МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ “МЕДИЧНІ ТОВАРИ”»**

Виконав: студент 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
В.О. Веретельніков
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.т.н. Мухін В.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент завідувач кафедри фундаментальної та прикладної математики, професор, д.т.н. Гребенюк С.М.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Веретельнікову Валерію Олеговичу
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка інтернет-магазину «Медичні товари»

керівник роботи Мухін Віталій Вікторович, к.т.н., доцент
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування.
3. Реалізація та тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

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

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	22.01.2024	
2.	Збір вихідних даних.	14.02.2024	
3.	Обробка методичних та теоретичних	11.03.2024	
	джерел.		
4.	Розробка першого та другого розділу.	15.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2024	

Студент _____ В.О. Веретельніков
(підпис) (ініціали та прізвище)

Керівник роботи _____ В.В. Мухін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інтернет-магазину «Медичні товари»»: 54 с., 28 рис., 1 табл., 13 джерел, 3 додатки.

ECOMMERCE, FRAMEWORK, LARAVEL, MVC, MYSQL, PHP, UML, UNIT.

Об’єкт дослідження – система, інструменти для роботи з базами даних та платіжними системами.

Мета роботи – розробити інтернет-магазин медичних товарів.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

У роботі розглянуто проєктування та реалізацію системи управління замовленнями для інтернет-магазину медичних товарів з використанням фреймворку Laravel. Застосовано UML для візуалізації системи, зокрема діаграми варіантів використання, діяльності, послідовності та розгортання.

Система надає можливість реєстрації користувачів, управління кошиком, оформлення замовлень, вибору платіжних методів, керування товарами та користувачами. Реалізовано функції для обробки замовлень, керування каталогом товарів, інтеграцію з платіжними системами та забезпечення безпеки даних. Проведено юніт- та інтеграційне тестування.

Таким чином, розроблена система забезпечує зручний інтерфейс для користувачів інтернет-магазину, дозволяючи легко оформлювати замовлення, управляти кошиком та отримувати консультації щодо товарів. Застосування фреймворку Laravel та сучасних підходів до розробки програмного забезпечення сприяло створенню ефективної та масштабованої системи.

SUMMARY

Bachelor’s qualifying paper “Development of the "Medical products" Online Store”: 54 pages, 28 figures, 1 table, 13 references, 3 supplements.

ECOMMERCE, FRAMEWORK, LARAVEL, MVC, MYSQL, PHP, UML, UNIT.

The object of the study is a system and tools for working with databases and payment systems.

The aim of the study is to develop an online medical products store.

The methods of research are modeling, design, programming, analytical.

The paper presents a discussion of the design and implementation of an order management system for an online medical supply store, utilizing the Laravel framework. The Unified Modeling Language (UML) is employed to illustrate the system's architecture, encompassing use case diagrams, activities, sequence, and deployment.

The system enables users to register, manage their shopping carts, place orders, select payment methods, and manage products and users. It also incorporates functions for order processing, product catalog management, integration with payment systems, and data security. Furthermore, unit and integration testing were conducted.

Consequently, the developed system offers a user-friendly interface for online store users, enabling them to effortlessly place orders, manage their shopping carts, and obtain product advice. The utilisation of the Laravel framework and contemporary methodologies in software development has resulted in the creation of an effective and scalable system.

ЗМІСТ

ВСТУП

Електронна комерція, особливо у сфері медичних товарів, стала важливою складовою сучасного ринку. Щоденно мільйони людей у всьому світі здійснюють онлайн-покупки, замовляючи медичні товари та послуги через інтернет. Проте часто користувачі стикаються з проблемами при оформленні замовлень, управлінні кошиком, вибором способів оплати та отриманням консультацій щодо товарів. Існуючі платформи не завжди задовольняють специфічні потреби користувачів, що створює незручності та обмежує можливості для комфортного онлайн-шопінгу.

Покупці шукають зручні та орієнтовані на їхні потреби платформи для здійснення онлайн-покупок медичних товарів. Наявність такої системи дозволить задовольнити цю потребу та забезпечити комфортне середовище для користувачів інтернет-магазинів.

Інтернет-магазин медичних товарів дозволить користувачам легко оформлювати замовлення, управляти кошиком, вибирати способи оплати та отримувати консультації щодо товарів. Можливість інтеграції системи з популярними платіжними платформами дозволить користувачам легко здійснювати платежі, збільшуючи зручність та безпеку онлайн-покупок.

Виходячи з цього, було вирішено створити інтернет-магазин медичних товарів, який був би простим у використанні, безпечним та функціональним.

Актуальність дослідження: актуальність теми зумовлена потребою створення спеціалізованої платформи для управління замовленнями в інтернет-магазинах медичних товарів.

З огляду на це, можна виділити наступні цілі і задачі нашого дослідження:

Мета: розробити інтернет-магазин медичних товарів.

Задачі:

- сформувати вимоги до системи;
- спроєктувати та побудувати архітектуру системи;

– реалізувати інтернет-магазин медичних товарів;

– протестувати роботу системи.

Об’єкт дослідження:процес процес розробки системи управління замовленнями для інтернет-магазину медичних товарів, інструменти для роботи з базами даних та платіжними системами.

Предмет дослідження: фреймворк Laravel та бібліотеки JavaScript.

Методи дослідження: моделювання, проєктування, програмний, аналітичний.

Перший розділ присвячено збору та аналізуванню вимог до системи, огляду подібних рішень і опису функціональних можливостей системи.

У другому розділі розглянуто етапи проєктування системи, наведено детальний опис прецедентів та побудовано діаграми.

Третій розділ присвячено реалізації та тестуванню роботи системи, наведено керівництво користувача, яке описує процес роботи з системою управління замовленнями для інтернет-магазину медичних товарів.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

1.1.1 Загальні терміни

Система – інтернет-магазин «Медичні товари» на основі фреймворку Laravel.

Laravel – це PHP-фреймворк з відкритим кодом, який використовується для створення вебдодатків.

ВВ – Варіант Використання чи Use Case.

ДВВ – Діаграма Варіантів Використання.

ДП – Діаграма Послідовності.

ДД – Діаграма Діяльності.

ДР – Діаграма Розгортання.

Відвідувач – не зареєстрований або неавторизований користувач системи.

Клієнт – користувач, який авторизувався в системі.

Адміністратор – користувач, який має права на керування системою.

1.1.2 Технічні терміни

БД – база даних, місце збереження інформації системи.

Framework – це набір інструментів, бібліотек та рекомендацій, призначених для розробки програмного забезпечення.

1.2 Функціональні вимоги

1.2.1 Призначення і цілі створення системи

Функціональне призначення системи – реалізувати функціонал інтернет-магазину з продажу медичних товарів.

Експлуатаційне призначення системи – система може експлуатуватися користувачами на різних рівнях, залежно від ролі та наданих прав.

Мета створення системи – розробка інтернет-магазину «Медичні товари».

1.2.2 Загальні функціональні можливості системи

Система має надавати відвідувачам такі можливості:

- реєстрація в системі;
- вхід до системи;
- перегляд каталогу товарів;
- перегляд товару;
- пошук/фільтрація товарів.

Система має надавати клієнтам такі додаткові можливості:

- додавання/видалення товару до/з кошика;
- вказання кількості одиниць товару;
- оформлення замовлення;
- вказання даних для доставки;
- вибір платіжного методу;
- вихід з системи.

Система має надавати адміністраторам наступні можливості:

- керування замовленнями;
- керування категоріями товарів;

- керування товарами;
- керування брендами;
- керування користувачами.

1.3 Нефункціональні вимоги

1.3.1 Інтерфейс користувача

Система повинна мати адаптивний інтерфейс, який забезпечить оптимальний і зручний досвід роботи для користувачів незалежно від пристрою, на якому вони працюють – чи то настільний комп’ютер, ноутбук, планшет або смартфон. Такий підхід гарантує безперешкодне використання системи для всіх категорій користувачів: від звичайних відвідувачів до адміністраторів.

1.3.2 Підтримка браузерів

Система повинна бути сумісною з найновішими версіями провідних веббраузерів, таких як Mozilla Firefox, Google Chrome, Safari, Microsoft Edge та Opera. Це гарантуватиме безперебійну і належну роботу для користувачів, незважаючи на те, який саме браузер вони використовують для доступу до системи.

1.3.3 Вимоги до продуктивності

Важливими вимогами до системи є висока швидкодія та ефективність у роботі з даними. Сторінки мають завантажуватися блискавично, не більше однієї секунди. Процес оформлення замовлень не повинен перевищувати 3 секунди.

Система також має демонструвати здатність безперебійно опрацьовувати великі масиви інформації, включно із завантаженням та візуалізацією значних обсягів даних, при цьому не знижуючи загальну продуктивність роботи.

1.3.4 Вимоги до безпеки

Система повинна надійно захищати особисту інформацію та платіжні дані від несанкціонованого доступу. Неавторизовані відвідувачі не можуть переглядати конфіденційну інформацію чи здійснювати адміністративні дії.

Крім того, система не повинна зберігати чи використовувати платіжні дані покупців без їхньої явної згоди. Під час оформлення замовлення клієнти самостійно вводитимуть платіжні реквізити, які використовуватимуться лише для проведення поточної транзакції.

1.4 Опис предметної області

Предметною областю є розробка онлайн-магазина. Система надає користувачам можливість переглядати каталог товарів, вибирати потрібні продукти, додавати їх до кошика та здійснювати покупки. Функціонал системи залежить від ролі користувача: гості можуть лише переглядати асортимент, а зареєстровані користувачі мають додаткові можливість взаємодіяти з кошиком та робити замовлення.

Процес оформлення замовлення відбувається наступним чином. По-перше, користувач повинен зареєструватися або увійти до свого облікового запису. Потім він переходить до каталогу, вибирає бажані товари та додає їх до кошика. У кошику можна детально переглянути опис товарів, фотографії та характеристики. Далі користувач має можливість змінити кількість або видалити непотрібні товари, а також побачити загальну вартість замовлення.

Якщо вартість влаштовує, користувач вводить дані для доставки та обирає спосіб оплати. Після заповнення форми замовлення, він натискає кнопку оформлення, і система обробляє замовлення та зберігає інформацію в базі даних. Наостанок, користувач отримує електронний лист з підтвердженням замовлення та відомостями для відстеження доставки.

1.5 Опис системи

Інтернет-магазин, пропонує онлайн-платформу для продажу медичного обладнання, ліків та інших супутніх товарів широкому колу споживачів. Ця система надає зручний інтерфейс як для кінцевих користувачів, так і для адміністраторів, що включає управління каталогом товарів, обробку замовлень, інтеграцію з платіжними системами, логістику доставки та підтримку клієнтів.

Інтернет-магазин має бути масштабованим та адаптивним, здатним витримувати високе навантаження та забезпечувати високу доступність сервісів для користувачів з різних регіонів. Забезпечення інтуїтивно зрозумілого користувацького інтерфейсу, надійності операцій та високої швидкості обробки даних є ключовими для успішної роботи такої системи.

Фреймворк Laravel забезпечує необхідний функціонал та можливості для створення потужного та гнучкого інтернет-магазину медичних товарів, здатного задовольнити потреби як покупців, так і адміністраторів.

1.6 Огляд і порівняння аналогів

Щоб детально зрозуміти переваги та недоліки нашої системи, розглянемо декілька популярних онлайн-магазинів з продажу медичних товарів: Apteka24, MedMag, Pharmex, Tabletki.ua, 1mg.

Apteka24 – одина з найбільших українських онлайн-аптек, пропонує широкий асортимент медичних товарів, включаючи ліки, медичні прилади, косметику та товари для здоров’я. Відрізняється зручним інтерфейсом, швидкою доставкою та можливістю консультацій з фармацевтами [1].

MedMag спеціалізується на продажу медичної техніки та обладнання. Пропонує широкий вибір товарів для професійного та домашнього використання, включаючи апарати для діагностики, реабілітації та догляду за хворими. Має зручну навігацію та інформаційні матеріали для користувачів [7].

Pharmex – онлайн-аптека з акцентом на продажі ліків та медичних препаратів. Пропонує знижки, акції та програму лояльності для постійних клієнтів. Відрізняється широким асортиментом і оперативною доставкою по всій Україні [13].

Tabletki.ua – один з найпопулярніших українських онлайн-сервісів для пошуку та замовлення ліків та медичних товарів. Пропонує функцію порівняння цін та наявності товарів у різних аптеках [10].

1mg – індійський онлайн-аптечний сервіс, що пропонує широкий вибір медичних товарів, консультації лікарів та лабораторні тести. Має мобільний додаток та програму лояльності [11].

Проаналізувавши інші інтернет-магазини, можемо порівняти їх з нашою системою (табл. 1.1).

Порівняльний аналіз показує, що наша система забезпечує основні функції для продажу медичних товарів, однак поступається конкурентам в аспектах додаткової функціональності, підтримки клієнтів та SEO-оптимізації. Інші магазини, такі як Apteka24, MedMag, Pharmex, Tabletki.ua та 1mg, пропонують розширені функції, такі як онлайн-консультації, програми лояльності та професійні рекомендації, що підвищують їх привабливість для користувачів. Впровадження подібних покращень у наш магазин може значно підвищити його конкурентоспроможність та привабливість для клієнтів.

Таблиця 1.1 – Порівняння аналогів

Характеристика	Платформа	Функціональність	Дизайн	SEO	Підтримка клієнтів
Наша Система	Laravel	Основні функції	Адаптивний, сучасний	Основні практики	Електронна пошта
Apteka24	Власна розробка	Онлайн-консультації	Зручний, UX/UI	Висока оптимізація	Онлайн-чат, телефон
MedMag	Magento	Професійні рекомендації	Професійний	Висока оптимізація	Телефон, електронна
Pharmex	OpenCart	Програма лояльності	Орієнтований на користувача	Висока оптимізація	Онлайн-чат, телефон
Tabletki.ua	Власна розробка	Порівняння цін	Зручний, UX/UI	Висока оптимізація	Онлайн-чат, телефон
1mg	Власна розробка	Консультації лікарів, лабораторні тести	Зручний, UX/UI	Висока оптимізація	Онлайн-чат, телефон

2 ПРОЄКТУВАННЯ

2.1 Використання UML під час розробки системи

Використання уніфікованої мови моделювання (UML) під час розробки системи є критично важливим для забезпечення послідовного та ефективного процесу створення програмного забезпечення. UML є стандартизованою графічною нотацією, яка дозволяє розробникам візуалізувати різні аспекти системи на різних етапах її життєвого циклу. Цей підхід сприяє кращому розумінню вимог, архітектури та поведінки системи, забезпечуючи спільну платформу для комунікації між усіма зацікавленими сторонами проєкту [12].

Застосування UML на початкових етапах розробки допомагає точно визначити вимоги до системи та зафіксувати їх у зрозумілій формі за допомогою діаграм прецедентів використання та діаграм діяльності. Ці діаграми забезпечують наочну ілюстрацію функціональності системи та взаємодії між різними суб’єктами, полегшуючи процес узгодження вимог та виявлення потенційних проблем на ранніх стадіях проєкту [12].

На етапі проєктування UML відіграє ключову роль у моделюванні структури системи та визначенні її компонентів. Діаграми класів, діаграми компонентів та діаграми розгортання допомагають розробникам організувати логічну архітектуру системи, визначити взаємозв’язки між класами та компонентами, а також спланувати розміщення елементів системи на фізичних вузлах [12].

Під час імплементації системи UML забезпечує ефективний опис поведінки окремих компонентів та їхньої взаємодії. Діаграми послідовності та діаграми комунікації дозволяють візуалізувати потік повідомлень між об’єктами, що полегшує розуміння динамічної поведінки системи та спрощує процес її тестування та налагодження.

Нарешті, UML допомагає документувати та передавати знання про

систему іншим розробникам або майбутнім командам підтримки. Модельований підхід забезпечує чітке та формальне представлення всіх аспектів системи, що сприяє ефективному обміну інформацією та полегшує процес супроводу та модернізації системи в майбутньому.

2.2 Діаграма варіантів використання

Діаграма варіантів використання є одним з ключових артефактів уніфікованої мови моделювання (UML), що використовується для візуалізації функціональних вимог до системи з перспективи різних акторів, які взаємодіють із нею. Ця діаграма являє собою графічне представлення набору дій або послідовностей взаємодій між системою та її зовнішніми користувачами або іншими системами, відомими як актори [12].

У центрі діаграми варіантів використання знаходиться система, яка моделюється у вигляді прямокутника або еліпса. Навколо системи розташовані актори, зображені у вигляді фігурок людини або іншої умовної позначки, що репрезентує зовнішню сутність, яка взаємодіє з системою. Кожен актор пов’язаний з одним або кількома варіантами використання, що являють собою окремі функціональні можливості або послуги, які система надає акторам [12].

Варіанти використання зображуються у формі овалів, а їх назви відображають дії або завдання, які можуть виконувати актори за допомогою системи. Зв’язки між акторами та варіантами використання вказують на те, хто може ініціювати або брати участь у певній послідовності дій. Ці зв’язки можуть бути доповнені текстовими описами або примітками для додаткового пояснення ролі кожного актора у відповідному варіанті використання [12].

Діаграма варіантів використання є корисним інструментом для комунікації між замовниками, кінцевими користувачами та розробниками, оскільки вона надає зрозумілий та наочний спосіб представлення функціональних вимог до системи. Вона допомагає виявити прогалини або неузгодженості у вимогах, а

також слугує основою для подальшого проєктування та розробки системи.

Діаграма варіантів використання, зображена на рисунку 2.1, ілюструє основні функціональні можливості, доступні різним акторам у контексті головного бізнес-процесу «Оформити замовлення». Крім того, ця діаграма відображає взаємозв’язки та взаємодію між цим ключовим процесом і суміжними або допоміжними процесами, що забезпечують повноцінну підтримку його реалізації.

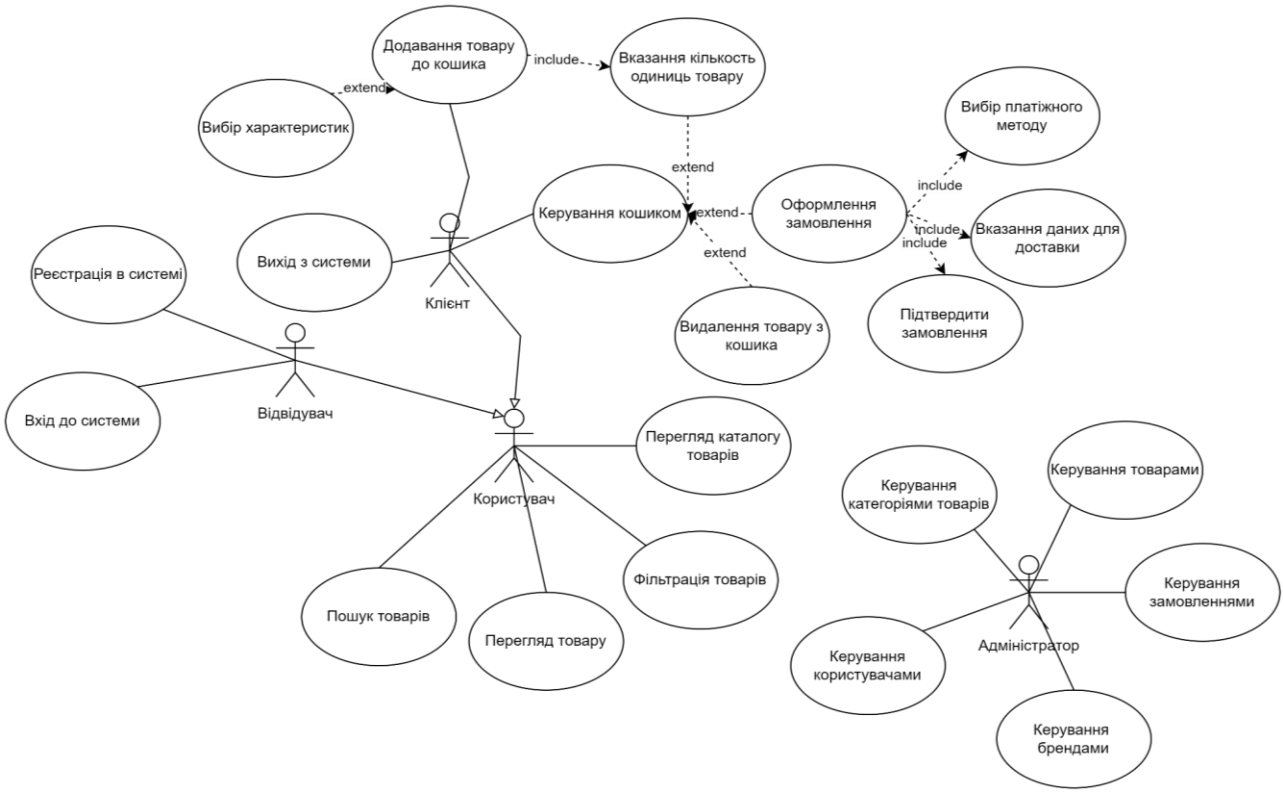

Рисунок 2.1 – Діаграма варіантів використання

На діаграмі представлено акторів «Відвідувач», «Клієнт» та «Адміністратор», кожен з яких може взаємодіяти з системою, згідно ролі та наданих прав. Також представлено актора «Користувач», який включає в себе спільні функціональні можливості клієнта та відвідувача, як користувачів системи.

Центральним варіантом використання є «Оформлення замовлення», який представляє головний бізнес-процес системи. Цей варіант використання

ініціюється актором «Клієнт» і охоплює кілька пов’язаних підпроцесів та дій. Перш за все, клієнт взаємодіє з варіантами використання «Перегляд каталогу товарів» та «Додавання товару до кошика», які дозволяють йому вибрати бажані товари та додати їх до віртуального кошика. Під час цього процесу клієнт має доступ до детальної інформації про товари, включаючи опис, фотографії та характеристики.

Після вибору товарів клієнт переходить до варіанту використання «Керування кошиком», де може змінювати кількість або видаляти товари з кошика, а також бачити загальну вартість замовлення.

Наступним етапом є варіант використання «Вибір платіжного методу», де клієнт обирає метод оплати і вводить платіжні дані. Оскільки клієнт авторизований, його особисті дані та інформація про адресу доставки автоматично заповнюється з профіля користувача.

Після введення всіх необхідних даних клієнт ініціює варіант використання «Підтвердити замовлення», натискаючи відповідну кнопку. Система обробляє замовлення та зберігає всю інформацію в базі даних.

По завершенню оформлення замовлення клієнт отримує електронний лист з підтвердженням замовлення та інформацією для відстеження доставки.

Ця діаграма надає зрозумілу візуалізацію основних елементів процесу оформлення замовлення, включаючи взаємодію клієнта з різними функціональними можливостями системи на кожному етапі.

Варіанти використання є ключовим елементом в уніфікованій мові моделювання (UML) для опису функціональних вимог до системи з точки зору її користувачів та зовнішніх акторів. Кожен варіант використання представляє певний сценарій або послідовність дій, які актор може виконати для досягнення конкретної мети під час взаємодії з системою.

Варіанти використання детально описують, як кінцеві користувачі або зовнішні системи можуть використовувати функціональні можливості системи, що розробляється. Вони визначають різні шляхи, за якими актори взаємодіють з системою для вирішення своїх завдань або задоволення потреб.

Кожен варіант використання є окремим сценарієм, який складається з послідовності кроків або дій, необхідних актору для реалізації цільової діяльності в межах системи. Ці сценарії відображають різноманітні способи застосування функціоналу системи акторами для досягнення своїх цілей.

Таким чином, варіанти використання забезпечують ефективний механізм для моделювання та документування поведінкових вимог до системи з перспективи її користувачів та зовнішніх сутностей. Вони наочно демонструють різні сценарії взаємодії акторів з відповідними функціональними можливостями системи, що є критично важливим для повного розуміння та узгодження вимог між усіма зацікавленими сторонами проєкту (рис. 2.2 – 2.3).

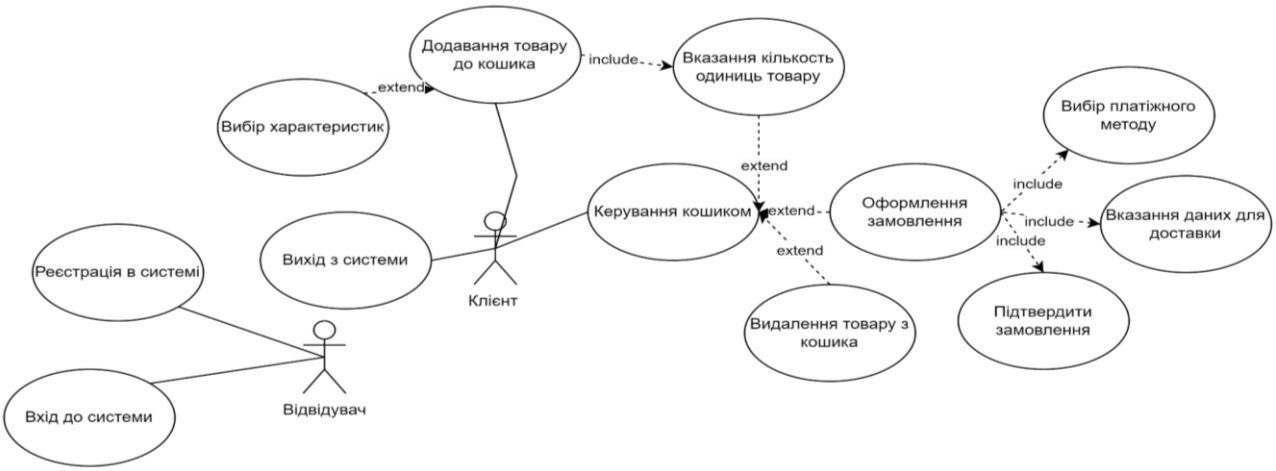

Рисунок 2.2 – ДВВ акторів «Відвідувач» та «Клієнт»

Рисунок 2.3 – ДВВ акторів «Користувач» та «Адміністратор»

2.2.1 Опис варіантів використання

Прецедент «Реєстрація в системі».

Призначення: даний варіант використання надає можливість відвідувачу зареєструватися в системі.

Основний потік подій: цей варіант використання розпочинається, коли відвідувач вибирає пункт меню «Зареєструватися». Система відображає форму реєстрації. Після введення даних, відвідувач натискає кнопку «Зареєструватися», і система створює нового користувача та зберігає його дані.

Виняткова ситуація 1: невалідні дані – відображається повідомлення про невалідні дані, відвідувач може повторно ввести дані.

Виняткова ситуація 2: користувач вже зареєстрований – відображається повідомлення про наявність користувача з таким email, відвідувач може змінити дані.

Виняткова ситуація 3: помилка сервера – відображається відповідне повідомлення, створення нового користувача не відбувається.

Прецедент «Вхід до системи».

Призначення: даний варіант використання надає можливість відвідувачу авторизуватися в системі.

Основний потік подій: цей варіант використання починається, коли відвідувач вибирає пункт меню «Вхід». Система відображає форму входу. Після введення електронної пошти та пароля, відвідувач натискає кнопку «Вхід», і система відкриває сторінку профілю користувача.

Передумова: перед початком виконання даного варіанта використання користувач повинен зареєструватися в системі.

Виняткова ситуація 1: невалідні дані – відображається повідомлення про невалідні дані, відвідувач може повторно ввести дані.

Виняткова ситуація 2: користувача не існує – відображається повідомлення про те, що користувача з таким email не існує в системі, відвідувач може змінити дані.

Виняткова ситуація 3: помилка сервера – відображається відповідне повідомлення, вхід до системи не відбувається.

Прецедент «Вихід з системи».

Призначення: даний варіант використання надає можливість клієнту виходити з системи.

Основний потік подій: цей варіант використання виконується, коли авторизований користувач натискає на своє ім’я в профілі та обирає пункт меню «Вихід». Система видаляє дані про сесію користувача та відображає головну сторінку.

Прецедент «Перегляд каталогу товарів».

Призначення: даний варіант використання надає можливість користувачу переглядати каталог товарів.

Основний потік подій: цей варіант використання починається, коли користувач вибирає будь-який пункт основного або навігаційного меню, пов’язаного з товаром. Система відображає сторінку з товарами відповідної категорії.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, сторінка каталогу товарів не відкривається.

Прецедент «Перегляд товару».

Призначення: даний варіант використання надає можливість користувачу переглядати картку товару.

Основний потік подій: цей варіант використання починається, коли користувач натискає на назву товару. Система відкриває сторінку обраного товару, де користувач може ознайомитися з описом та взаємодіяти з товаром.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці каталогу товарів.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, сторінка товару не відкривається.

Прецедент «Фільтрація товарів».

Призначення: даний варіант використання надає можливість користувачу

фільтрувати товари.

Основний потік подій: цей варіант використання запускається, коли користувач обирає один із фільтрів на лівій панелі. Система відображає сторінку з товарами, які відповідають застосованим критеріям фільтрації.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці каталогу товарів.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, фільтрація не відбувається.

Прецедент «Пошук товарів».

Призначення: даний варіант використання надає можливість користувачу шукати товари за назвою.

Основний потік подій: цей варіант використання починається, коли користувач вводить повну або часткову назву товару в поле пошуку та натискає на піктограму пошуку. Система відображає товари, які відповідають критеріям пошуку.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, пошук не відбувається.

Прецедент «Додавання товару до кошика».

Призначення: даний варіант використання надає можливість клієнту додавати обрані товари до кошика.

Основний потік подій: цей варіант використання виконується, коли клієнт обирає характеристики товару, вводить кількість і натискає кнопку «Додати до кошика» на сторінці товару, або натискає на відповідну кнопку на сторінці каталогу товарів. Система додає товар до кошика та оновлює зображення кількості товарів біля піктограми кошика.

Передумова: перед початком виконання даного варіанта використання клієнт повинен знаходитися на сторінці каталогу товарів або на сторінці товару.

Виняткова ситуація 1: товару немає в наявності – відображається відповідне повідомлення, товар не додається до кошика.

Виняткова ситуація 2: помилка сервера – відображається відповідне

повідомлення, товар не додається до кошика.

Прецедент «Вказання кількості одиниць товару».

Призначення: даний варіант використання надає можливість клієнту вказати кількість одиниць товару для додання до кошика.

Основний потік подій: цей варіант використання починається, коли клієнт вводить значення в поле «Кількість одиниць товару». Система запам’ятовує кількість товару для додавання до кошика. При зміні кількості в кошику система автоматично перераховує суму замовлення.

Передумова: перед початком виконання даного варіанта використання користувач повинен знаходитися на сторінці товару.

Прецедент «Керування кошиком».

Призначення: даний варіант використання надає можливість клієнту взаємодіяти з кошиком.

Основний потік подій: цей варіант використання запускається, коли клієнт натискає на піктограму кошика. Система відкриває інтерфейс керування кошиком.

Альтернативний потік подій: якщо жодного товару не було додано, то система відображає відповідне повідомлення та пропонує перейти до вибору товарів.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, кошик не відкривається.

Прецедент «Видалення товару з кошика».

Призначення: даний варіант використання надає можливість клієнту видаляти товари з кошика.

Основний потік подій: цей варіант використання починається, коли клієнт натискає на кнопку «Видалити» в картці відповідного товару. Система видаляє товар з кошика та перераховує суму замовлення.

Передумова: перед початком виконання даного варіанта використання клієнт повинен додати як мінімум один товар до кошика та знаходитись на сторінці керування кошиком.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, товар не видаляється.

Прецедент «Оформлення замовлення».

Призначення: даний варіант використання надає можливість клієнту оформити замовлення.

Основний потік подій: цей варіант використання запускається, коли клієнт натискає кнопку «Перейти до оформлення». Система відкриває форму оформлення, де клієнт обирає метод оплати і вводить платіжні дані. Оскільки він вже авторизований, його особисті дані та інформація про адресу доставки автоматично заповнюється з профіля. За необхідності, клієнт може їх змінити. Після заповнення форми і натискання «Підтвердити замовлення» система обробляє замовлення, зберігає інформацію в базі даних і надсилає користувачу електронний лист з підтвердженням та даними для відстеження доставки.

Передумова: перед початком виконання даного варіанта використання клієнт повинен додати як мінімум один товар до кошика та знаходитись на сторінці керування кошиком.

Виняткова ситуація 1: помилка сервера – відображається відповідне повідомлення, замовлення не оформлюється.

Виняткова ситуація 2: невалідні дані – відображається повідомлення про невалідні дані, клієнт може повторно ввести дані.

Виняткова ситуація 3: сервіс оплати не відповідає – відображається відповідне повідомлення, клієнт може спробувати знов або змінити метод оплати.

Виняткова ситуація 4: недостатньо коштів на рахунку – відображається відповідне повідомлення, клієнт може спробувати знов або змінити реквізити картки.

Виняткова ситуація 5: товару немає в наявності – відображається відповідне повідомлення, клієнт може оформити замовлення знов.

Прецедент «Вибір платіжного методу».

Призначення: даний варіант використання надає можливість клієнту

вибрати платіжну систему.

Основний потік подій: Цей варіант використання запускається, коли клієнт обирає один з платіжних методів з переліку. Система відображає форму для введення реквізитів (за потреби).

Прецедент «Вказання даних для доставки».

Призначення: даний варіант використання надає можливість клієнту ввести або підтягнути дані для доставки.

Основний потік подій: Цей варіант використання починається, коли клієнт переходить до форми оформлення замовлення, система підтягує збережені дані для доставки (ПІБ, адреса) у відповідні поля форми оформлення замовлення. За потреби, клієнт може змінити ці дані. Система запам’ятовує ці дані.

Виняткова ситуація 1: невалідні дані – відображається повідомлення про невалідні дані, клієнт може повторно ввести дані.

Прецедент «Вибір характеристик».

Призначення: даний варіант використання надає можливість клієнту вибирати характеристики товару перед додаванням його до кошика.

Основний потік подій: цей варіант використання починається, коли клієнт переглядає сторінку товару. Система відображає доступні варіанти характеристик товару, такі як колір, розмір чи інші опції. Клієнт вибирає бажані характеристики, і система оновлює відображення товару відповідно до вибраних опцій.

Прецедент «Підтвердити замовлення».

Призначення: даний варіант використання дозволяє клієнту завершити оформлення замовлення після введення всіх необхідних даних.

Основний потік подій: цей варіант використання запускається, коли клієнт натискає кнопку «Підтвердити замовлення» на формі оформлення замовлення. Система перевіряє введені дані на коректність, обробляє платіж (якщо застосовно) та створює замовлення. Клієнту надсилається підтвердження про успішне оформлення замовлення.

Виняткова ситуація 1: помилка сервера – відображається відповідне

повідомлення, замовлення не оформлюється.

Прецедент «Керування товарами».

Призначення: даний варіант використання дозволяє адміністратору керувати товарами в системі, включаючи додавання, редагування та видалення товарів.

Основний потік подій: цей варіант використання починається, коли адміністратор переходить до розділу керування товарами. Система відображає список наявних товарів з можливістю їх фільтрації та пошуку. Адміністратор може додавати нові товари, редагувати інформацію про існуючі або видаляти товари з системи.

Прецедент «Керування користувачами».

Призначення: даний варіант використання дозволяє адміністратору керувати обліковими записами користувачів у системі.

Основний потік подій: цей варіант використання запускається, коли адміністратор переходить до розділу керування користувачами. Система відображає список зареєстрованих користувачів з можливістю їх фільтрації та пошуку. Адміністратор може переглядати деталі облікових записів.

Прецедент «Керування категоріями товарів».

Призначення: даний варіант використання дозволяє адміністратору керувати категоріями товарів у системі.

Основний потік подій: цей варіант використання починається, коли адміністратор переходить до розділу керування категоріями товарів. Система відображає існуючі категорії. Адміністратор може створювати нові категорії, редагувати існуючі (назви, опис, зображення тощо) або видаляти непотрібні категорії.

Прецедент «Керування брендами».

Призначення: даний варіант використання дозволяє адміністратору керувати списком брендів товарів у системі.

Основний потік подій: цей варіант використання запускається, коли адміністратор переходить до розділу керування брендами. Система відображає

список наявних брендів. Адміністратор може додавати нові бренди, редагувати інформацію про існуючі (назва, логотип, опис тощо) або видаляти бренди з системи.

Прецедент «Керування замовленнями».

Призначення: даний варіант використання дозволяє адміністратору переглядати та керувати замовленнями, оформленими клієнтами у системі.

Основний потік подій: цей варіант використання починається, коли адміністратор переходить до розділу керування замовленнями. Система відображає список всіх замовлень з можливістю фільтрації та пошуку. Адміністратор може переглядати деталі замовлення, оновлювати його статус, керувати процесом доставки та виконувати інші дії з обробки замовлень.

2.3 Діаграма діяльності

Діаграма діяльності є одним з видів діаграм в уніфікованій мові моделювання (UML), яка використовується для графічного представлення потоків керування між різними діяльностями або кроками в межах певного процесу чи використання системи. Ця діаграма складається з вузлів, що символізують окремі дії або стани, та переходів між ними, які визначають послідовність виконання цих дій або переходів з одного стану в інший [12].

Діаграма діяльності допомагає візуалізувати та документувати динамічну поведінку системи, зображуючи її як потік керування між різними діями або станами. Вона дозволяє моделювати паралельні або альтернативні потоки, розгалуження та з’єднання, а також цикли та ітерації. Крім того, діаграма діяльності може включати елементи, такі як початкові та кінцеві вузли, розвилки, злиття, вирішувачі та інші спеціальні символи, щоб точно відобразити логіку процесу.

На рисунках 2.4 – 2.5 наведено діаграму діяльності прецеденту «Оформити замовлення».

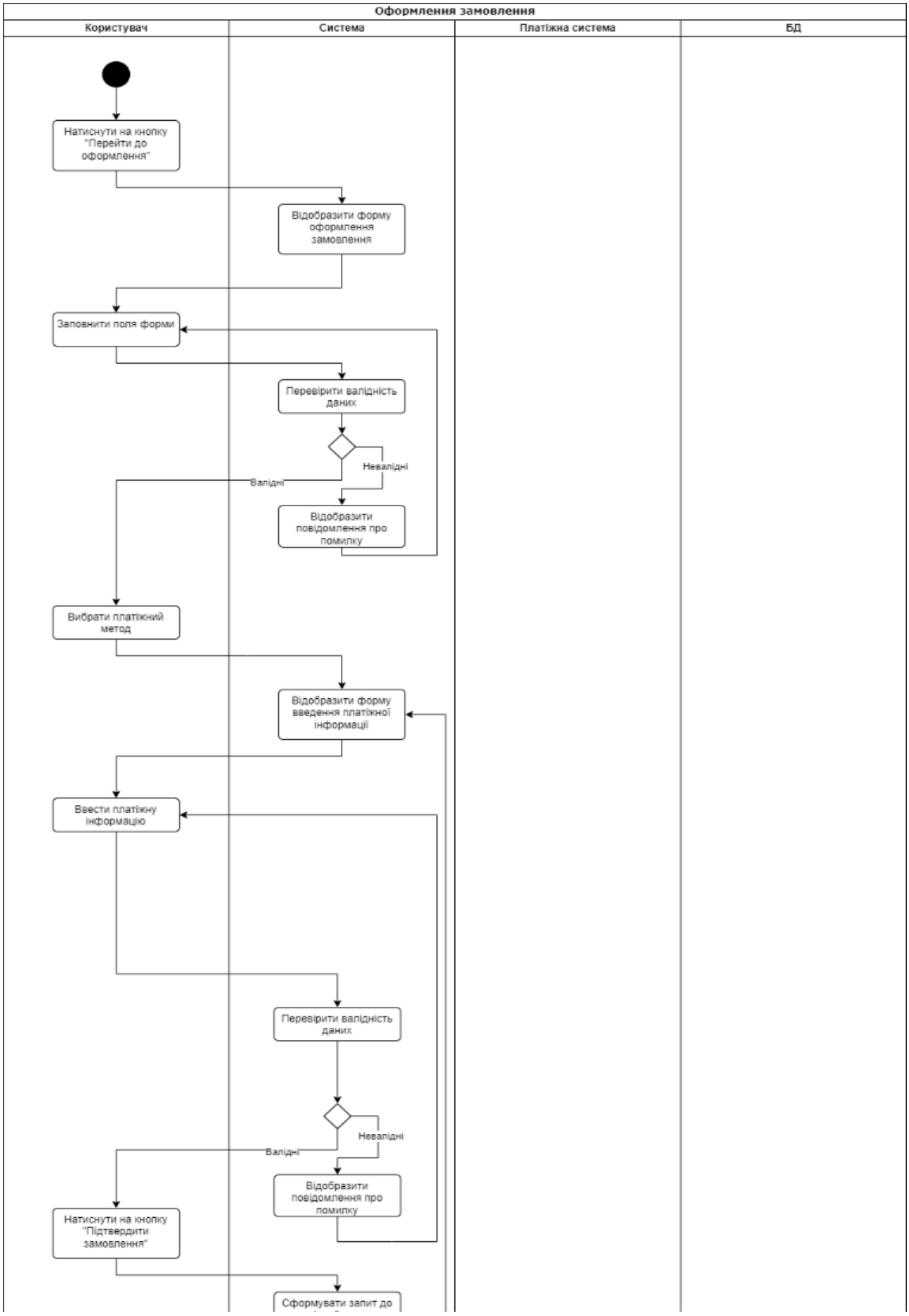

Рисунок 2.4 – ДД «Оформити замовлення» (1 частина)

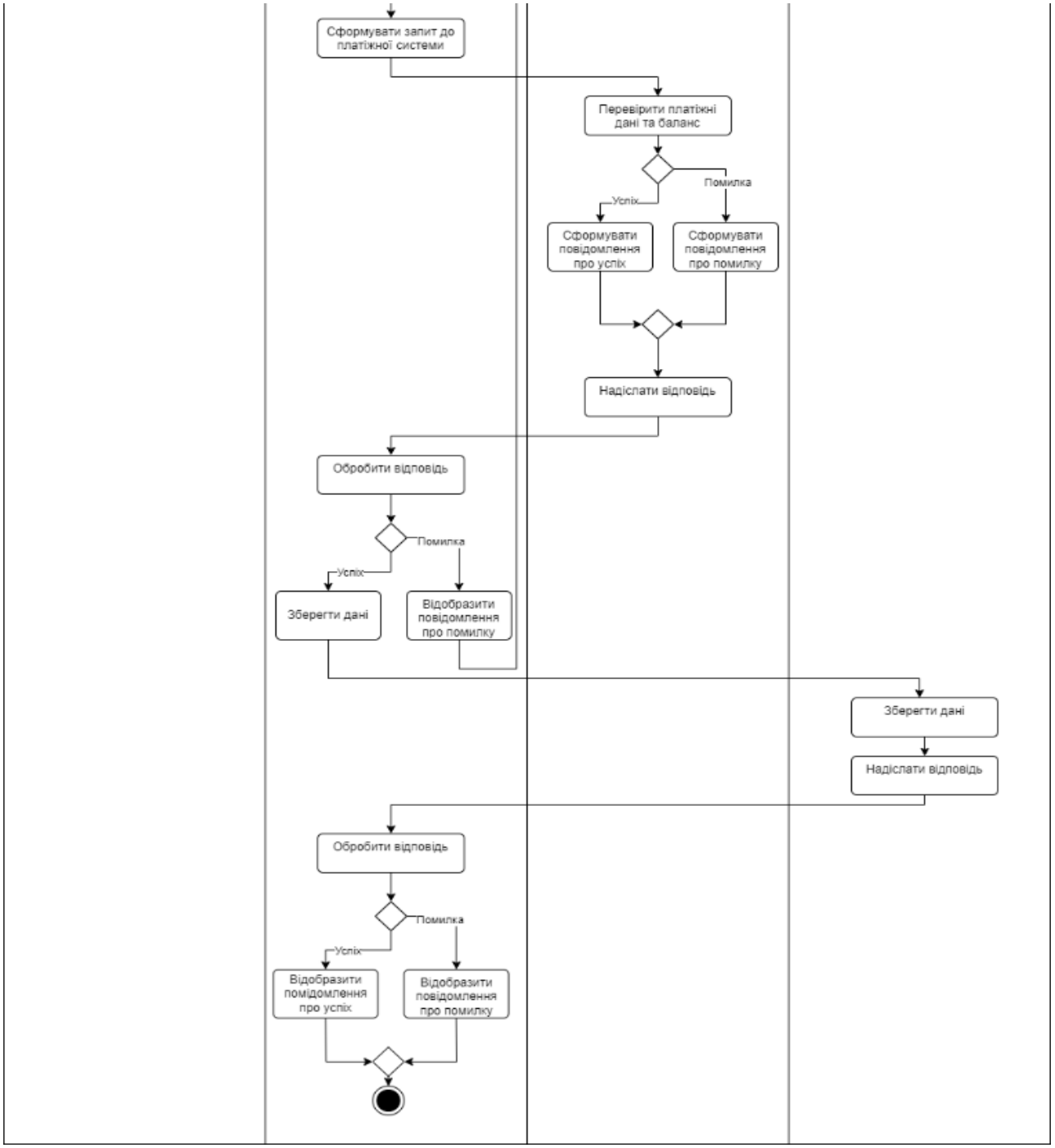

Рисунок 2.5 – ДД «Оформити замовлення» (2 частина)

2.4 Діаграма послідовності

Діаграма послідовності є одним з видів діаграм взаємодії в уніфікованій мові моделювання (UML), яка використовується для графічного представлення динамічної поведінки системи шляхом візуалізації обміну повідомленнями між

об’єктами або компонентами протягом певного проміжку часу. Ця діаграма допомагає зрозуміти, як різні об’єкти взаємодіють між собою, щоб виконати певний сценарій або процес у системі [12].

На рисунку 2.6 описана діаграма послідовності прецеденту «Оформити замовлення».

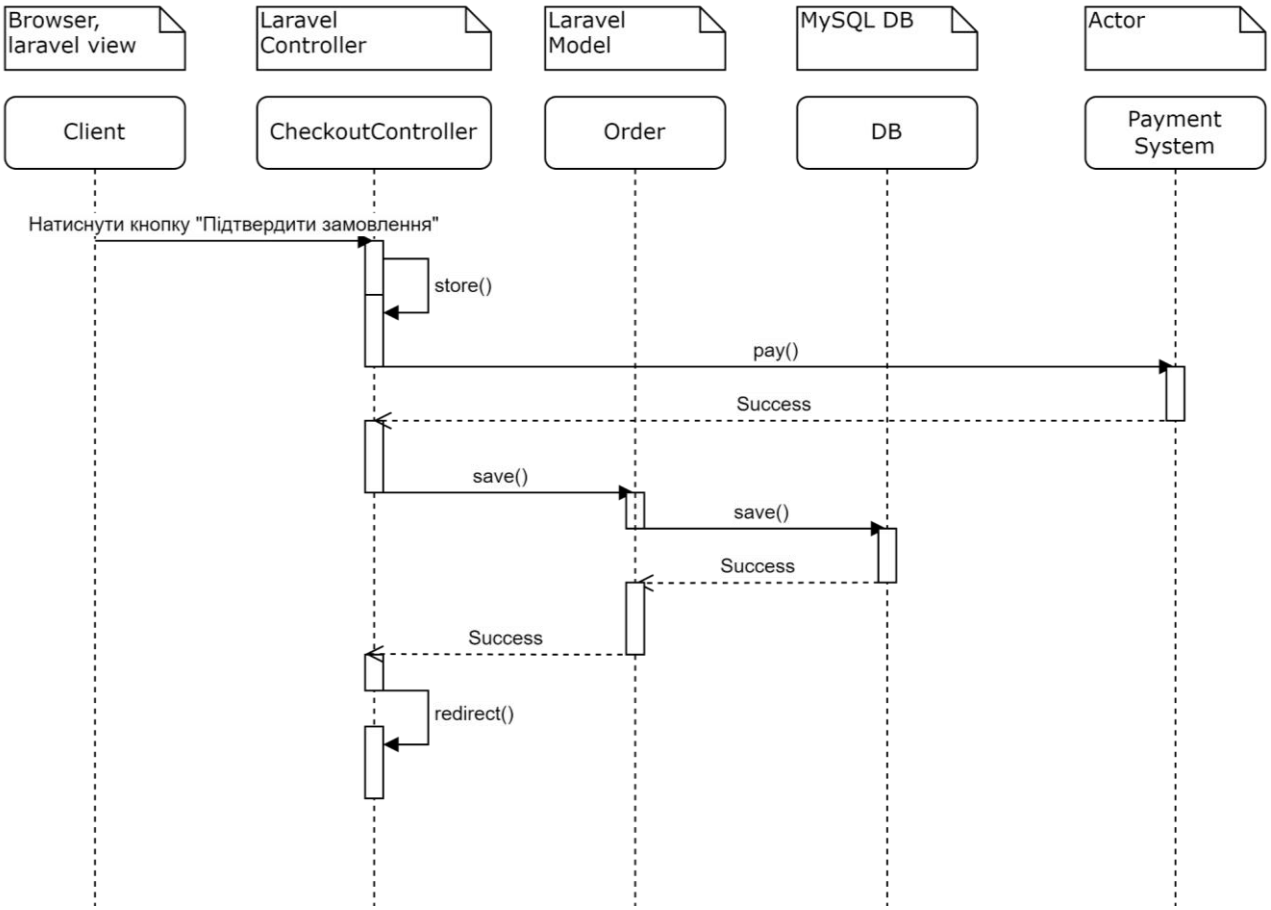

Рисунок 2.6 – Діаграма послідовності

На діаграмі послідовності об’єкти розташовуються горизонтально вздовж верхньої частини діаграми, а їхні лінії життя, які представляють існування об’єктів у часі, відображаються вертикально. Послідовність взаємодії між об’єктами зображується за допомогою стрілок або повідомлень, які проходять між лініями життя в хронологічному порядку зверху вниз. Кожне повідомлення містить назву операції або методу, який викликається одним об’єктом в іншому [12].

Діаграма послідовності дозволяє наочно представити, як об’єкти активуються, коли вони відправляють або отримують повідомлення, а також візуалізувати часову послідовність подій та взаємозв’язки між об’єктами. Вона може включати додаткові елементи, такі як фрагменти, що представляють альтернативні або паралельні потоки виконання, а також примітки та коментарі для додаткового пояснення [12].

Ця діаграма є корисним інструментом для розробників програмного забезпечення, оскільки вона допомагає зрозуміти складні взаємодії між компонентами системи та візуалізувати динамічну поведінку програми. Вона також може використовуватися для документування та передачі знань про систему іншим членам команди, полегшуючи комунікацію та забезпечуючи спільне розуміння поведінки системи.

2.5 Діаграма розгортання

Діаграма розгортання є одним з видів структурних діаграм в уніфікованій мови моделювання (UML), яка використовується для візуального представлення фізичної архітектури системи та її розміщення на апаратних ресурсах. Ця діаграма допомагає зрозуміти, як різні компоненти програмного забезпечення розподілені та взаємодіють між собою на різних вузлах або пристроях у системі [12].

На діаграмі розгортання присутні два основні типи елементів: вузли та артефакти. Вузли представляють фізичні ресурси, такі як сервери, робочі станції, мобільні пристрої або будь-які інші апаратні компоненти, на яких розгортається система. Артефакти, з іншого боку, є екземплярами або реалізаціями програмних компонентів, таких як виконувані файли, бібліотеки або інші програмні модулі, які розміщуються на відповідних вузлах [12].

Взаємозв’язки між вузлами та артефактами зображуються за допомогою ліній зв’язку, які показують, як компоненти програмного забезпечення

розподілені між різними фізичними ресурсами. Діаграма розгортання також може включати додаткові елементи, такі як асоціації, залежності та примітки, що надають додаткову інформацію про особливості розгортання системи.

На рисунку 2.7 наведено діаграму розгортання системи.

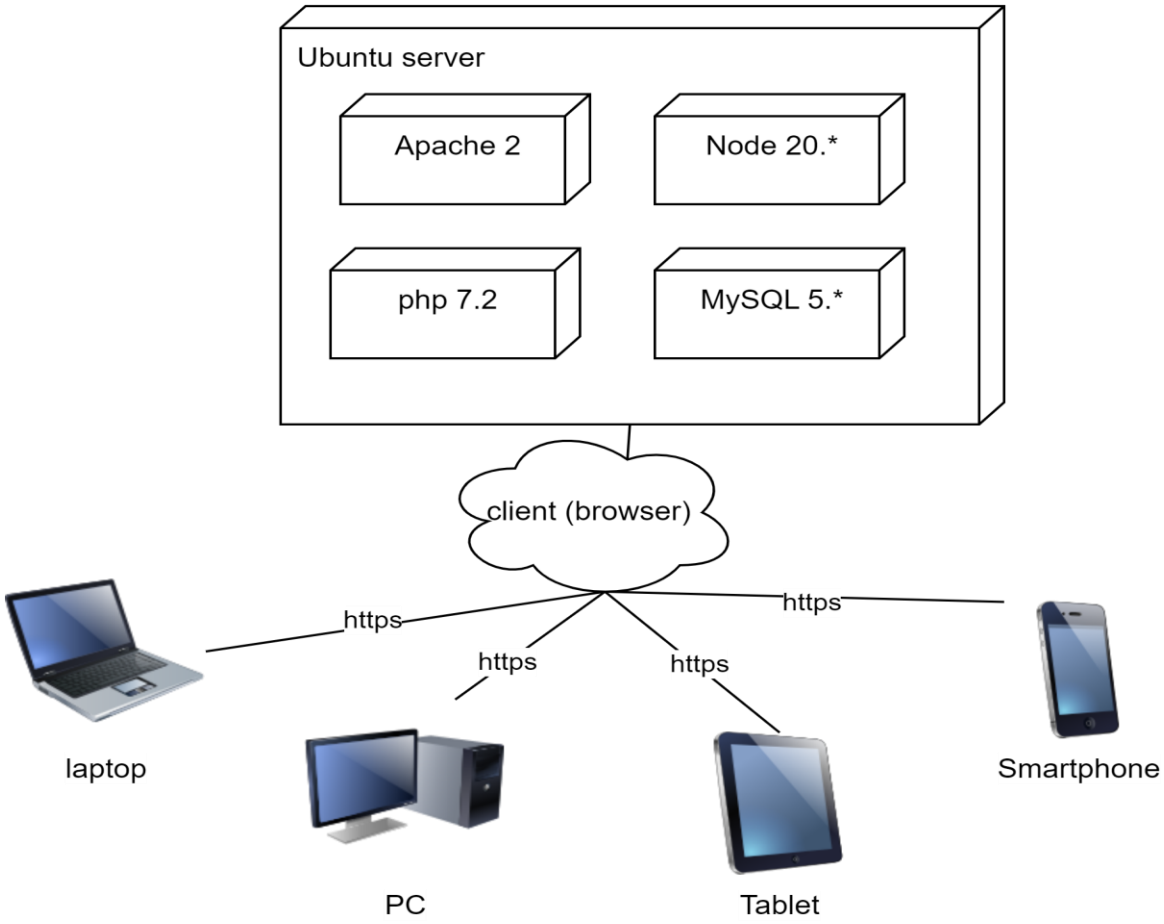

Рисунок 2.7 – Діаграма розгортання

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Опис інструментів розробки

Для реалізації було використано framework Laravel, який має набір відповідних інструментів для створення подібних систем.

jQuery – бібліотека JavaScript, що фокусується на взаємодії JavaScript і HTML.

Popper.js – JavaScript-бібліотека, що призначена для керування позиціонуванням поп-апів, випадаючих списків, тултипів та інших елементів вебінтерфейсу.

3.2 Основні класи системи

Оскільки основним прецедентом системи є «Оформити замовлення», то основними класами системи будуть ті, що надають можливість взаємодіяти з кошиком, сервісами оплати та доставки, а також збереження даних замовлення.

Розглянемо основні класи системи.

OrderController – керує створенням, оновленням та видаленням замовлень. Він також відповідає за відображення інформації про замовлення користувачу.

CheckoutController – відповідає за процес оформлення замовлення, включаючи підтвердження замовлення та обробку платежів.

ProductController – відповідає за управління товарами, які можуть бути додані до кошика та замовлення. Керує створенням, оновленням та видаленням товарів.

CartController – керує процесом додавання товарів до кошика, видаленням їх звідти та оновленням кількості товарів у кошику. Забезпечує

перевірку наявності товарів.

Order – представляє замовлення в базі даних, зберігає інформацію про товари, кількість, загальну вартість та статус замовлення.

Product – представляє товар в базі даних, зберігає інформацію про назву, опис, ціну, кількість та інші атрибути товару.

Cart – зберігає інформацію про товари, додані до кошика користувача, їх кількість та загальну вартість.

OrderItems – зберігає інформацію про окремі товари в замовленні, включаючи їх кількість і ціну.

Зазначені класи мають різні обов'язки, що стосуються різних етапів опрацювання замовлень. Деякі з них відповідають за керування вмістом кошика, інші – за створення самого замовлення, пов'язаних з ним платежів, адрес доставки та рахунків.

Детально ознайомитися з кодом системи можна в додатку В.

3.3 Підготовка та налаштування оточення

Для розгортання системи, спершу необхідно клонувати репозиторій проєкту з GitHub за допомогою команди «git clone *посилання_на_проєкт*», після чого перейти до директорії проєкту командою «*cd назва_проєкту*».

Далі слід встановити всі необхідні залежності за допомогою Composer, виконавши команду «*composer install*» [2]. Наступним кроком є створення файлу .env на основі наявного .env.example за допомогою команди «*cp .env.example .env*». Потім необхідно відредагувати файл .env, вказавши налаштування бази даних та інші параметри (рис. 3.1) [4, 6, 9].

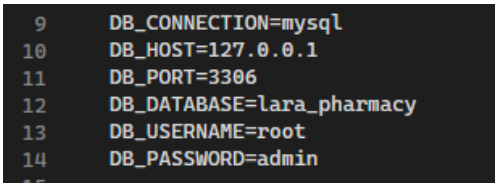

Рисунок 3.1 – Налаштування підключення до БД

Після цього необхідно згенерувати ключ додатка, використовуючи команду «*php artisan key:generate*» [5, 6, 8]. Далі потрібно створити нову базу даних на MySQL сервері з назвою, яку ми вказали у файлі .env (рис. 3.2).

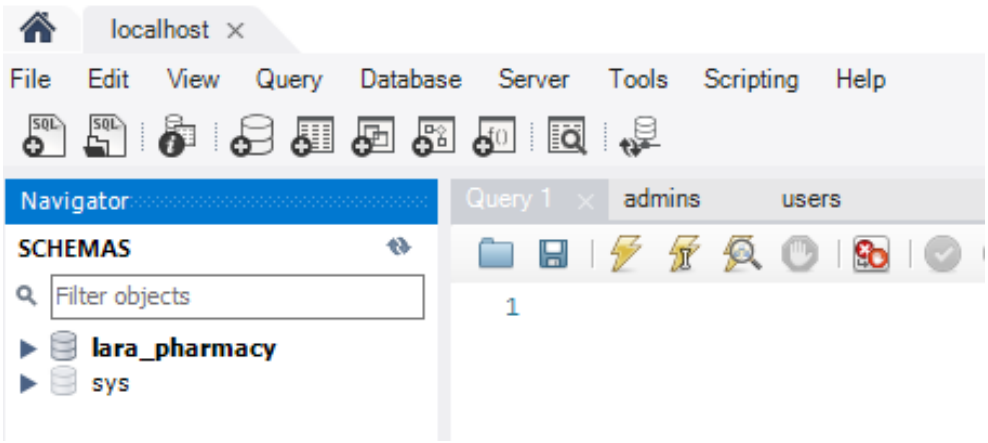

Рисунок 3.2 – Створення БД

Після створення бази даних, запускаємо міграції для створення таблиць у базі даних за допомогою команди «*php artisan migrate*», а потім заповнюємо БД тестовими даними, виконавши команду «*php artisan db:seed*» [6, 8, 9].

Далі необхідно запустити локальний сервер розробки командою «*php artisan serve*» і відкрити браузер, перейшовши за адресою http://127.0.0.1:8000 (рис. 3.3).

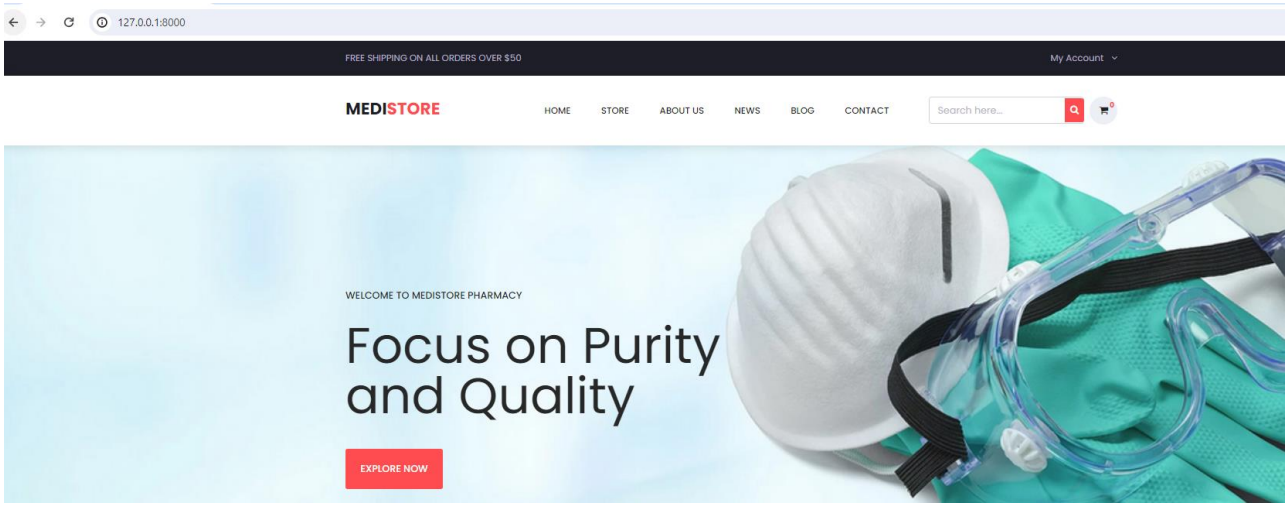

Рисунок 3.3 – Запуск проєкту

Потім встановлюємо NPM-залежності, перейшовши до директорії проєкту та виконавши команду «*npm install*». Після цього компілюємо assets за допомогою Laravel Mix командою «*npm run dev*».

У результаті, після виконання всіх вищевказаних дій, наш локальний сервер налаштований і готовий для роботи з проєктом.

З файлом оточення можна ознайомитися в додатку А.

3.4 Тестування проєкту

Unit-тестування в Laravel – це процес написання і виконання тестів, які перевіряють окремі частини коду вашого додатка, такі як функції, методи або класи. Мета unit-тестів полягає в тому, щоб ізолювати кожен компонент системи і підтвердити, що він працює відповідно до очікувань. Laravel використовує PHPUnit, потужний фреймворк для тестування на PHP, що дозволяє розробникам легко створювати, запускати і підтримувати тести [2, 3].

На рисунку 3.4 наведено приклад unit-тесту для функції створення замовлення класу CheckoutController.

```php
class CheckoutControllerTest extends TestCase
{

    public function it_creates_an_order_and_associates_carts()
    {
        $user = factory(User::class)->create();

        $cart1 = factory(Cart::class)->create(['user_id' => $user->id]);
        $cart2 = factory(Cart::class)->create(['user_id' => $user->id]);

        $orderData = [
            'name' => 'Test User',
            'phone_no' => '1234567890',
            'email' => 'test@example.com',
            'message' => 'Test message',
            'shipping_address' => '123 Test St',
        ];

        Auth::shouldReceive('check')->andReturn(true);
        Auth::shouldReceive('id')->andReturn($user->id);

        $response = $this->actingAs($user)->post('/checkout', $orderData);
```

Рисунок 3.4 – Unit-тестування CheckoutController

У Laravel unit-тести зазвичай розміщуються в директорії tests/Unit. Тести можна створювати за допомогою команди Artisan, що генерує необхідні файли і структуру для тестування. Тести виконуються в ізольованому середовищі, де база даних імітується або використовується тестова база даних, щоб забезпечити чистий стан для кожного тесту. Це дозволяє виявляти і виправляти помилки на ранніх стадіях розробки, покращуючи якість коду і забезпечуючи стабільність додатка [4, 5].

Детально ознайомитися з тестами можна в додатку Б.

3.5 Керівництво користувача

3.5.1 Реєстрація

Щоб зареєструватися на сайті, перейдіть на головну сторінку та натисніть кнопку «Register» у верхньому правому куті. Заповніть форму, ввівши своє ім’я, адресу електронної пошти, пароль, підтвердження паролю, адресу та контактні дані (рис. 3.5).

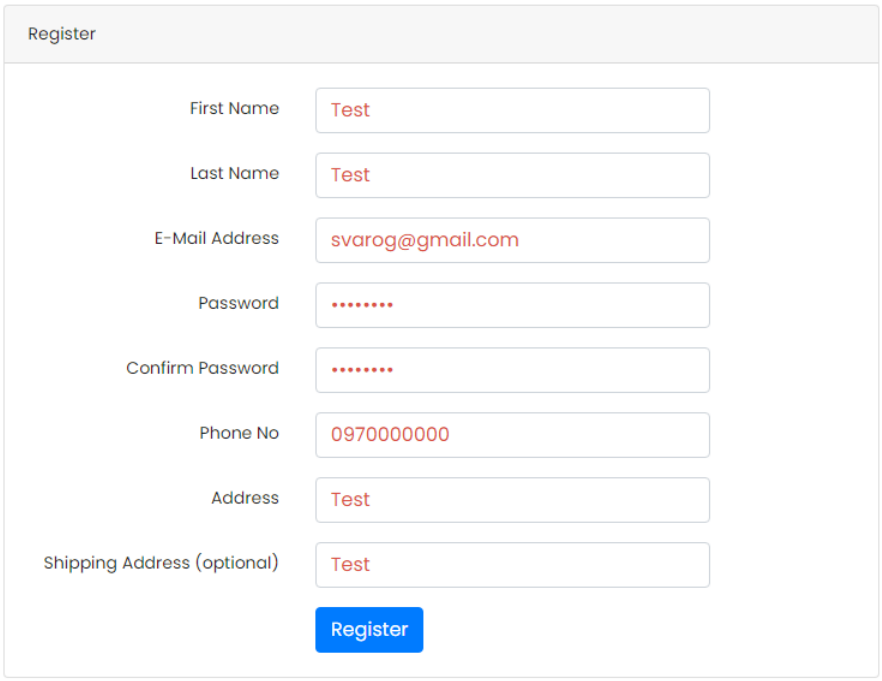

Рисунок 3.5 – Форма реєстрації

Натисніть кнопку «Register». Після успішної реєстрації ви будете перенаправлені на головну сторінку (рис. 3.6).

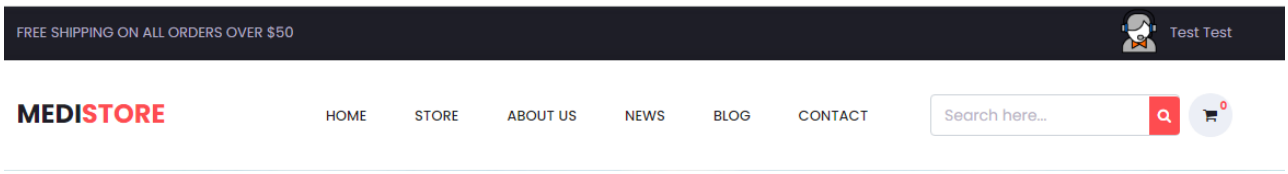

Рисунок 3.6 – Головна сторінка

Якщо, під час реєстрації, введено невалідні дані, то система відобразить повідомлення, відповідно до помилки (рис. 3.7).

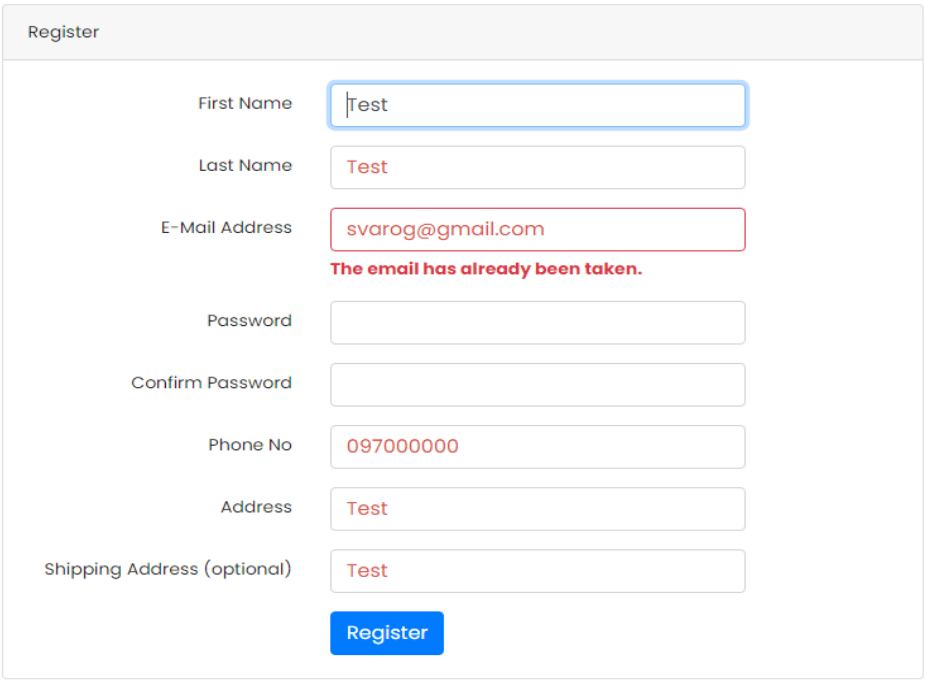

Рисунок 3.7 – Невалідні дані

3.5.2 Вхід

Щоб увійти на сайт, натисніть кнопку «Login» у верхньому правому куті. Введіть свою адресу електронної пошти та пароль, які ви вказали під час реєстрації (рис. 3.8).

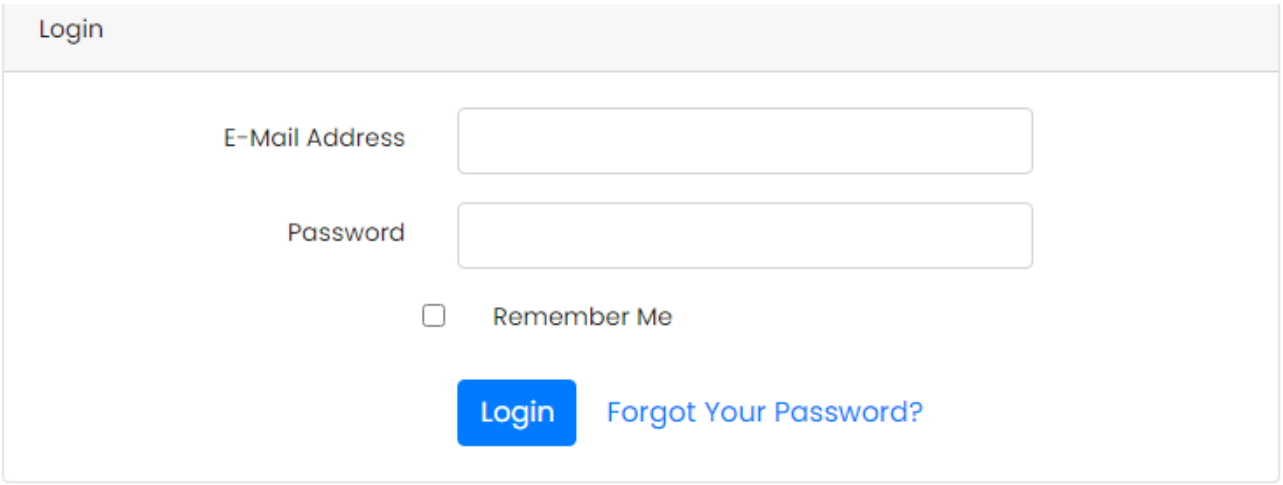

Рисунок 3.8 – Форма входу

Натисніть кнопку «Login». Якщо дані введені правильно, ви будете перенаправлені на головну сторінку, якщо ні – система відобразить повідомлення, відповідно до помилки (рис. 3.9).

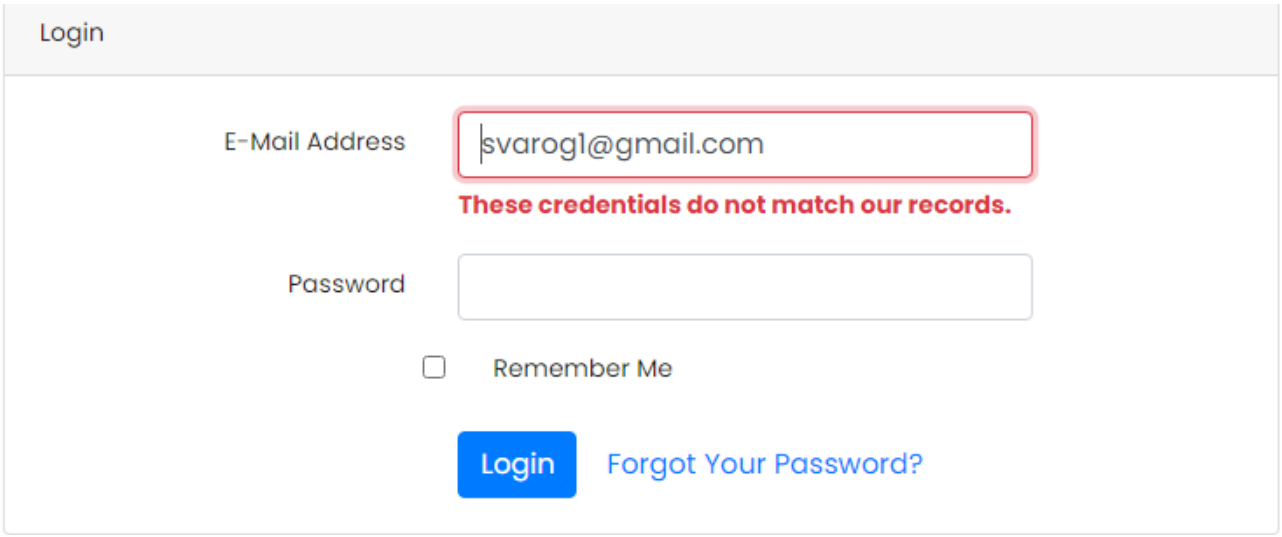

Рисунок 3.9 – Помилка входу

3.5.3 Вибір товару

Для вибору товару перейдіть до каталогу товарів, натиснувши на відповідний розділ у меню навігації (рис. 3.10).

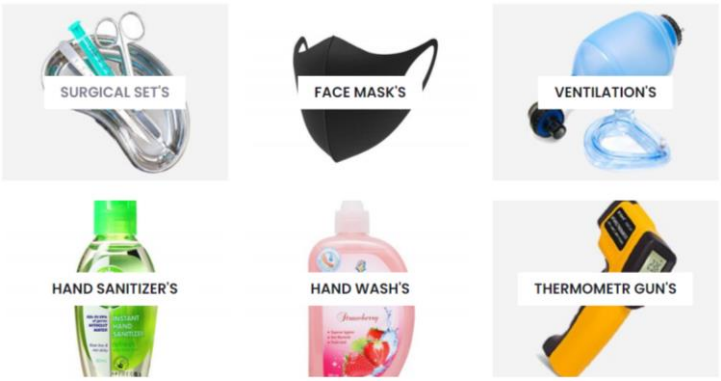

Рисунок 3.10 – Меню

Ви можете скористатися фільтрами для пошуку товарів за категоріями, ціною або брендом (рис. 3.11 – 3.12).

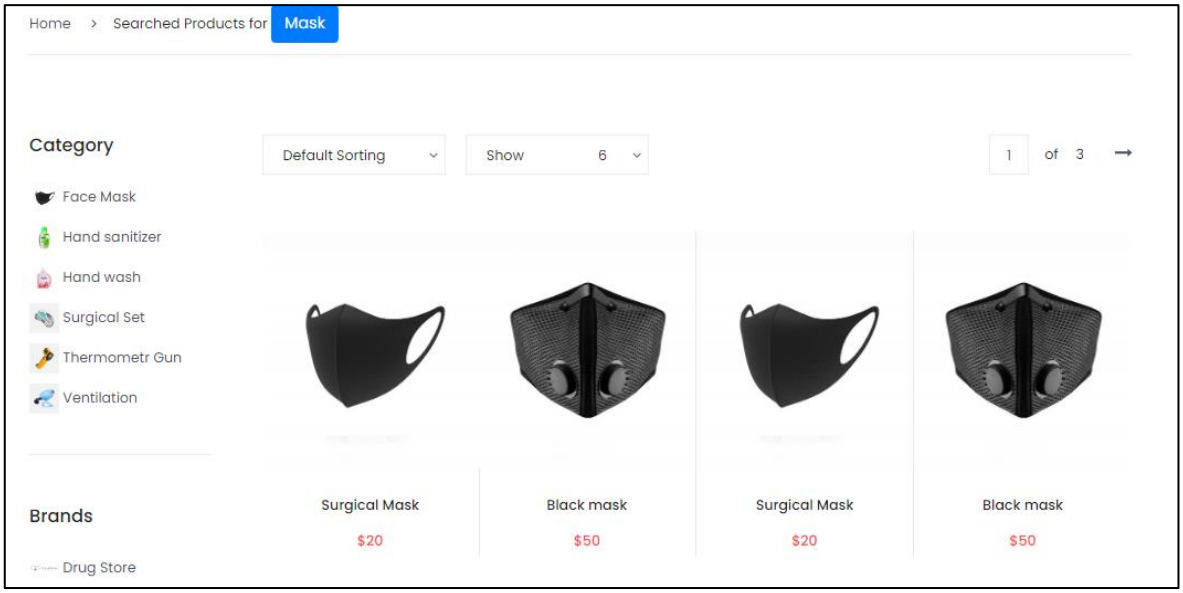

Рисунок 3.11 – Фільтр за категорією «Face Mask»

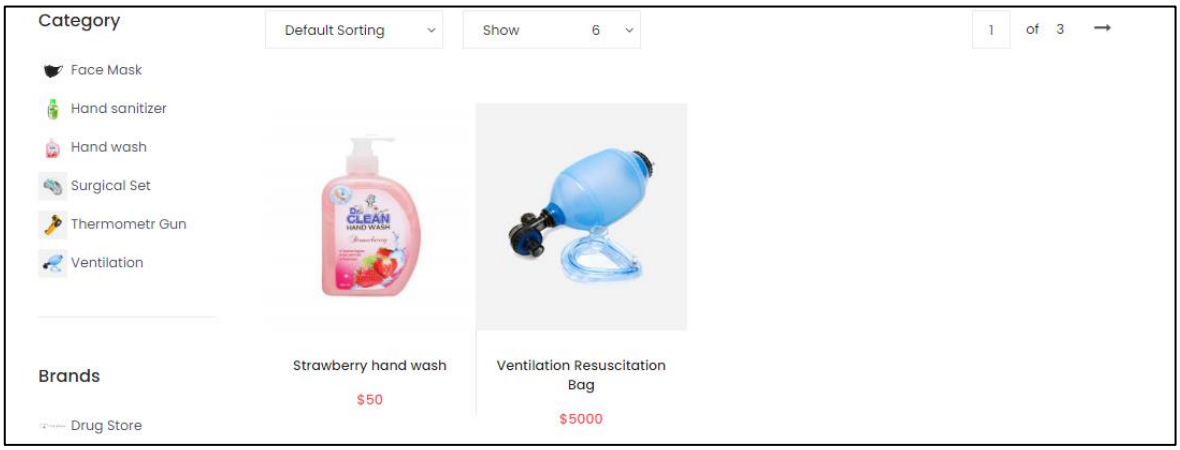

Рисунок 3.12 – Фільтр за брендом «Drug Store»

Коли знайдете потрібний товар, натисніть на нього, щоб переглянути деталі. Додайте товар до кошика, натиснувши кнопку «Add to Cart». За потреби можна обрати кількість та колір товару (рис. 3.13).

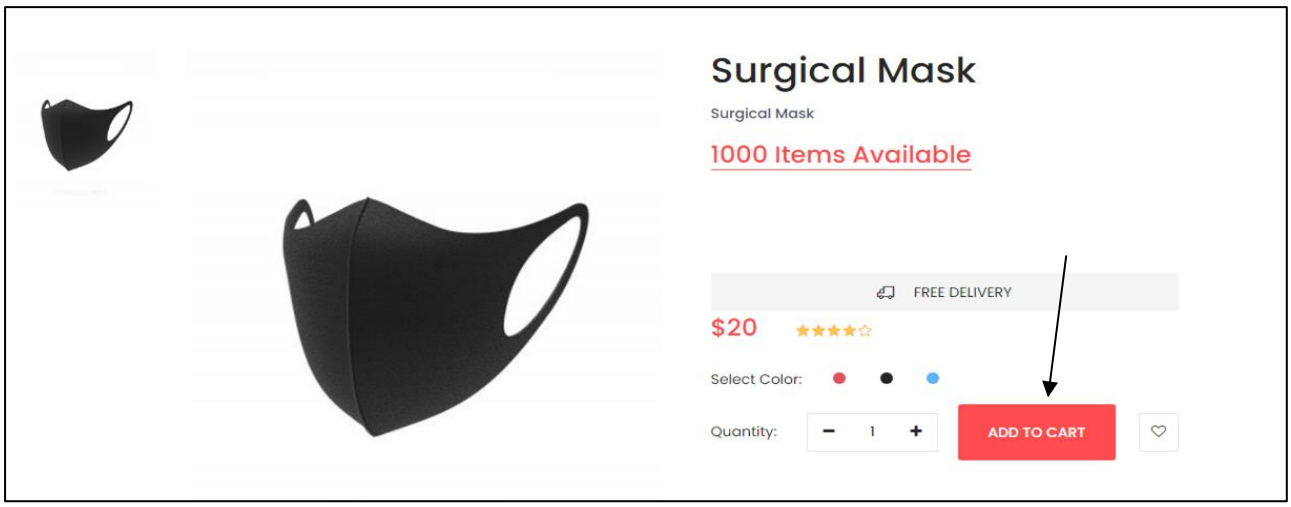

Рисунок 3.13 – Картка товару

Після додавання товару до кошика, система відобразить повідомлення про успішну дію (рис. 3.14).

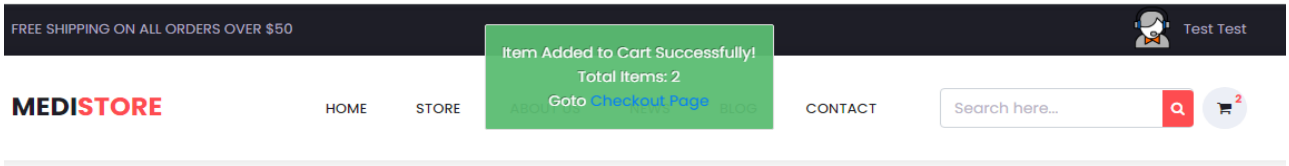

Рисунок 3.14 – Повідомлення про додавання товару до кошика

3.5.4 Оформлення замовлення

Після вибору товарів натисніть на піктограму кошика у верхньому правому куті сторінки (рис. 3.15).

Перегляньте список вибраних товарів і натисніть кнопку «Checkout» (рис. 3.16).

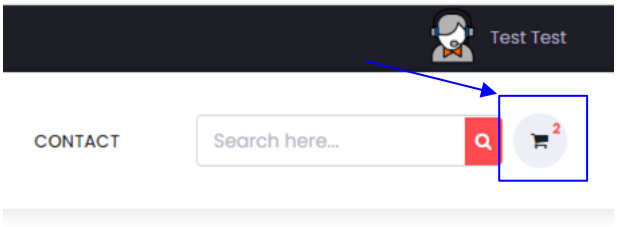

Рисунок 3.15 – Піктограма кошика

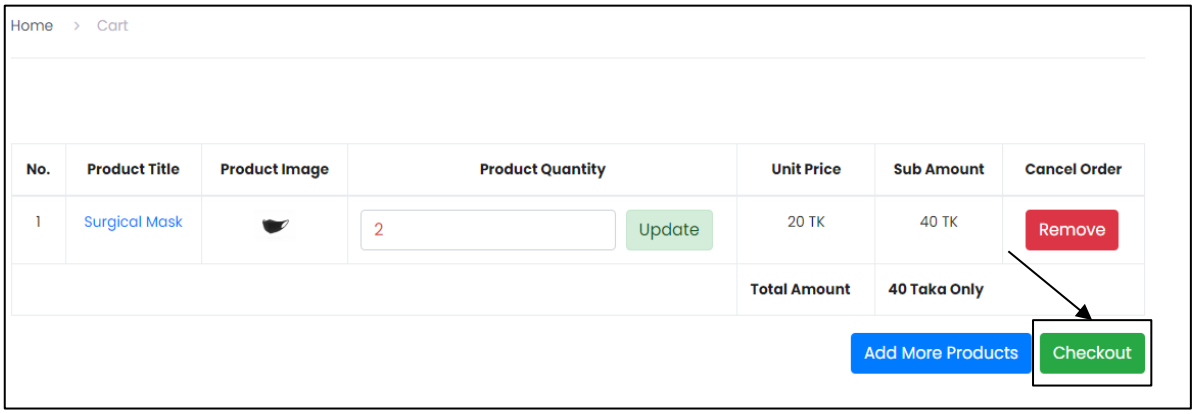

Рисунок 3.16 – Інтерфейс кошика

Заповніть форму з інформацією про доставку, включаючи ім’я, номер телефону, адресу доставки та інші необхідні дані. Підтвердіть замовлення, натиснувши кнопку «Order Now» (рис. 3.17).

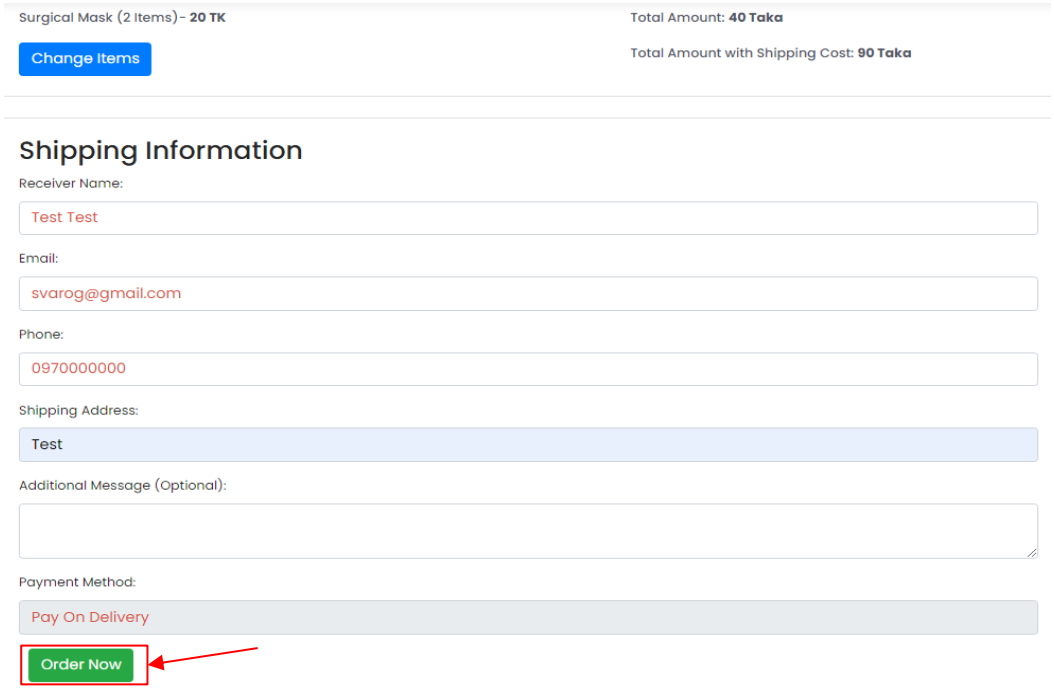

Рисунок 3.17 – Форма оформлення замовлення

3.5.5 Вхід до панелі адміністратора

Щоб увійти до панелі адміністратора, перейдіть на сторінку входу адміністратора (наприклад, yourwebsite.com/admin). Система відобразить форму входу до панелі адміністратора. Важливою умовою є вихід з профіля користувача, перед входом до адмін-панелі (рис. 3.18).

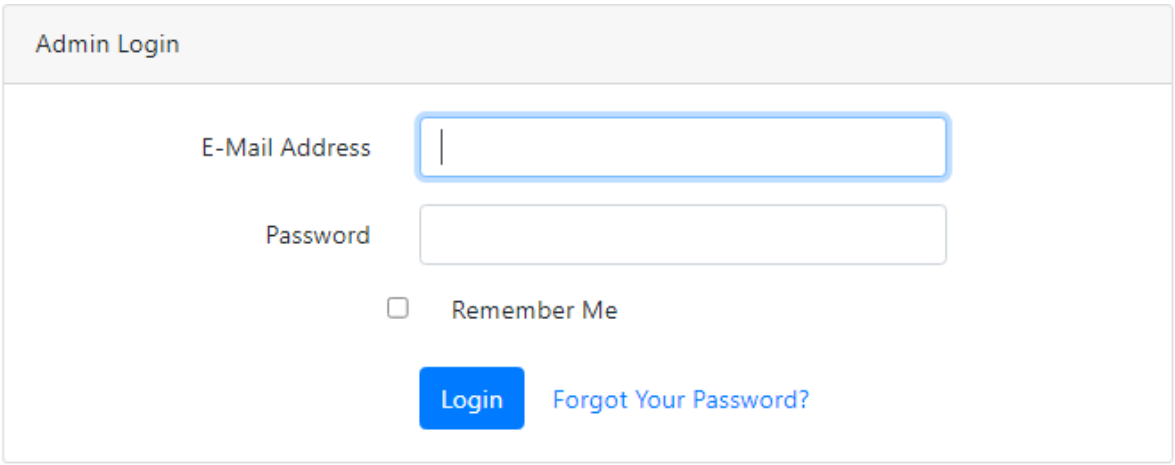

Рисунок 3.18 – Форма входу адміністратора

Введіть адміністративний логін і пароль, які вам були надані. Натисніть кнопку «Login». Якщо дані введені правильно, ви будете перенаправлені до панелі адміністратора (рис. 3.19).

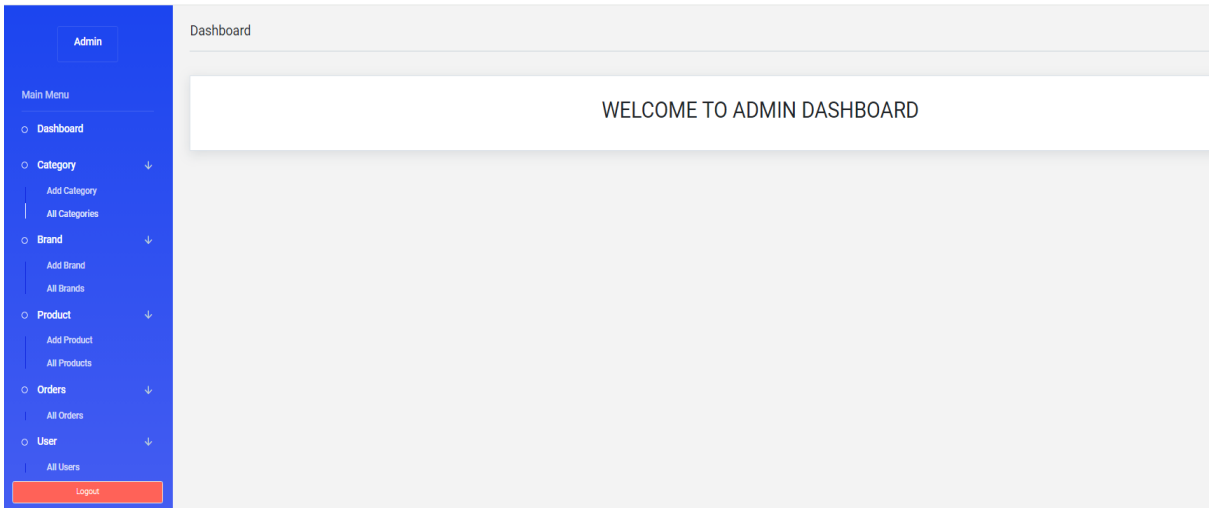

Рисунок 3.19 – Панель адміністратора

3.5.6 Панель адміністратора

У панелі адміністратора ви маєте доступ до наступних розділів:

– керування товарами (додавання, редагування та видалення товарів);
– керування категоріями (додавання, редагування та видалення категорій товарів);
– замовлення (перегляд і обробка замовлень клієнтів);
– користувачі (керування зареєстрованими користувачами);
– бренди (додавання, редагування та видалення брендів).

Перейдіть до потрібного розділу, використовуючи меню навігації панелі адміністратора, щоб виконувати необхідні дії для керування інтернет-магазином (рис. 3.20 – 3.21).

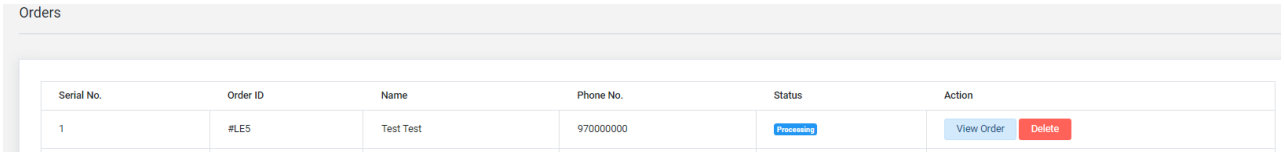

Рисунок 3.20 – Розділ «Orders»

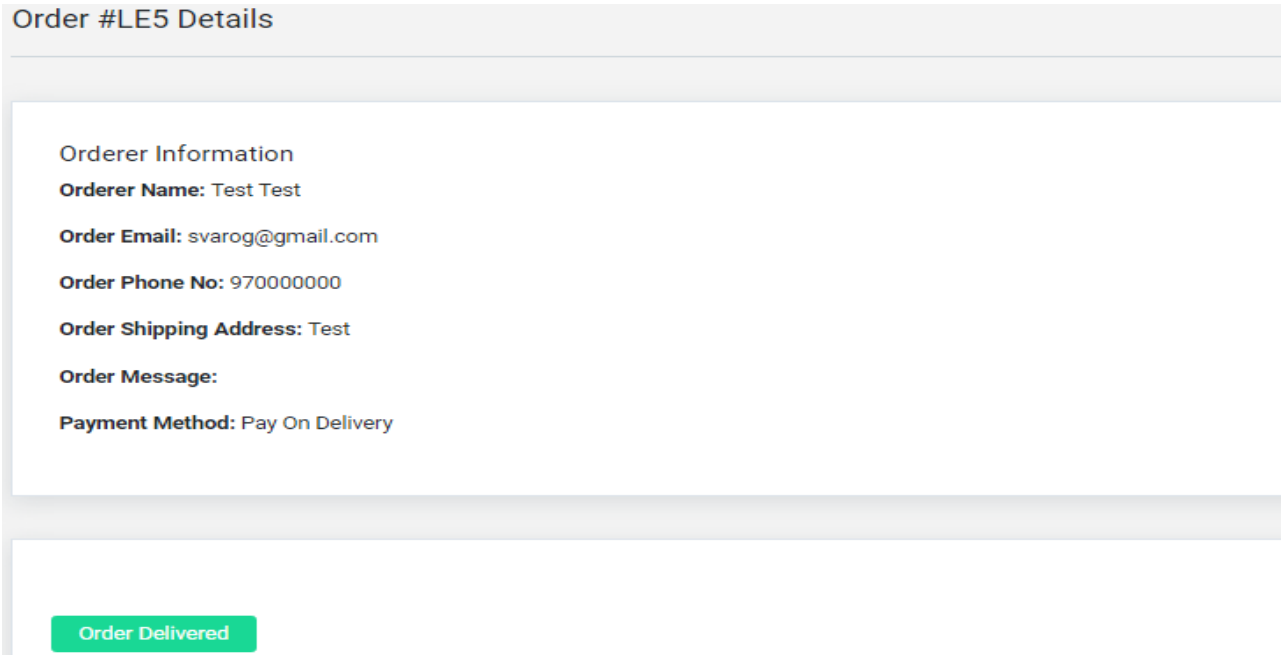

Рисунок 3.21 – Деталі замовлення

ВИСНОВКИ

У ході виконаної роботи було створено та протестовано інтернет-магазин медичних товарів на базі фреймворку Laravel. Використання UML для моделювання процесів та структурних компонентів дозволило забезпечити високу якість та надійність розробки. Система успішно виконує функції обробки кошика, оформлення замовлень, керування товарами та користувачами. Проведене unit-тестування підтвердило правильність роботи основних компонентів системи.

У відповідності з метою кваліфікаційної роботи було розроблено інтернет-магазин медичних товарів з використаннями наступних технологій:

- Laravel для реалізації backend і frontend;
- MySQL для зберігання даних.

У відповідності з поставленими задачами були виконані наступні етапи створення додатку:

- сформовані функціональні та нефункціональні вимоги до системи, а також оглянуто та порівняно аналоги;
- спроєктована та побудована структура системи (побудовані діаграми прецедентів, діяльності, послідовності та розгортання; надано опис кожного з варіантів використання);
- реалізовано інтернет-магазин медичних товарів (наведена інструкція по налаштуванню та розгортанню оточення, надано керівництво користувача);
- протестована робота системи.

ПЕРЕЛІК ПОСИЛАНЬ

1. Apteka24.ua. URL: https://www.apteka24.ua/ (дата звернення: 23.02.2024).
2. Correa D., Vallejo P. Practical Laravel: Develop clean MVC web applications. Michigan : Independently published, 2022. 177 p.
3. Daubois A., Windler C. Clean Code in PHP: Expert tips and best practices to write beautiful, human-friendly, and maintainable PHP. Birmingham : Packt Publishing, 2022. 264 p.
4. Delcione Lopes da Silva. Framework PHP Laravel 8 & AJAX. CBL, 2022. 445 p.
5. Dwivedi R. Laravel Essentials: A Step-by-Step Guide for Web Developers: “Laravel by Example: Building Real-World Applications”. Michigan : Independently published, 2023. 269 p.
6. Laravel 8 Developer Documentation. URL: https://laravel.com/docs/8.x/ (дата звернення: 15.03.2024).
7. Med-Magazin.ua. URL: https://med-magazin.ua/ (дата звернення: 23.02.2024).
8. Scholtens A. Mastering Laravel. Cary : Sas155, 2023. 102 p.
9. Stauffer M. Laravel: Up & Running: A Framework for Building Modern PHP Apps. Sebastopol : O’Reilly Media, 2019. 544 p.
10. Tabletki.ua. URL: https://tabletki.ua/ (дата звернення: 23.02.2024).
11. TATL 1mg. URL: https://www.1mg.com/ (дата звернення: 23.02.2024).
12. Unhelkar B. Software Engineering with UML. Boca Raton : Auerbach Publications, 2020. 426 p.
13. Фармекс Груп. URL: https://www.pharmex.com.ua/ (дата звернення: 23.02.2024).

ДОДАТОК А

Файл оточення

```
APP_NAME=Laravel
APP_ENV=local
APP_KEY=base64:9IEA8xhO4kf3ToOhWUPkeDjaf5bHuilgGMs8pMJmghE=
APP_DEBUG=true
APP_URL=http://localhost

LOG_CHANNEL=stack

DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=lara_pharmacy
DB_USERNAME=root
DB_PASSWORD=admin

BROADCAST_DRIVER=log
CACHE_DRIVER=file
QUEUE_CONNECTION=sync
SESSION_DRIVER=cookie
SESSION_LIFETIME=120

REDIS_HOST=127.0.0.1
REDIS_PASSWORD=null
REDIS_PORT=6379

MAIL_DRIVER=smtp
MAIL_HOST=smtp.mailtrap.io
MAIL_PORT=2525
MAIL_USERNAME=null
MAIL_PASSWORD=null
MAIL_ENCRYPTION=null
MAIL_FROM_ADDRESS=null
MAIL_FROM_NAME="${APP_NAME}"
```

```
AWS_ACCESS_KEY_ID=
AWS_SECRET_ACCESS_KEY=
AWS_DEFAULT_REGION=us-east-1
AWS_BUCKET=

PUSHER_APP_ID=
PUSHER_APP_KEY=
PUSHER_APP_SECRET=
PUSHER_APP_CLUSTER=mt1

MIX_PUSHER_APP_KEY="${PUSHER_APP_KEY}"
MIX_PUSHER_APP_CLUSTER="${PUSHER_APP_CLUSTER}"
```

ДОДАТОК Б

Тести

```
<?php

namespace Tests\Unit;

use Tests\TestCase;
use Illuminate\Foundation\Testing\RefreshDatabase;
use App\Order;
use App\Cart;
use App\User;
use Illuminate\Support\Facades\Auth;

class CheckoutControllerTest extends TestCase
{

    public function it_creates_an_order_and_associates_carts()
    {
        $user = factory(User::class)->create();

        $cart1 = factory(Cart::class)->create(['user_id' => $user->id]);
        $cart2 = factory(Cart::class)->create(['user_id' => $user->id]);

        $orderData = [
            'name' => 'Test User',
            'phone_no' => '1234567890',
            'email' => 'test@example.com',
            'message' => 'Test message',
            'shipping_address' => '123 Test St',
        ];

        Auth::shouldReceive('check')->andReturn(true);
        Auth::shouldReceive('id')->andReturn($user->id);

        $response = $this->actingAs($user)->post('/checkout', $orderData);
```

```
        $response->assertStatus(302);

        $this->assertDatabaseHas('orders', [
            'name' => 'Test User',
            'phone_no' => '1234567890',
            'email' => 'test@example.com',
            'message' => 'Test message',
            'shipping_address' => '123 Test St',
            'user_id' => $user->id,
        ]);

        $order = Order::where('user_id', $user->id)->first();

        $this->assertDatabaseHas('carts', [
            'id' => $cart1->id,
            'order_id' => $order->id,
        ]);

        $this->assertDatabaseHas('carts', [
            'id' => $cart2->id,
            'order_id' => $order->id,
        ]);
    }
}

<?php

namespace Tests\Unit;

use Tests\TestCase;
use App\Order;
use App\User;

class OrderControllerTest extends TestCase
{

    /** @test */
    public function it_creates_an_order()
```

```
    {
        // Створення тестового користувача
        $user = User::first();

        // Дані для створення замовлення
        $orderData = [
            "id" => 4,
            "user_id"=> 1,
            "ip_address"=> null,
            "name"=> "Afia Raihana",
            "phone_no"=> 1777112233,
            "shipping_address"=> "Cumilla",
            "email"=> "afia@gmail.com",
            "message"=> "Extra box with wrap.",
            "processing"=> 1,
            "paid"=> 0,
            "delivered"=> 1,
            "created_at"=> "2020-07-26 13:40:06",
            "updated_at"=> "2020-07-26 13:55:20"
        ];

        // Відправка POST-запиту до методу створення замовлення
        $response = $this->actingAs($user)->post('/orders', $orderData);

        // Перевірка, що замовлення створене в базі даних
        $this->assertDatabaseHas('orders', $orderData);

        // Перевірка статусу відповіді
        $response->assertStatus(201);
    }
}
```

ДОДАТОК В

Посилання на Git

https://bitbucket.org/s_var_og/larapharm/src/master/