primary-color);
}

#footer .columns .col .follow {
    margin-top: 20px;
}

#footer .columns .col .follow .icon i {
    font-size: var(--font-size-normal);
    margin-right: 10px;
    cursor: pointer;

    color: #000;
    transition: color 0.2s ease;
}

#footer .copyright {
    clear: both;
    margin-top: 20px;
    text-align: center;
}

#footer .copyright p {
    margin: 0;
    padding-top: 20px;
}

/* SHOP PAGE */
#page-header {
    background-image: url('../img/header1.jpg');
    width: 100%;
    height: 240px;
    background-size: cover;

    display: flex;
    text-align: center;
    justify-content: center;
    flex-direction: column;
    padding: 14px;
}
#page-header h2,
#page-header p {
    color: white;
}

/* pagi */
#prod-pagi,
```

```
#blog-pagi {
    user-select: none;
    display: flex;
    justify-content: center;
    flex-wrap: wrap;
    align-items: center;

    padding: 20px 40px;
}
#prod-pagi p,
#blog-pagi p {
    cursor: pointer;
    padding: 6px 10px;
    margin: 0 4px;
    border-radius: 4px;
    color: #222;
    font-weight: 600;
}
#prod-pagi p.active,
#blog-pagi p.active {
    background-color: var(--primary-color);
    color: white;
}

#prod-pagi p i,
#blog-pagi p i {
    font-size: var(--font-size-normal);
    font-weight: 600;
}

/* SINGLE PRODUCT PAGE */
#prodetails {
    display: flex;
    margin-top: 20px;
}

#prodetails .single-pro-image {
    width: 40%;
    margin: 0 50px 40px 0;
}

#prodetails .single-pro-image img {
    border: solid 1px #c9cdce;
    border-radius: 4px;
}

#prodetails .small-img-groop {
    display: flex;
    justify-content: start;
    gap: 6px; /* Set the desired space between elements */
}
```

```
#prodetails .small-img-groop .small-img.active {
    border: solid 2px var(--primary-color);
}

#prodetails .small-img-col {
    margin-top: 10px;
    flex-basis: 24%;
    cursor: pointer;
}

/*  */

#prodetails .single-pro-details {
    width: 50%;
    padding-top: 10px;
}

#prodetails .single-pro-details .details {
    margin-top: 40px;
}
#prodetails .single-pro-details p {
    margin-bottom: 20px;
    font-size: var(--font-size-normal);
}
#prodetails .single-pro-details h5 {
    margin-bottom: 20px;
    font-size: var(--font-size-h5);
}
#prodetails .single-pro-details h4 {
    font-size: var(--font-size-h4);
}

#prodetails .single-pro-details select {
    display: block;
    padding: 5px 10px;
    margin-bottom: 10px;
}
#prodetails .single-pro-details select:focus {
    outline: none;
}
#prodetails .single-pro-details input {
    width: 80px;
    height: 47px;
    padding-left: 10px;
    font-size: var(--font-size-normal);
    margin-right: 10px;
}
#prodetails .single-pro-details input:focus {
    outline: none;
}
```

```
#prodetails .single-pro-details button {
    background-color: var(--primary-color);
    color: white;
}
#prodetails .single-pro-details span {
    font-size: var(--font-size-normal);
    line-height: 25px;
}

/* CREATE REVIEW  */
#reviews h5 {
    margin-bottom: 10px;
    font-size: var(--font-size-h5);
}
#reviews p {
    margin-bottom: 20px;

    font-size: var(--font-size-normal);
}
#reviewForm {
    display: flex;
    flex-direction: column;
    width: 100%;

    margin-bottom: 40px;
}
#reviewForm .rating {
    display: flex;
    flex-direction: row;
    text-align: center;

    margin-bottom: 10px;
}
#reviewForm .rating p {
    margin: 0 20px 0 0;
    font-weight: 500;
}
#reviewForm .rating i {
    font-size: 18px;
    color: var(--secondary-color);
}
#reviewForm .comment {
    font-size: var(--font-size-normal);
    width: 100%; /* Change width to 100% */
    padding: 10px 5px 10px 15px;
    box-sizing: border-box; /* Ensure padding is included in width calculation */
}
#reviewForm .comment:focus {
    outline: none;
}
#reviewForm button {
```

```
    width: auto; /* Change width to auto */
    align-self: flex-end; /* Align to the right within the flex container */
    margin-top: 8px;
    background: #088178;
    color: var(--white);
}

/* ALL REVIEWS */

#reviews-container {
    display: flex;
    flex-direction: column;
}

#reviews-container .review {
    display: flex;
    flex-direction: row;

    padding: 20px 40px;
    margin-bottom: 20px;

    background-color: #f6f6f6;
    border-radius: 3px;

    /* border: 1px solid #ddd; */
}

#reviews-container .review .user-info {
    display: flex;
    flex-direction: column;

    align-items: center;
    text-align: center;

    margin-right: 20px;
}

#reviews-container .review .user-info p {
    font-size: var(--font-size-normal);
    font-weight: 600;
    margin: 10px 0 0 0;
}

#reviews-container .review .user-info .user-photo {
    width: 70px;
    height: 70px;
    border-radius: 50%;
}

#reviews-container .review .review-content {
    display: flex;
```

```
    flex-direction: row;
    justify-content: space-between;

    gap: 20px;
    width: 100%;
}
#reviews-container .review .review-content p {
    margin: 10px 0;
}

#reviews-container .review .review-content .stars {
    font-size: var(--font-size-normal);
}

#reviews-container .review .review-content .comment {
    font-style: italic;
    font-size: var(--font-size-normal);
}
#reviews-container .review .review-content .specifics {
    display: flex;
    flex-direction: column;
    justify-content: flex-end;
    align-items: flex-end;
}
#reviews-container .review .review-content .specifics .options {
    display: flex;
    flex-direction: row;

    gap: 10px;
}
#reviews-container .review .review-content .specifics .options i {
    color: var(--secondary-color);
    cursor: pointer;
}

#reviews-container .review .review-content .date {
    font-size: var(--font-size-small);
}

#reviews-container p#no-reviews-message {
    font-size: var(--font-size-normal);
    text-align: start;
}

/* BLOG HEADER */
#page-header.blog-header {
    background-image: url('../img/header2.jpg');
}

#blog {
    padding: 150px 150px 0 150px;
```

```
}
#blog .blog-box {
    display: flex;
    align-items: center;
    width: 100%;
    padding-bottom: 100px;

    position: relative;
}
#blog .blog-img {
    width: 50%;
    margin-right: 40px;
}
#blog img {
    width: 100%;
    height: 300px;
    object-fit: cover;
}
#blog .blog-details {
    width: 50%;
}
#blog .blog-details a {
    text-decoration: none;
    font-size: var(--font-size-normal);
    color: var(--black);
    font-weight: 700;

    position: relative;
    transition: 0.2s;
}
#blog .blog-details a::after {
    content: '';
    width: 50px;
    height: 1px;
    background-color: var(--black);
    position: absolute;
    top: 4px;
    right: -60px;
}
#blog .blog-details a:hover {
    color: var(--primary-color);
}
#blog .blog-details a:hover::after {
    background-color: var(--primary-color);
}
#blog .blog-details h1 {
    position: absolute;
    top: -40px;
    left: 0;
    font-size: var(--font-size-h1);
    font-weight: 700;
```

```
    color: #c9cdce;
    z-index: -1;
}

/* ABOUT PAGE */
#page-header.about-header {
    background-image: url('../img/header5.jpg');
}
#about-head {
    display: flex;
    align-items: center;
}

#about-head img {
    width: 50%;
    height: 100%;
    object-fit: cover;
}

#about-head div {
    padding-left: 40px;
}

/* ABOUT APP */
#about-app {
    text-align: center;
}
#about-app .video {
    width: 100%;
    height: 0; /* Set initial height to 0 */
    padding-bottom: 75%;
    position: relative;
}

#about-app .video iframe {
    position: absolute;
    top: 0;
    left: 0;
    width: 100%;
    height: 100%; /* Fill the entire container */
    border-radius: 20px;
}

/* CONTACT PAGE */
#page-header.contact-header {
    background-image: url('../img/header4.jpg');
}

#contact-details {
    display: flex;
    align-items: center;
```

```
    justify-content: space-between;
}
#contact-details .details {
    width: 40%;
}
#contact-details .details span,
#form-details form {
    font-size: var(--font-size-normal);
}
#contact-details .details h2 {
    font-size: var(--font-size-h4);
    line-height: 36px;
    padding: 20px 0;
}
#contact-details .details h3 {
    font-size: var(--font-size-normal);
    padding-bottom: 15px;
}
#contact-details .details li {
    list-style: none;
    display: flex;
    padding: 10px 0;
}
#contact-details .details li i {
    font-size: var(--font-size-normal);
    padding-right: 22px;
}
#contact-details .details li p {
    margin: 0;
    font-size: var(--font-size-normal);
}
#contact-details .map {
    width: 55%;
    height: 400px;
}
#contact-details .map iframe {
    width: 100%;
    height: 100%;
}

#form-details {
    display: flex;
    margin-bottom: 40px;
    align-items: center;
    justify-content: space-between;
}
#form-details h2 {
    font-size: var(--font-size-h4);
}
#form-details form {
    width: 65%;
```

```
    margin-right: 20px;
    display: flex;
    flex-direction: column;
    align-items: flex-start;
}
#form-details form input,
#form-details form textarea {
    width: 100%;
    padding: 12px 15px;
    outline: none;
    margin-bottom: 20px;
    border: 1px solid #e1e1e1;
    border-radius: 4px;
}
#form-details form button {
    background-color: var(--primary-color);
    color: white;
}
#form-details .people div {
    padding-bottom: 25px;

    display: flex;
    align-items: flex-start;
}
#form-details .people div img {
    width: 65px;
    height: 65px;
    object-fit: cover;
    margin-right: 15px;
}
#form-details .people div p {
    margin: 0;
    font-size: var(--font-size-small);
    line-height: 25px;
}
#form-details .people div p span {
    display: block;
    font-size: var(--font-size-normal);
    font-weight: 600;
    color: var(--black);
}

/* SHOPING CART PAGE */
#page-header.cart-header {
    background-image: url('../img/header3.jpg');
}
#cart {
    overflow-x: auto;
    text-decoration: none;
}
#cart a {
```

```
    cursor: pointer;
    text-decoration: none;
    color: black;
    transition: color 0.2s ease;
}
#cart a:hover {
    color: var(--primary-color);
}

#cart table {
    width: 100%;
    border-collapse: collapse;
    table-layout: fixed;
    white-space: nowrap;
}
#cart table img {
    width: 70px;
}
#cart table td:nth-child(1) {
    width: 100px;
    text-align: center;
}
#cart table td:nth-child(2) {
    width: 150px;
    text-align: center;
}
#cart table td:nth-child(3) {
    width: 250px;
    text-align: center;
}
#cart table td:nth-child(4),
#cart table td:nth-child(5),
#cart table td:nth-child(6) {
    width: 150px;
    text-align: center;
}
#cart table td:nth-child(5) input {
    width: 70px;
    padding: 10px 5px 10px 15px;
}
#cart table td:nth-child(5) input:focus {
    outline: none;
}
#cart table thead {
    border: 1px solid #c9cdce;
    border-left: none;
    border-right: none;
}
#cart table thead td {
    font-weight: 700;
    text-transform: uppercase;
```

```
    font-size: var(--font-size-normal);
    padding: 18px 0;
}
#cart table tbody tr td {
    padding-top: 15px;
}
#cart table tbody td {
    font-size: var(--font-size-normal);
}
#cart i {
    color: #222;
}

#cart-add {
    display: flex;
    flex-wrap: wrap;
    justify-content: space-between;
}

/* CUPON AND SUBTOTAL TABLE */

#coupon {
    width: 50%;
    margin-bottom: 30px;
}
#coupon h3,
#subtotal h3 {
    padding-bottom: 15px;
}
#coupon input {
    padding: 10px 20px;
    outline: none;
    width: 60%;
    margin-right: 10px;
    border: 1px solid #e2e9e1;
}
#coupon button,
#subtotal button {
    background-color: var(--primary-color);
    color: white;
    padding: 12px 20px;
}

#subtotal {
    width: 50%;
    margin-bottom: 30px;
    border: 1px solid #e2e9e1;
    padding: 30px;
}
#subtotal table {
    border-collapse: collapse;
```

```
    width: 100%;
    margin-bottom: 20px;
}
#subtotal table td {
    width: 50%;
    border: 1px solid #e2e9e1;
    padding: 10px;
    font-size: var(--font-size-normal);
}

/* ADMIN PANEL PAGE */

#page-header.admin-header {
    background-image: url('../img/header6.jpg');
}

#admin-panel {
    max-width: 100%;
}

#admin-panel .table-container {
    overflow-x: auto;
}

#admin-panel table {
    width: 100%;
    border-collapse: collapse;
    table-layout: fixed;
    white-space: nowrap;
}
#admin-panel table img {
    width: 100%;
    padding: 10px;
}
#admin-panel table td {
    border-bottom: 1px solid #c9cdce;
}
#admin-panel table td:nth-child(1) {
    width: 75px;
    text-align: center;
}
/* IMAGE ROW */
#admin-panel table td:nth-child(2) {
    width: calc(125px + 60px);
    text-align: center;
    align-items: center;
}
#admin-panel table td:nth-child(2) .image-td {
    display: flex;
    flex-direction: row;
    align-items: center;
```

```
    background-color: #c9cdce;
    margin: 10px;
    border: 1px solid #c9cdce;
    border-radius: 5px;
}
#admin-panel table td:nth-child(2) .image-td .upload-btn {
    width: 100%;
    text-align: center;
}
#admin-panel table td:nth-child(2) .image-td .upload-btn i {
    font-size: 24px;
    color: white;
    cursor: pointer;
}

#admin-panel table td:nth-child(9),
#admin-panel table td:nth-child(10),
#admin-panel table td:nth-child(11) {
    width: 125px;
    text-align: center;
}
#admin-panel table td:nth-child(3),
#admin-panel table td:nth-child(9) {
    width: 225px;
    text-align: center;
}
#admin-panel table td:nth-child(4),
#admin-panel table td:nth-child(5),
#admin-panel table td:nth-child(6) {
    width: 125px;
    text-align: center;
}
#admin-panel table td:nth-child(7),
#admin-panel table td:nth-child(8) {
    width: 275px;
    text-align: center;
}

#admin-panel table td:nth-child(3) input,
#admin-panel table td:nth-child(9) input {
    width: 200px;
    padding: 10px 5px 10px 15px;
}
#admin-panel table td:nth-child(4) input,
#admin-panel table td:nth-child(5) input {
    width: 100px;
    padding: 10px 5px 10px 15px;
}
#admin-panel table td:nth-child(7) textarea,
#admin-panel table td:nth-child(8) textarea {
```

```
    width: 250px;
    padding: 10px 5px 10px 15px;
}

#admin-panel table td:nth-child(3) input:focus,
#admin-panel table td:nth-child(4) input:focus,
#admin-panel table td:nth-child(5) input:focus,
#admin-panel table td:nth-child(7) textarea:focus,
#admin-panel table td:nth-child(8) textarea:focus,
#admin-panel table td:nth-child(9) input:focus {
    outline: none;
}
#admin-panel table thead {
    border: 1px solid #e2e9e1;
    border-left: none;
    border-right: none;
}
#admin-panel table thead td {
    font-weight: 700;
    text-transform: uppercase;
    font-size: var(--font-size-normal);
    padding: 18px 0;
}
#admin-panel table tbody tr td {
    padding-top: 15px;
}
#admin-panel table tbody td {
    font-size: var(--font-size-small);
}
#admin-panel i {
    font-size: var(--font-size-large);
    color: #222;
}

#createProdBtn {
    border: 1px solid #e2e9e1;
    margin-bottom: 20px;
    width: 100%;
    background-color: var(--primary-white);
    color: var(--black);
    padding: 12px 20px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.1);
    transition: box-shadow 0.1s ease;
}

#createProdBtn:hover {
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.2);
}

/* CART PURCHASE HISTORY */
```

```
#purchase-history p.date {
    text-align: center;
    margin: 40px 0 10px 0;
    font-size: var(--font-size-normal);
}

#purchase-history .table-container {
    overflow-x: auto;
}

#purchase-history table {
    width: 100%;
    border-collapse: collapse;
    border-radius: 3px;
    border-bottom: 1px solid #c9cdce;
    border-top: 1px solid #c9cdce;
}

#purchase-history table tr {
    height: auto;
}

#purchase-history table tr td {
    padding: 5px;
}
#purchase-history table tr td.img {
    min-width: 125px;
    width: 125px;
}
#purchase-history table tr td img {
    width: 100%;
}

#purchase-history .product-info {
    display: flex;
    flex-direction: column;
    min-width: 400px;
}

#purchase-history .product-info p {
    margin: 0;
}
#purchase-history .product-info h5.name {
    font-size: var(--font-size-normal);
}

#purchase-history .product-info p.time {
    margin-top: auto;
}

#purchase-history .product-info p.address {
```

```
    margin: 10px 0;
}

#purchase-history .time {
    display: flex;
    justify-content: space-between;
    align-items: center;
}

#purchase-history .price {
    padding-right: 20px;
    width: 25%;
    min-width: 220px;
    text-align: end;
}

#purchase-history .price h5 {
    font-size: var(--font-size-normal);
}
#purchase-history .price p {
    margin: 0;
}

/* ADMIN TABLE SLIDER IMAGES */

#admin-panel table .image-slider {
    position: relative;
    border-radius: 5px 0 0 5px;
    overflow: hidden;
    margin: 0 auto;
    max-width: 125px;
}

#admin-panel table .image-slider .slider-content {
    display: flex;
    transition: transform 0.5s ease;
}

#admin-panel table .image-slider .slider-image {
    flex: 0 0 auto;
    padding: 0;
    width: 100%;
    transition: opacity 0.5s ease;
}

#admin-panel table .image-slider .slider-btn {
    position: absolute;
    top: 50%;
    transform: translateY(-50%);
    width: 30px;
    height: 30px;
```

```
    border: none;
    cursor: pointer;
    display: flex;
    justify-content: center;
    align-items: center;
    font-size: 14px;
    color: #333;
    opacity: 0;
    transition: opacity 0.3s ease;

    background-color: rgba(255, 255, 255, 0.6);
    border-radius: 50%;
}

#admin-panel table .image-slider .slider-btn:hover {
    opacity: 1;
}

#admin-panel table .image-slider .prev {
    left: 4px;
}

#admin-panel table .image-slider .next {
    right: 4px;
}

/* MODAL AUTH */

#modal-auth.modal-overlay {
    background-color: rgba(0, 0, 0, 0.75);
    position: fixed;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    z-index: 1;
    display: none;
    opacity: 0; /* Start with the modal being transparent */
    transition: opacity 0.2s; /* Transition the opacity over 0.5 seconds */
}

#modal-auth.active.modal-overlay {
    display: flex;
    opacity: 1; /* Fully visible when active */
    justify-content: center;
    align-items: flex-start;
    padding-top: 80px;
    padding-bottom: 10px;
}

#modal-auth.modal-overlay .modal-container {
```

```
        margin: auto;
        min-width: 400px;
        width: 480px;
        max-width: 100%;
        max-height: calc(
            100vh - 80px
        ); /* Set maximum height to 50px less than viewport height to accommodate the header
*/
        overflow-y: auto;
        padding: 20px;
        z-index: 2;
        background-color: white;
        border: 1px solid #e1e1e1;
        border-radius: 5px;
        box-shadow: 0 5px 10px rgba(0, 0, 0, 0.137);
    }

    #modal-auth.modal-overlay .modal-container .modal-head {
        display: flex;
        justify-content: space-between;
        align-items: center;
        width: 100%;
        padding: 10px 16px;
        border-bottom: 1px solid #e1e1e1;
    }

    #modal-auth.modal-overlay .modal-container .modal-head h6 {
        margin: 0;
    }

    #modal-auth.modal-overlay .modal-container .modal-head i {
        cursor: pointer;
        font-size: var(--font-size-h5);
        color: #222;
        padding: 6px;
        border-radius: 50%;
        background-color: #e1e1e1;
        width: 24px;
        height: 24px;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    #modal-auth.modal-overlay form {
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        width: 100%;
        padding: 20px;
```

```
    margin-top: 20px;
}

/* UDATE USER PHOTO */
#modal-auth.modal-overlay form#form-update .avatar {
    display: flex;
    flex-direction: column;
    align-items: center;
    margin: 20px 0;
}

#modal-auth.modal-overlay form#form-update .avatar .menu {
    display: flex;
    flex-direction: column;
    text-align: center;
}
#modal-auth.modal-overlay form#form-update .avatar .menu p {
    cursor: pointer;
    margin: 0 auto;
}
#modal-auth.modal-overlay form#form-update .avatar .menu p i {
    margin-left: 10px;
}
#modal-auth.modal-overlay form#form-update .avatar img {
    margin-bottom: 20px;
    border-radius: 50%;
    width: 120px;
    height: 120px;
}

/*  */

#modal-auth.modal-overlay .group {
    width: 100%;
    display: flex;
    flex-direction: row;
    justify-content: space-evenly;
}

#modal-auth.modal-overlay form.hidden {
    display: none;
}

#modal-auth.modal-overlay form input {
    height: 100%;
    font-size: var(--font-size-normal);
    width: 100%;
    padding: 12px 15px;
    outline: none;
    margin-bottom: 20px;
    border: 1px solid #e1e1e1;
```

```
    border-radius: 4px;
}

#modal-auth.modal-overlay form button,
#modal-auth.modal-overlay #profile button {
    background-color: var(--primary-color);
    color: white;
}

#modal-auth.modal-overlay form h5 {
    text-align: center;
}

#modal-auth.modal-overlay form p {
    text-align: center;
}

#modal-auth.modal-overlay form p a {
    color: var(--primary-color);
    font-weight: 700;
    cursor: pointer;
    text-decoration: none;
}

/* MODAL ADDRESS */

#modal-address.modal-overlay {
    background-color: rgba(0, 0, 0, 0.75);
    position: fixed;
    top: 0;
    left: 0;
    height: 100%;
    width: 100%;
    z-index: 1;
    display: none;
    opacity: 0; /* Start with the modal being transparent */
    transition: opacity 0.2s; /* Transition the opacity over 0.5 seconds */
}

#modal-address.active.modal-overlay {
    display: flex;
    opacity: 1; /* Fully visible when active */
    justify-content: center;
    align-items: flex-start;
    padding-top: 80px;
    padding-bottom: 10px;
}

#modal-address.modal-overlay .modal-container {
    margin: auto;
    min-width: 400px;
```

```
        width: 480px;
        max-width: 100%;
        max-height: calc(
            100vh - 80px
        ); /* Set maximum height to 50px less than viewport height to accommodate the header
*/
        overflow-y: auto;
        padding: 20px;
        z-index: 2;
        background-color: white;
        border: 1px solid #e1e1e1;
        border-radius: 5px;
        box-shadow: 0 5px 10px rgba(0, 0, 0, 0.137);
    }

    #modal-address.modal-overlay .modal-container .modal-head {
        display: flex;
        justify-content: space-between;
        align-items: center;
        width: 100%;
        padding: 10px 16px;
        border-bottom: 1px solid #e1e1e1;
    }

    #modal-address.modal-overlay .modal-container .modal-head h6 {
        margin: 0;
    }

    #modal-address.modal-overlay .modal-container .modal-head i {
        cursor: pointer;
        font-size: var(--font-size-h5);
        color: #222;
        padding: 6px;
        border-radius: 50%;
        background-color: #e1e1e1;
        width: 24px;
        height: 24px;
        display: flex;
        justify-content: center;
        align-items: center;
    }

    #modal-address.modal-overlay form {
        display: flex;
        flex-direction: column;
        align-items: center;
        justify-content: center;
        width: 100%;
        padding: 20px;
        margin-top: 20px;
    }
```

```
#modal-address.modal-overlay form .address-fields {
    width: 100%;
}

#modal-address.modal-overlay form input {
    height: 100%;
    font-size: var(--font-size-normal);
    width: 100%;
    padding: 12px 15px;
    outline: none;
    margin-bottom: 20px;
    border: 1px solid #e1e1e1;
    border-radius: 4px;
}

#modal-address.modal-overlay form button {
    background-color: var(--primary-color);
    color: white;
    padding: 10px 20px;
    margin: 10px 0;
    border: none;
    border-radius: 4px;
    cursor: pointer;
    font-size: var(--font-size-normal);
}

#modal-address.modal-overlay form h5 {
    text-align: center;
}

#modal-address.modal-overlay form p {
    text-align: center;
}

/* ALERT */

.alert {
    position: fixed;
    top: 80px;
    left: 50%;
    -webkit-transform: translateX(-50%);
    transform: translateX(-50%);
    z-index: 999;

    color: white;
    font-size: var(--font-size-normal);
    text-align: center;

    background-color: var(--primary-color);
    border-bottom-right-radius: 5px;
```

```
    border-bottom-left-radius: 5px;
    padding: 10px 40px;
    box-shadow: 0 2px 4px rgba(0, 0, 0, 0.2);

    opacity: 1;
    transition: opacity 1s ease-in-out; /* Added transition property */
}
.alert i {
    font-size: var(--font-size-normal);
    color: white;
    margin-right: 10px;
}

/* ERROR */

#error {
    display: flex;
    flex-direction: column;
    justify-content: center;
    min-height: 50vh;
}

/*




*/

#prod-filter {
    display: flex;
    flex-direction: column;
    align-items: center;
    width: 100%;
}
#prod-filter .search {
    display: flex;
    width: 100%;
    max-width: 500px;
    justify-content: center;

    margin-bottom: 20px;
}
#prod-filter .text-field {
    display: flex;
    align-items: center;
    width: 100%;
    border: 1px solid #e1e1e1;
    border-radius: 8px;
```

```
    border-top-right-radius: 0;
    border-bottom-right-radius: 0;
}
#prod-filter .text-field div {
    display: flex;
    align-items: center;
    padding: 20px 20px;
}
#prod-filter .text-field input {
    height: 16px;
    padding: 0 20px;
    font-size: var(--font-size-normal);
    width: 100%;
    height: 100%;
    outline: none;
    border: none;

    border-radius: 8px;

    border-top-right-radius: 0;
    border-bottom-right-radius: 0;
}
#prod-filter .search .text-field i {
    color: #222;
    font-size: var(--font-size-h5);
    cursor: pointer;
}
#prod-filter .search button {
    min-width: fit-content;
    background-color: var(--primary-color);
    color: white;
    white-space: nowrap;
}
#prod-filter h5 {
    margin-bottom: 1rem;
}
#prod-filter button {
    border-radius: 8px;
    border-top-left-radius: 0;
    border-bottom-left-radius: 0;
}

#prod-filter .categories {
    display: flex;
    justify-content: center;
    align-items: center;
    width: 100%;
    flex-wrap: wrap;
    margin-bottom: 30px;
}
#prod-filter .cat-btn {
```

```
    color: rgb(207, 207, 207);
    background-color: white;
    font-size: var(--font-size-normal);
    font-weight: 600;

    border-radius: 4px;
    border: none;

    transition: 0.3s ease;
    margin: 6px 6px;
    border: 1px solid transparent;
}
#prod-filter .cat-btn:hover {
    box-shadow: 0px 8px 32px rgb(0, 0, 0, 0.1);
}
#prod-filter .cat-btn.active {
    border: 1px solid #606063;
    color: var(--black);
}
#prod-filter .cat-btn.active:hover {
    box-shadow: none;
}

/* product filter sorting order */

#prod-filter .sorting {
    display: flex;
    justify-content: center;
    margin: auto;
    position: relative; /* Added position relative to the container */
}

#prod-filter .sorting .name,
#prod-filter .sorting .price,
#prod-filter .sorting .date {
    cursor: pointer;
    display: flex;
    margin: 0 24px;
    align-items: center;
    font-size: var(--font-size-normal);
    font-weight: 600;
    user-select: none;
    position: relative; /* Added position relative to the sorting element */
}

#prod-filter .sorting i {
    margin-left: 10px;
    position: absolute; /* Changed to absolute positioning */
    top: 50%; /* Positioned in the middle vertically */
    transform: translateY(-50%); /* Centered vertically */
    right: -20px;
```

```
}

#prod-filter .sorting .hidden {
    display: none;
}

/*




*/

/* MEDIA */
@media (max-width: 799px) {
    .section-p1 {
        padding: 40px 40px;
    }

    #header {
        padding: 10px 10px;
    }

    #navbar {
        display: flex;
        flex-direction: column;
        align-items: flex-start;
        justify-content: center;
        position: fixed;
        top: -15px;
        right: -300px;
        height: 100vh;
        width: 300px;

        background-color: white;
        box-shadow: 0 40px 60px rgb(0, 0, 0, 0.1);
        padding: 80px 0 0 10px;

        transition: 0.3s ease-out;
    }

    #navbar li {
        margin-bottom: 25px;
    }

    #mobile {
        display: flex;
    }
```

```
#close {
    display: initial;
    position: absolute;
    top: 30px;
    left: 30px;
    color: #222;
    font-size: var(--font-size-h4);

    cursor: pointer;
}

#feature {
    justify-content: center;
}

#feature .fe-box {
    margin: 15px 15px;
}

#newsletter .form {
    width: 70%;
}

/* SINGLE PRODUCT PAGE */
#prodetails {
    display: flex;
    flex-direction: column;
}
#prodetails .single-pro-image {
    width: 100%;
    margin-right: 0px;
}
#prodetails .single-pro-details {
    width: 100%;
}

/* BLOG PAGE */
#blog {
    padding: 100px 20px 0 20px;
}
#blog .blog-box {
    display: flex;
    flex-direction: column;
    align-items: flex-start;
}
#blog .blog-img {
    width: 100%;
    margin-right: 0px;
    margin-bottom: 30px;
}
#blog .blog-details {
```

```
    width: 100%;
}

/* ABOUT PAGE */

#about-head {
    display: flex;
    flex-direction: column;
    align-items: center;
}
#about-head img.image {
    width: 100%;
    margin-bottom: 20px;
}
#about-head div {
    padding-left: 0px;
}

#about-app .video {
    width: 100%;
}

/* CONTACT PAGE */
#contact-details {
    flex-direction: column;
}
#contact-details .details {
    width: 100%;
    margin-bottom: 30px;
}
#contact-details .map {
    width: 100%;
}

#form-details {
    margin: 10px;
    padding: 30px 20px;

    flex-wrap: wrap;
}
#form-details form {
    width: 100%;
    margin-bottom: 30px;
}

/* SHOPING CART PAGE */
#cart-add {
    flex-direction: column;
}
#coupon {
    width: 100%;
```

```
    }
    #subtotal {
        width: 100%;
        padding: 20px;
    }

    /* FOOTER */
    #footer .columns .col {
        width: 50%;
    }
}

@media (max-width: 477px) {
    .section-p1 {
        padding: 20px;
    }

    #header {
        padding: 10px 10px;
    }

    h1 {
        font-size: var(--font-size-h3);
    }
    h2 {
        font-size: var(--font-size-h4);
    }

    #feature {
        justify-content: space-between;
    }
    #feature .fe-box {
        width: 155px;
        margin: 0 0 15px 0;
    }

    #product1 .pro {
        width: 100%;
    }

    #newsletter {
        padding: 40px 20px;
    }

    #newsletter .form {
        width: 100%;
    }

    /* SINGLE PRODUCT PAGE */

    #prodetails {
```

```
        display: flex;
        flex-direction: column;
    }
    #prodetails .single-pro-image {
        width: 100%;
        margin-right: 0px;
    }
    #prodetails .single-pro-details {
        width: 100%;
    }

    /* FOOTER */
    #footer .columns .col {
        width: 100%;
    }
}
```

7 PUG TEMPLATES (VIEWS AND PARTIALS)

views/about.pug

```
extends base

block content

  // page-header
  - var headerTitle = "#ABOUT"
  - var headerDescription = "Empowering your home with top-quality electrical goods. Discover our story and commitment to excellence. Welcome to our world of innovation and convenience."
  include partials/page-header

  // about-head
  include partials/about-head
  // about-app
  include partials/video
  // feature
  include partials/feature
  // newsletter
  include partials/newsletter
```

views/admin.pug

```
extends base

block content
```

```
block content

  // page-header
  - var headerTitle = "#ADMIN"
  - var headerDescription = "Empower your electrical goods store with our admin page. Manage inventory, orders, and more seamlessly. Simplify operations, maximize efficiency."

  include partials/page-header
  // blog
  include partials/admin-table
  // newsletter
  include partials/newsletter
```

views/base.pug

```
doctype html
html(lang="en")
  head
    meta(charset='UTF-8')
    meta(name='viewport' content='width=device-width, initial-scale=1.0')


    script(defer, src="https://cdn.jsdelivr.net/npm/@popperjs/core@2.9.1/dist/umd/popper.min.js", integrity="sha384-SR1sx49pcuLnqZUnnPwx6FCym0wLsk5JZuNx2bPPENzswTNFaQU1RDvt3wT4gWFG", crossorigin="anonymous")
    script(defer, src="https://cdn.jsdelivr.net/npm/bootstrap@5.0.0-beta3/dist/js/bootstrap.min.js", integrity="sha384-j0CNLUeiqtyaRmlzUHCPZ+Gy5fQu0dQ6eZ/xAww941Ai1SxSY+0EQqNXNE6DZiVc", crossorigin="anonymous")
    script(defer, src="https://kit.fontawesome.com/dbdd5a5067.js", crossorigin="anonymous")

    //- confetti effect
    script(defer, src="https://cdn.jsdelivr.net/npm/canvas-confetti@1.6.0/dist/confetti.browser.min.js")


    link(rel="stylesheet", href="https://pro.fontawesome.com/releases/v5.10.0/css/all.css")
    link(rel="stylesheet", href="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/css/bootstrap.min.css", integrity="sha384-MCw98/SFnGE8fJT3GXwEOngsV7Zt27NXFoaoApmYm81iuXoPkFOJwJ8ERdknLPMO", crossorigin="anonymous")

    link(rel="stylesheet", href="./../css/style.css")
```

```
    title= ' VB | ' + title

  body
    // header
    include partials/navbar

    block content

    // footer
    include partials/footer

    // modal-auth
    include partials/modal-auth

    script(defer src='./../bundle/index.js')
```

views/blog.pug

```
extends base

block content

  // page-header
  - var headerTitle = "#BLOG"
  - var headerDescription = "Discover the Latest Insights and Tips for Your Household Electrical Needs! Stay Updated with Expert Advice and Trends."
    include partials/page-header

  // blog
  include partials/blog-all

  // newsletter
  include partials/newsletter
```

views/cart.pug

```
extends base


block content
  script(src="https://js.stripe.com/v3/")

  // page-header
  - var headerTitle = "#CART"
  - var headerDescription = "Revolutionize your shopping experience with our user-friendly cart page. Seamlessly manage your selections for household electrical goods."
    include partials/page-header
```

```
  // modal-address
  include partials/modal-address

  // cart
  include partials/cart-table

  // purchase-history
  include partials/cart-history

  // newsletter
  include partials/newsletter
```

views/contact.pug

```
extends base

block content

  // page-header
  - var headerTitle = "#CONTACTS"
  - var headerDescription = "Get in touch with us for all your electrical needs! Our contact page is here to assist you promptly. Reach out for inquiries, support, or feedback. We're ready to serve you."
  include partials/page-header

  // contact-details
  include partials/contact-details

  // newsletter
  include partials/newsletter
```

views/error.pug

```
extends base

block content
  main.section-p1
    #error
      .title
        h2 Uh oh! Something went wrong! 👀
      p.msg= msg
```

views/home.pug

```
extends base

block content
    // hero
    include partials/hero

    // banner
    //- include partials/banner-1

    // sm-banner
    include partials/banner-2

    // newsletter
    include partials/newsletter
```

views/prod-single.pug

```
extends base

block content

  // pro-details
  include partials/prod-details
  // newsletter
  include partials/newsletter
```

views/shop.pug

```
extends base

block content

  // page-header
  - var headerTitle = "#SHOP"
  - var headerDescription = "Browse our extensive collection of household electrical goods. Find top-notch appliances to enhance your home's efficiency and comfort. Shop now for the latest in innovation and quality."
  include partials/page-header

  // filter pro
  include partials/prod-filter
  // product
  include partials/prod-all
  // pagination
  include partials/prod-pagi
```

```
// newsletter
include partials/newsletter
```

about-head.pug

```
section.section-p1
  #about-head
    img.image(src='../img/about.jpg', alt='')
    div
      h2 About Our Website
      p
        | This website was developed as part of a diploma thesis focusing on the creation of a web application to support the operations of a household electrical goods store. The project was undertaken by Vadim Bohulov, a student from Group IT-01 at Sumy State University in 2024.
      h3 Additional Information
      p
        | Here is some supplementary information about the student:
        br
        | Group: IT-01
        br
        | Specialty: Information Technology Design
        br
        | Department: Faculty of Electronics and Information Technologies.
      br
      br
      marquee(bgcolor='#eee', loop='-1') Stay tuned for more updates! Stay tuned for more updates! Stay tuned for more updates! Stay tuned for more updates!
```

admin-table.pug

```
#admin-panel.section-p1
  button#createProdBtn.normal Create new product

  .table-container
    table(width="100%")
      thead
        tr
          td REMOVE
          td Image
          td Product
          td Price
          td Quantity
          td Subtotal
          td Description
          td Details
          td Category
          td Created At
```

```
        td Updated At
      tbody
```

banner-1.pug

```
section#sm-banner.section-p1
  .banner-box
    h4 About the Project
    h2 Web Application for Household Electrical Goods Store
    span This website was developed as a part of a diploma thesis focusing on the creation of a web application to support the operations of a household electrical goods store. The project was undertaken by Vadim Bohulov, a student from Group IT-01 at Sumy State University in 2024.
    button.white Learn More

  .banner-box.banner-box2
    h4 About the Student
    h2 Vadim Bohulov: Student Profile
    span Vadim Bohulov is a student enrolled in the IT-01 group at Sumy State University, specializing in Information Technology Design under the Faculty of Electronics and Information Technologies.
    button.white Learn More
```

banner-2.pug

```
section#sm-banner.section-p1
  .banner-box
    h4 About the Project
    h2 Web Application for Household Electrical Goods Store
    span This website was developed as a part of a diploma thesis focusing on the creation of a web application to support the operations of a household electrical goods store. The project was undertaken by Vadim Bohulov, a student from Group IT-01 at Sumy State University in 2024.
    button.white Learn More

  .banner-box.banner-box2
    h4 About the Student
    h2 Vadim Bohulov: Student Profile
    span Vadim Bohulov is a student enrolled in the IT-01 group at Sumy State University, specializing in Information Technology Design under the Faculty of Electronics and Information Technologies.
    button.white Learn More
```

blog-all.pug

```
section#blog.section-p1
  .blog-box
```

```
        .blog-img
          img(src='../img/post1.jpg', alt='')
        .blog-details
          h4 Introduction to Household Electrical Goods
          p Learn about the latest trends and innovations in household electrical goods. From smart appliances to energy-efficient solutions, explore how technology is transforming homes.
          a(href='#') CONTINUE READING
          h1 11/30

      .blog-box
        .blog-img
          img(src='../img/post2.jpg', alt='')
        .blog-details
          h4 Maximizing Efficiency with Electrical Appliances
          p Discover practical tips and tricks for maximizing efficiency with your household electrical appliances. From proper maintenance to smart usage habits, optimize your devices for peak performance.
          a(href='#') CONTINUE READING
          h1 11/30

      .blog-box
        .blog-img
          img(src='../img/post3.jpg', alt='')
        .blog-details
          h4 Sustainable Solutions for Everyday Living
          p Explore sustainable solutions for everyday living with household electrical goods. Learn about eco-friendly appliances and practices that can help reduce your environmental footprint while enhancing your quality of life.
          a(href='#') CONTINUE READING
          h1 11/30
```

cart-history.pug

```
section#purchase-history.section-p1
  h4 Cart History Purchases
  p Here you can find a comprehensive list of your past cart history purchases. Explore the items you've previously purchased and reminisce about your shopping adventures. Whether it's for tracking your orders or simply for browsing, your purchase history is at your fingertips.

  if groupedPurchases.length > 0
    each groop in groupedPurchases
      p.date= groop.date
      .table-container
        table(width="100%")
          each item in groop.items
            tr
              td.img
                if item.product && item.product.images && item.product.images.length > 0
                  img(src=`/img/products/${item.product.images[0]}`, alt=item.product.name)
```

```
                else
                  img(src="/img/placeholder.jpg", alt="Product Image Unavailable")
                td.product-info
                  h5.name= item.product.name
                  p.description= item.product.description
                  if item.address
                    p.address= `Delivered to: ${item.address.fullName}, ${item.address.addressLine1}${item.address.addressLine2 ? ', ' + item.address.addressLine2 : ''}, ${item.address.city}, ${item.address.state}, ${item.address.postalCode}, ${item.address.country}`
                  p.time= new Date(item.purchaseDate).toLocaleTimeString('en-US', { hour: 'numeric', minute: 'numeric' })
                td.price
                  if item.quantity > 1
                    h5= `$${(item.product.price * item.quantity).toFixed(2)}`
                    p= `Qty ${item.quantity}, $${item.product.price} each`
                  else
                    h5= `$${(item.product.price * item.quantity).toFixed(2)}`

        else
          p You have no products in your purchase history.
```

cart-table.pug

```
section#cart.section-p1
  table(width="100%")
    thead
      tr
        td REMOVE
        td Image
        td Product
        td Price
        td Quantity
        td Subtotal
    tbody
      each item in cart
        // Check if purchaseDate is null or undefined
        if !item.purchaseDate
          tr(data-item-id=item._id)
            td
              //- Button for removing product
              a(href=`#`)
                i.fa-solid.fa-circle-minus
            td
              if item.product && item.product.images && item.product.images.length > 0
                img(src=`/img/products/${item.product.images[0]}`, alt=item.product.name)
              else
                img(src="/img/placeholder.jpg", alt="Product Image Unavailable")
            td
```

```
            a(href=`/prod-single/${item.product.slug}`)= item.product.name
          td $#{item.product.price.toFixed(2)}
          td
            // Example usage:
            input(type="number", value=item.quantity, min="0", max=item.product.quantity)
          td $#{(item.product.price * item.quantity).toFixed(2)}



section#cart-add.section-p1
  div#coupon
    h3 Apply Coupon
    div
      input(type="text", placeholder="Enter Your Coupon")
      button.normal Apply

  div#subtotal
    h3
      table
        tr
          td Cart Subtotal
          td
        tr
          td Shipping
          td
        tr
          td Total
          td
    button#checkout.normal Proceed to checkout
```

contact-details.pug

```
section#contact-details.section-p1
  .details
    span Contact Us
    h2 Get in Touch
    h3 Store Information
    div
    ul
      li
        i.fal.fa-map
        p Visit us at 123 Main Street, City, Country.
      li
        i.fal.fa-envelope
        p Email us at info@householdelectrics.com.
      li
        i.fal.fa-phone-alt
        p Call us at +1 234 567 8901.
      li
```

```
            i.fal.fa-clock
            p We're open Monday to Friday, 9:00 AM - 6:00 PM.
        .map
          iframe(

src="https://www.google.com/maps/embed?pb=!1m18!1m12!1m3!1d10066.913594499487!2d34.834988680324!3d50.89173702466618!2m3!1f0!2f0!3f0!3m2!1i1024!2i768!4f13.1!3m3!1m2!1s0x4128fe0120892805%3A0xb837b8752f41a97e!2z0KHRg9C80YHRjNC60LjQuSDQtNC10YDQttCw0LLQvdC40Lkg0YPQvdGW0LLQtdGA0YHQuNGC0LXRgg!5e0!3m2!1suk!2sua!4v1701341645793!5m2!1suk!2sua"
            width="600"
            height="450"
            style="border: 0"
            allowfullscreen=""
            loading="lazy"
            referrerpolicy="no-referrer-when-downgrade"
          )
      section#form-details.section-p1
        form(action="")
          span LEAVE A MESSAGE
          h2 We'd Love to Hear from You
          input(type="text", name="Your Name", placeholder="Your Name")
          input(type="text", name="E-mail", placeholder="Your E-Mail")
          input(type="text", name="Subject", placeholder="Your Subject")
          textarea(name="", id="", cols="30", rows="10", placeholder="Your Message")
          button.normal Submit
        .people
          div
            img(src="img/people/1.png", alt="")
            p
              span John Doe
              |  Sales Manager
              br
              | Phone: +1 234 567 8901
              br
              | Email: john@householdelectrics.com
          div
            img(src="img/people/2.png", alt="")
            p
              span Jane Smith
              |  Customer Support
              br
              | Phone: +1 234 567 8902
              br
              | Email: jane@householdelectrics.com
          div
            img(src="img/people/3.png", alt="")
            p
              span Michael Johnson
              |  Technical Support
              br
```

```
        | Phone: +1 234 567 8903
        br
        | Email: michael@householdelectrics.com
```

feature.pug

```
section#feature.section-p1
  .fe-box
    img(src="img/features/f1.png", alt="")
    h6 Free Shipping
  .fe-box
    img(src="img/features/f2.png", alt="")
    h6 Online Order
  .fe-box
    img(src="img/features/f3.png", alt="")
    h6 Save Money
  .fe-box
    img(src="img/features/f4.png", alt="")
    h6 Promotions
  .fe-box
    img(src="img/features/f5.png", alt="")
    h6 Happy Sell
  .fe-box
    img(src="img/features/f6.png", alt="")
    h6 24/7 Support
```

footer.pug

```
footer
  #footer.section-p1
    .columns

      .col
        h4 Contact
        p
          strong Address:
          |  123 Main Street
        p
          strong Phone:
          |  +1 (555) 123-4567
        p
          strong Hours:
          |  Mon-Fri: 9am-6pm
        .follow
          h4 Follow us
          .icon
            i.fab.fa-facebook-f
```

```
            i.fab.fa-pinterest-p
            i.fab.fa-youtube
            i.fab.fa-instagram
            i.fab.fa-twitter

        .col
          h4 About
          a(href="#") About Us
          a(href="#") Delivery Information
          a(href="#") Privacy Policy
          a(href="#") Terms & Conditions
          a(href="#") Contact Us

        .col
          h4 My Account
          a(href="#") Sign In
          a(href="#") View Cart
          a(href="#") My Wishlist
          a(href="#") Track My Order
          a(href="#") Help

      .copyright
        p © 2024 Household Electrical Store. All rights reserved.
```

hero.pug

```
section#hero
  .hero-container
    p Welcome to our store of household electrical goods.
    h3 Discover the latest innovations for your home.
    h1 Upgrade Your Living Space with Quality Appliances.
    p
      | Explore our wide selection of household electrical goods, carefully curated to enhance your daily life. From kitchen essentials to smart home solutions, we have everything you need to create a more comfortable and efficient home environment.
    a(href="/shop") Shop Now
```

modal-address.pug

```
#modal-address.modal-overlay
  .modal-container
    .modal-head
      h6 Delivery Address
      i#close-modal-address.fa-solid.fa-xmark

    form#form-address
      h5 Update Address
```

```
      p Please fill in the delivery address details below.

      .address-fields
        input.full-name(type='text', placeholder='Full Name', required='required')
        input.address-line1(type='text', placeholder='Address Line 1', required='required')
        input.address-line2(type='text', placeholder='Address Line 2')
        input.city(type='text', placeholder='City', required='required')
        input.state(type='text', placeholder='State', required='required')
        input.postal-code(type='text', placeholder='Postal Code', required='required')
        input.country(type='text', placeholder='Country', required='required')

      div.group
        button#update-address.normal(type='submit') Update Address
```

modal-auth.pug

```
#modal-auth.modal-overlay
  .modal-container
    if user
      .modal-head
        h6 Profile
        i#close-modal-auth.fa-solid.fa-xmark

      form#form-update
        h5 Update User Data
        p You can choose a new picture or revert to default. Feel free to customize the name and email fields as needed.


        .avatar
        // render dinamicaly in client js

        input.name(type='text', placeholder='Your Name', value=user.name)
        input.email(type='email', placeholder='Your E-mail', value=user.email)

        div.group
          button.normal(type='submit') Update
          button#logout.normal(type='button') Logout

    else
      .modal-head
        h6 Authorization
        i#close-modal-auth.fa-solid.fa-xmark

      form#form-signup.hidden
        h5 Sign Up
        p
          | Already have an account?
          br
```

```
      a.toggle-auth(href='#') Log In.
    input.name(type='text', placeholder='Your Name')
    input.email(type='email', placeholder='Your E-mail')
    input.password(type='password', placeholder='Your password')
    input.password-confirm(type='password', placeholder='Confirm your password')
    button.normal Sign Up

  form#form-login
    h5 Log In
    p
      | Don't have an account yet?
      br
      a.toggle-auth(href='#') Sign Up
      | first.
    input.email(type='email', placeholder='Your E-mail')
    input.password(type='password', placeholder='Your password')
    p
      | Forget the password?
      br
      | Click here to
      a.forget-password Reset Password
      | .
    button.normal Login
```

navbar.pug

```
section#header
  a(href="#")
    img.logo(src="/img/logo.png", alt="")

  div
    ul#navbar
      li
        a(class=title === 'Home' ? 'active' : '', href='/') Home

      // Add the following conditional statement for the 'Admin' section
      if user && user.role === 'admin'
        li
          a(class=title === 'Admin' ? 'active' : '', href='/admin') Admin

      li
        a(class=title === 'Shop' ? 'active' : '', href='/shop') Shop
      li
        a(class=title === 'Blog' ? 'active' : '', href='/blog') Blog
      li
        a(class=title === 'About' ? 'active' : '', href='/about') About
      li
        a(class=title === 'Contact' ? 'active' : '', href='/contact') Contact
      li
```

```
        a#nav-bag(class=title === 'Cart' ? 'active' : '', href='/cart')
          i.far.fa-shopping-bag
      li
        a#nav-auth
          if user
            i.fa-regular.fa-user
          else
            i.fa-solid.fa-right-to-bracket



    a#close(href='#')
      i.fa-solid.fa-xmark

  div#mobile
    ul
      li
        a#nav-bag(class=title === 'Cart' ? 'active' : '', href='/cart')
          i.far.fa-shopping-bag
      li
        a#nav-auth
          i.fa-regular.fa-user
      li
        a#bar(href='#')
          i.fas.fa-outdent
```

newsletter.pug

```
section#newsletter.section-p1
  .newslatter-container
    .newstext
      h4 Sign up for Newsletter
      p
        | Get E-mail updates about out latter
        br
        | shop and
        span Specific offers.
    .form
      input(type='text', placeholder='Your email address')
      button.normal Sign Up
```

page-header.pug

```
section#page-header(class=title.toLowerCase() + '-header')
  h2= headerTitle
```

```
  p= headerDescription
```

prod-all.pug

```
section#product1.section-p1
  .pro-container
```

prod-details.pug

```
if product
  section#prodetails.section-p1(data-product-id=product.id)
    .single-pro-image
      if product.images && product.images.length > 0
        img(src=`/img/products/${product.images[0]}`,      width="100%",      id="main-image",
alt="")
        .small-img-groop
          each imgSrc in product.images
            .small-img-col
              img(src=`/img/products/${imgSrc}`, width="100%", class="small-img", alt="")
      else
        img(src="./../img/placeholder.jpg", width="100%", id="main-image", alt="")
    .single-pro-details
      h5= product.name
      p= product.description
      h4 $#{product.price}
      if product.quantity > 0
        input(type="number",  name="quantity",  value=  1,  min="1",  max=product.quantity,
required)
        button(id="addToCartBtn", type="submit", class="normal") Add to cart
      else
        h5 Product is out of stock!

      if product && product.details
        div.details
          each paragraph in product.details.split('\n')
            p= paragraph


  section#reviews.section-p1
    form#reviewForm
      h5 Leave a Review and Help Us Improve
      p We value your feedback and use it to constantly improve our products and services.
Please take a moment to share your experience with us.
      div.rating
        p Choose rating:
        i.fa-solid.fa-star(data-rating="1")
        i.fa-solid.fa-star(data-rating="2")
```

```
          i.fa-solid.fa-star(data-rating="3")
          i.fa-solid.fa-star.active(data-rating="4")
          i.fa-regular.fa-star(data-rating="5")
        textarea.comment(name="comment" rows="4" cols="50" maxlength="300", placeholder="Enter your review here. \nWhat did you like most about this product? \nAny suggestions for improvement?")
        button.normal(type="submit") Submit Review

      h5 Find out what other people think about
        em "#{product.name}"

      p Explore user reviews for firsthand insights! Join satisfied customers and discover why our product stands out. Get real feedback to guide your decision-making. Dive in now!
      #reviews-container
```

prod-filter.pug

```
section#prod-filter.section-p1
  .categories
    if categories
      each category in categories
        button.cat-btn.white(data-category=category) #{category}

  .search
    .text-field
      input(type="search", placeholder="Search product name here...")
      div.clear
        i.fa-solid.fa-xmark
    button#search.normal Search

  .sorting
    .name(name='name', sort-type="none")
      span Name
      i.fa-solid.fa-arrow-up.hidden.asc
      i.fa-solid.fa-arrow-down.hidden.desc

    .price(name='price', sort-type="none")
      span Price
      i.fa-solid.fa-arrow-up.hidden.asc
      i.fa-solid.fa-arrow-down.hidden.desc

    .date(name='date', sort-type="none")
      span Date
      i.fa-solid.fa-arrow-up.hidden.asc
      i.fa-solid.fa-arrow-down.hidden.desc
```

prod-pagi.pug

```
section#prod-pagi.section-p1
```

video.pug

```
section#about-app.section-p1
  h1 Welcome to Our Online Store!
  div.video
    iframe(width="1080" height="900" src="https://www.youtube.com/embed/wDaZTy3z1Gs?autoplay=1&mute=1" title="YouTube video player" frameborder="0" allow="accelerometer; autoplay; clipboard-write; encrypted-media; gyroscope; picture-in-picture; web-share" referrerpolicy="strict-origin-when-cross-origin" allowfullscreen)
```

8 OTHER FILES

.env

```
NODE_ENV=development
PORT=3000

USER=vadim

DATABASE_PASSWORD=PPnfcfPwPcvl5tto
DATABASE=mongodb+srv://vadim:<PASSWORD>@cluster0.iqt8kjn.mongodb.net/

JWT_SECRET=secretcode
JWT_EXPIRES_IN=7776000000 #90d
JWT_COOKIE_EXPIRES_IN=604800000 #7d


# EMAIL
EMAIL_FROM=bogulov.bb@gmail.com
GMAIL_PASSWORD='rqob vguf iqos npxa'


# STRIPE
STRIPE_SECRET_KEY=sk_test_51OQAJeQhM6j92pI0WmNPweGNyT6MBOiVwNwqq71evBBKwmaGYfN63dRmjDjIHLB1bsbwogRmYX3BtbUVdabpX1ET00N0dUCRQs
STRIPE_WEBHOOK_SECRET=whsec_Fi3j4m7wPkC51o6nhZZjwkeQRrCjhBmh
```

package.json

```
{
```

```
    "name": "vadim_shop",
    "version": "1.0.0",
    "description": "",
    "main": "app.js",
    "targets": {
      "main": false
    },
    "scripts": {
      "dev": "set NODE_ENV=development && nodemon ./server.js",
      "start": "set NODE_ENV=production && node server.js",
      "debug": "ndb server.js",
      "build": "parcel build ./public/js/index.js --out-dir ./public/bundle --out-file index.js --public-url .",
      "watch": "parcel watch ./public/js/index.js --out-dir ./public/bundle --out-file index.js --public-url ."
    },
    "author": "Vadym Bohulov",
    "license": "ISC",
    "dependencies": {
      "axios": "^1.6.2",
      "bcrypt": "^5.1.1",
      "bcryptjs": "^2.4.3",
      "body-parser": "^1.20.2",
      "canvas-confetti": "^1.9.2",
      "compression": "^1.7.4",
      "cookie-parser": "^1.4.6",
      "cors": "^2.8.5",
      "dotenv": "^16.3.1",
      "ejs": "^3.1.9",
      "express": "^4.18.3",
      "express-mongo-sanitize": "^2.2.0",
      "express-rate-limit": "^7.1.5",
      "helmet": "^7.1.0",
      "hpp": "^0.2.3",
      "html-to-text": "^9.0.5",
      "jsonwebtoken": "^9.0.2",
      "mailgen": "^2.0.28",
      "mongoose": "^8.0.2",
      "morgan": "^1.10.0",
      "multer": "^1.4.5-lts.1",
      "nodemailer": "^6.9.13",
      "pug": "^3.0.2",
      "save-dev": "^0.0.1-security",
      "sharp": "^0.33.2",
      "slugify": "^1.6.6",
      "stripe": "^15.1.0",
      "validator": "^13.11.0",
      "xss-clean": "^0.1.4"
    },
    "devDependencies": {
      "buffer": "^6.0.3",
```

```
        "nodemon": "^3.0.2",
        "parcel-bundler": "^1.12.5"
    },
    "engines": {
        "node": "20.9.0"
    }
}
```