

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Інженерія програмного забезпечення»)

на тему:

«Інтернет-магазин мобільних телефонів та аксесуарів»

Виконав студент групи ІПЗ-19-2
ДОВГАЛЮК Ілля Сергійович

Керівник роботи:
ВЛАСЕНКО Олег Васильович

Рецензент:
ЛОБАНЧИКОВА Надія Миколаївна

Житомир – 2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

В.о. зав. кафедри інженерії програмного
забезпечення

Андрій МОРОЗОВ

«14» лютого 2023 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ДОВГАЛЮК Ілля Сергійович**

Керівник роботи: **ВЛАСЕНКО Олег Васильович**

Тема роботи: «Інтернет-магазин мобільних телефонів та аксесуарів»,

затверджена Наказом університету від «14» лютого 2023 р. № 70/с

Вихідні дані роботи: Об'єктом дослідження є методи та засоби, необхідні для продажу мобільних телефонів та аксесуарів. Предметом дослідження є процес розробки повноцінної вебсистеми для реалізації електронних товарів з використанням сучасних програмних технологій, таких як React, Laravel, MySQL

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
1	Власенко О.В., ст. викладач кафедри ІІЗ	22.02.2023р.	05.03.2023р.
2	Власенко О.В., ст. викладач кафедри ІІЗ	06.03.2023р.	17.05.2023р.
3	Власенко О.В., ст. викладач кафедри ІІЗ	18.05.2023р.	28.05.2023р.

Керівник _____ Олег ВЛАСЕНКО

Календарний план

№ з/п	Назва етапів випускної кваліфікаційної роботи	Термін виконання етапів роботи	Примітка
1	Постановка задачі	5 лютого 2023 – 14 лютого 2023	Виконано
2	Аналіз сайтів конкурентів	15 лютого 2023 - 20 лютого 2023	Виконано
3	Формулювання технічного завдання	20 лютого 2023 - 28 лютого 2023	Виконано
4	Опрацювання теоретичних матеріалів	1 березня 2023 - 15 березня 2023	Виконано
5	Проектування структури	16 березня 2023 - 31 березня 2023	Виконано
6	Розробка бази даних та написання програмного коду	1 квітня 2023 - 15 травня 2023	Виконано
7	Тестування	16 травня 2023 - 20 травня 2023	Виконано
8	Написання пояснювальної Записки	21 травня 2023 – 28 травня 2023	Виконано
9	Попередній захист	29 травня 2023	Виконано

РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу інтернет-магазину мобільних телефонів та аксесуарів та пояснювальної записки. Текстова частина викладена на 53 сторінках друкованого тексту. Пояснювальна записка має 15 сторінок додатків. Список використаних джерел містить 11 пунктів та займає 2 сторінки. В роботі наведено 41 малюнок. Загальний обсяг роботи 70 сторінок.

Метою роботи є розробка інтернет-магазину мобільних телефонів та аксесуарів для подальшого його використання.

В розділах пояснювальної записки було проаналізовано аналогічні продукти, розглянути методи та засоби для реалізації проєкту та вибрано найкращі технології для реалізації вебсайту. Також було побудовано схему варіантів використання веб-додатку. Крім того, було проведено тестування вебсайту та продемонстрованого його функціонал.

КЛЮЧОВІ СЛОВА: ВЕБСАЙТ, ІНТЕРНЕТ-МАГАЗИН, БАЗА ДАНИХ, REACT, LARAVEL.

					ІПЗ.КР.Б-121-23-ІПЗ			
Змн.	Арк.	№ докум.	Підпис	Дата	Інтернет-магазин мобільних телефонів та аксесуарів	Літ.	Арк.	Аркушів
Розроб.		І. Довгалоук					4	70
Керівник		О. Власенко				Житомирська політехніка, група ІПЗ-19-2		
Н. контр.		О. Тичина						
Зав. каф.		А. Морозов						
					Пояснювальна записка			

ABSTRACT

The bachelor's thesis consists of a software complex of an online store of mobile phones and accessories and an explanatory note. The text part is laid out on 53 pages of printed text. The explanatory note has 15 pages of annexes. The list of used sources contains 11 items and occupies 2 pages. There are 41 drawings in the work. The total volume of work is 70 pages.

The purpose of the work is the development of an online store of mobile phones and accessories for its further development.

In the sections of the explanatory note, similar products were analyzed, methods and means for the implementation of the project were considered, and the best technologies were selected for the implementation of the website. A web application use case diagram was also built. In addition, the website was tested and its functionality was demonstrated.

KEYWORDS: WEBSITE, WEB APPLICATION, ONLINE STORE, DATABASE, REACT, LARAVEL.

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Апр.
						5

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБСАЙТУ	10
1.1 Постановка задачі	10
1.2 Аналіз схожих вебсайтів	11
1.3 Вибір архітектури програмного продукту	16
1.4 Обґрунтування вибору засобів та вимог до програмного забезпечення.....	16
ВИСНОВКИ ДО РОЗДІЛУ 1:.....	18
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ.....	19
2.1 Визначення варіантів використання та структури системи	19
2.2 Розробка бази даних вебсайту	23
2.3 Проєктування та реалізація алгоритмів роботи вебсайту.....	26
2.4 Реалізація функціонала інтернет-магазину мобільних телефонів та аксесуарів.....	31
ВИСНОВКИ ДО РОЗДІЛУ 2:.....	38
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ	40
3.1 Порядок встановлення та налаштування веб-додатку.	40
3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з вебсайтом	40
ВИСНОВКИ ДО РОЗДІЛУ 3:.....	54
ВИСНОВКИ	55
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	56
ДОДАТКИ.....	57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних;

ПЗ – Програмне забезпечення;

ОС – Операційна система;

Backend – серверна частина;

Frontend – клієнтська частина.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						7

ВСТУП

Сучасний розвиток технологій та поширення Інтернету створюють нові можливості для покупців та підприємців у сфері торгівлі. Зі свого досвіду можемо сказати, що одним з найбільш успішних напрямків електронної комерції є інтернет-магазини. Ці віртуальні платформи надають зручність та широкий вибір продуктів, що привертає увагу споживачів та сприяє зростанню обсягів онлайн-торгівлі.

Один із найбільш популярних сегментів товарів у сфері електронної комерції - мобільні телефони та аксесуари до них. Швидкі темпи розвитку телефонної технології, поява нових функцій та моделей приводять до постійного попиту на ці товари. Відповідно, віртуальні магазини, спеціалізовані на продажу мобільних телефонів та аксесуарів, стають все більш популярними та важливими у цьому сегменті ринку.

Мета даної кваліфікаційної роботи полягає у детальному вивченні та аналізі інтернет-магазинів мобільних телефонів та аксесуарів. В рамках дослідження будуть розглянуті основні аспекти створення та функціонування таких магазинів, дослідження методів створення програмного забезпечення відповідно до обраної теми.

Дослідження цієї теми має велике значення, оскільки дозволяє з'ясувати особливості та фактори успіху в електронній торгівлі мобільними пристроями.

Основне завдання кваліфікаційної роботи полягає в наступному:

- Аналіз подібних вебсайтів та сучасних трендів в розробці даного вебсайту;
- Визначення основних функціональних вимог до ПЗ;
- Проектування інтерфейсу та дизайну вебсайту;
- Розробка бази даних;
- Розробка функціоналу вебсайту та його вдосконалення;
- Тестування та оптимізація ПЗ.

Предметом дослідження кваліфікаційної роботи є аналіз та використання найкращих методів та засобів для розробки вебсайту, які допоможуть розробити зручний та функціональний вебсайт, який в подальшому буде завантажений на хостинг та буде використовуватися для продажу одного з найбільш популярних сегментів на сучасному ринку, а саме мобільних телефонів та аксесуарів.

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ СТВОРЕННЯ ВЕБСАЙТУ

1.1 Постановка задачі

Основна мета створення інтернет-магазину мобільних телефонів та аксесуарів полягає в подальшому його використанні. Це допоможе збільшити обсяги продажів офлайн магазину, збільшити впізнаваність бренду та розвинути свою торгову мережу.

Інтерфейс програмного продукту повинен бути максимально зручним у використанні для того, щоб покупці легко могли знайти потрібний їм товар. Функціонал вебсайту повинен мати усі основні аспекти, які дозволять користувачам легко робити замовлення, а саме:

- Пошук товарів;
- Перегляд детальної інформації про товар;
- Додавання товару в список бажаного;
- Додавання товару в кошик;
- Здійснення замовлення товару;
- Вибір зручної адреси доставки;
- Вибір зручного методу оплати.

Основні етапи розробки вебсайту включають наступні пункти:

1. Створення дизайну та інтерфейсу користувача:
 - 1.1. Вибір основних кольорів для вебсайту;
 - 1.2. Розробка дизайну головної сторінки;
 - 1.3. Розробка інтерфейсу;
2. Верстка сайту:
 - 2.1. Верстка головної сторінки сайту;
 - 2.2. Верстка статичних сторінок сайту (Інформація про компанію, співпраця, повернення/обмін товарів та інші);
 - 2.3. Верстка динамічних сторінок сайту.

3. Створення бази даних:

3.1. Створення таблиць бази даних

4. Розробка Backend частини:

4.1. Підключення бази даних;

4.2. Створення моделей та міграційних таблиць;

4.3. Міграція таблиць в БД;

4.4. Створення контролерів;

4.5. Створення Роутів;

5. Поєднання Backend та Frontend частин:

5.1. Визначення даних, які потрібно відобразити з backend на frontend

5.2. Реалізація взаємодії з API;

5.3. Тестування реалізованих методів та коректності їх роботи.

1.2 Аналіз схожих вебсайтів

В інтернеті можна знайти дуже велику кількість сайтів, які продають мобільні телефони, аксесуари до них та іншу електроніку та побутову техніку. Найпопулярнішими інтернет-магазинами у цій сфері належать наступним компаніям: Comfy, Hotline, Алло та Цитрус. На мою думку, це найкращі та найпопулярніші сайти в Україні, саме тому їх можна брати за приклад.

Аналізуючи вище зазначені веб-ресурси варто оцінювати їх за наступними критеріями:

- Дизайн;
- Функціональність;
- Зручність використання;
- Контент;
- Взаємодію з користувачем;
- Адаптивність для пристроїв;
- Надійність.

Перший сайт, який я розгляну буде **Hotline**. Це доволі зручний сайт, як для покупців так і для власників невеликих магазинів.

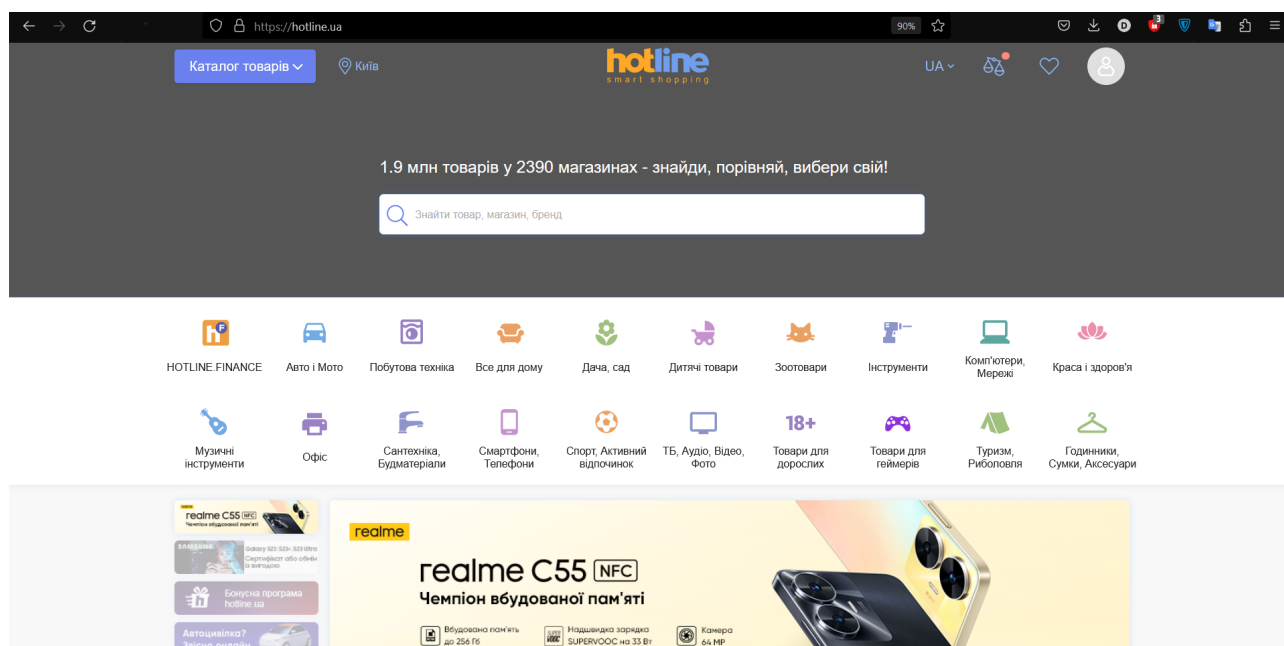


Рисунок 1.1 – Головна сторінка сайту Hotline

Сайт має зручну структуру, яка дозволяє користувачам швидко знайти необхідну інформацію. Головне меню містить категорії товарів, що спрощує пошук конкретної категорії товарів. Також на сайті присутній пошуковий рядок, який дозволяє швидко знайти конкретний товар за назвою або параметрами. На сайті представлений доволі великий каталог товарів від товарів для дому до великої побутової техніки. Завдяки функціоналу сайту користувачі можуть порівнювати різні товари за багатьма характеристиками просто додаючи їх до списку порівняння. Особисто для мене корисним є те, що Hotline порівнює ціни в різних інтернет-магазинах, що дозволяє обрати для себе найвигіднішу пропозицію.

Другий сайт, який хотілося б відзначити є **Comfy**. Сайт Comfy є одним з найбільших та найвідоміших інтернет-магазинів в Україні, спеціалізуючись на продажі різноманітної побутової техніки, електроніки, комп'ютерів та інших товарів.

Один з ключових аспектів сайту Comfy - його привабливий та сучасний дизайн. Вебсайт має чистий і легкий для сприйняття дизайн, добре

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						12

збалансований використанням кольорів та графіки. Навігаційне меню розташоване зручно і дозволяє легко знайти необхідні категорії товарів.

Крім того, Comfy пропонує різні способи оплати та доставки, забезпечуючи зручність і гнучкість для покупців. Клієнти можуть обирати між різними варіантами доставки, включаючи самовивіз з магазину. Також надається можливість отримати консультацію від спеціалістів і звернутися до сервісного центру у разі потреби.

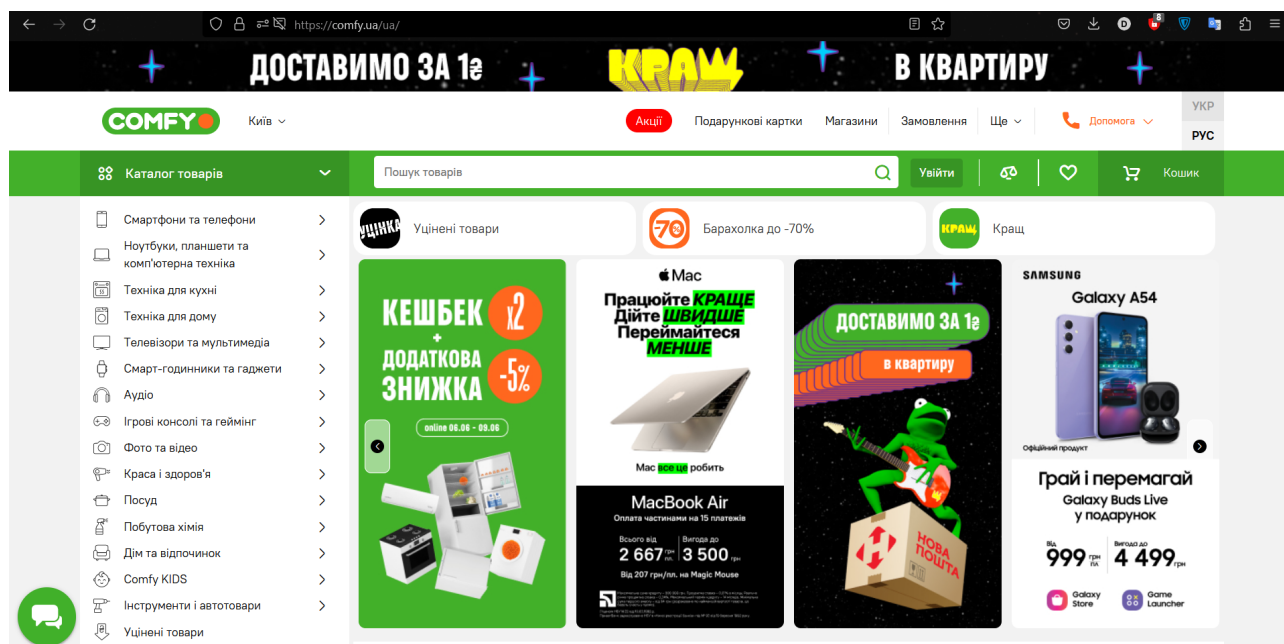


Рисунок 1.2 – Головна сторінка сайту Comfy

Сайт **Алло** є одним з найвідоміших та популярних інтернет-магазинів в Україні, спеціалізуючись на продажі електроніки, побутової техніки, гаджетів та інших товарів.

Дизайн сайту Алло приваблює своєю сучасністю та естетикою. Інтерфейс сайту є добре структурованим та організованим, що сприяє зручній навігації для користувачів. Навігаційне меню дозволяє швидко знайти потрібну категорію товарів, а також доступні додаткові фільтри та пошуковий рядок для точного пошуку.

Одна з головних переваг сайту Алло - широкий асортимент. Він пропонує великий вибір електроніки, мобільних телефонів, комп'ютерів, побутової техніки

та аксесуарів. Кожна сторінка товару містить докладну інформацію про продукт, включаючи фотографії, технічні характеристики, опис та відгуки користувачів.

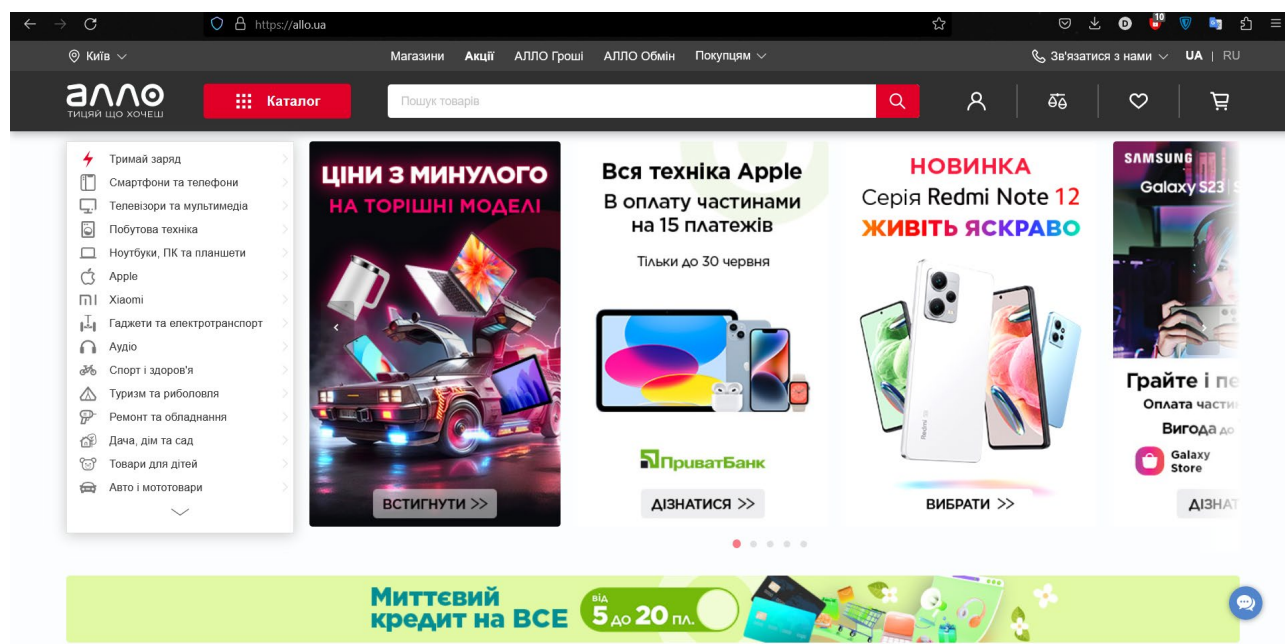


Рисунок 1.3 – Головна сторінка сайту Алло

Дизайн сайту Цитрус є сучасним, стильним та привабливим для користувачів. Інтерфейс сайту є добре організованим та зручним для навігації. Головне меню містить розділи товарів, що дозволяє користувачам легко знайти потрібну категорію. Також на сайті є можливість швидкого пошуку товарів за назвою або параметрами.

Асортимент сайту доволі різноманітний та містить великий перелік товарів таких як: телефони, планшети, ноутбуки, аксесуари, комп'ютери та багато іншого.

На сайті також присутня функція порівняння товарів, що дозволяє вибрати кілька товарів та вибрати той, який буде найбільше задовольняти потреби клієнта.

Крім того, сайт Алло пристосований для мобільних пристроїв і має адаптивний дизайн. Версія сайту для мобільних пристроїв зручна для перегляду та навігації, що дозволяє користувачам зручно здійснювати покупки на мобільних пристроях.

На сайті є можливість реєстрації та авторизації користувачів, що дозволяє зберігати адреси доставлення, переглядати історію замовлень тощо. Кожний товар має окрему сторінку з детальним описом, технічними характеристиками, відгуками покупців та фотографіями.

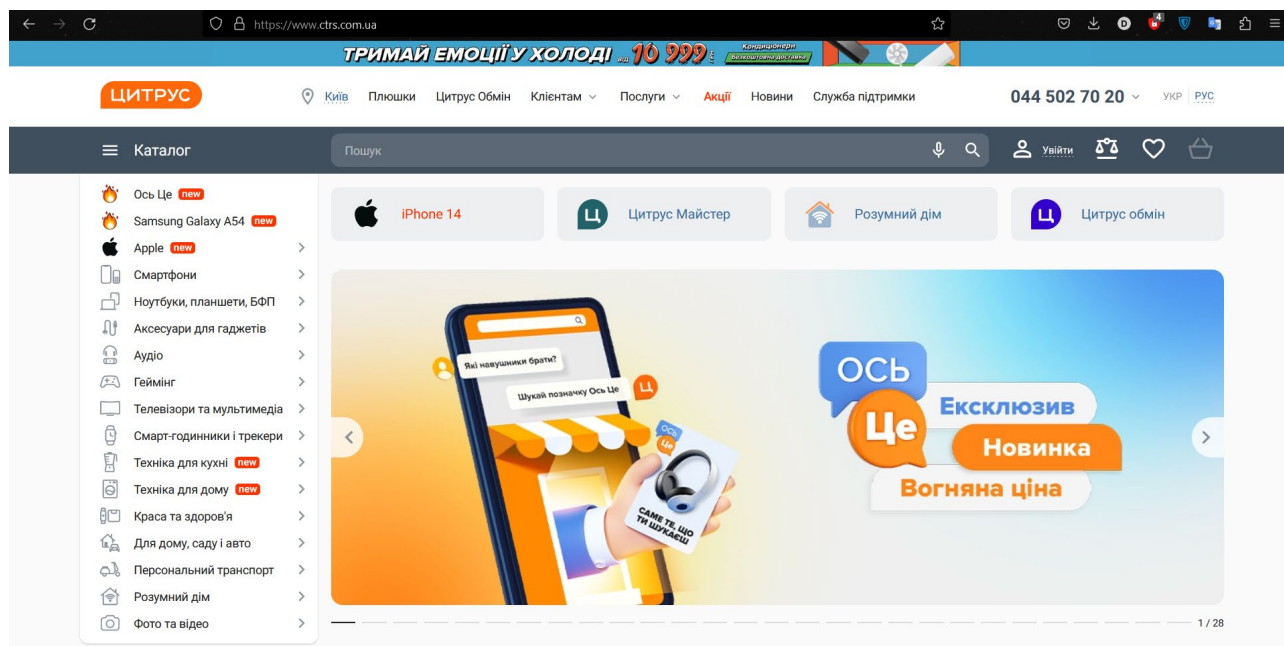


Рисунок 1.4 – Головна сторінка сайту Цитрус

Таблиця 1.1

Характеристика	Hotline	Comfy	Алло	Цитрус
Дизайн	Простий	Сучасний	Сучасний	Сучасний
Навігація	Зручна	Зручна	Зручна	Зручна
Можливість порівняння товарів	Є	Є	Є	Є
Асортимент	Доволі широкий	Широкий	Великий	Великий
Адаптивність	Є	Є	Є	Є
Можливість реєстрації та авторизації	Є	Є	Є	Є

В таблиці 1.1, яка наведена вище порівняно певні характеристики інтернет-магазинів, які я описував вище. Аналізуючи ці дані можна зробити висновок що кожен сайт має свою перевагу, але в загальному вони дуже схожі та всі вони максимально зручні у використанні.

1.3 Вибір архітектури програмного продукту

Надійність, ефективність та масштабованість системи залежить від багатьох факторів, але одним з головних є вибір архітектури програмного продукту. При розробці програмного продукту я обрав для себе клієнт-серверну архітектуру^[7] тому, що вона найбільше підходить для e-commerce проєктів. В першу чергу вона дає змогу розділити відповідальність, тобто клієнтська частина (frontend) відповідає за відображення усіх даних та взаємодію з клієнтом, а серверна частина (backend) відповідає за обробку запитів клієнтів, логіку роботи сайту, забезпечення безпеки та доступ до бази даних.

Також, клієнт-серверна модель допомагає оптимізувати використання ресурсів завдяки тому, що клієнтська частина виконується на пристроях самих користувачів, а серверна відповідно на фізичних серверах, що і дозволяє оптимізувати роботу сайту.

На мою думку, клієнт-серверна архітектура повністю задовольняти потреби e-commerce сайту, забезпечує ефективну, а головне надійну роботу усієї системи, а також відповідає сучасним стандартам розробки вебсайтів.

1.4 Обґрунтування вибору засобів та вимог до програмного забезпечення

Вивчаючи документацію та аналізуючи методи та засоби створення інтернет-магазину я обрав для себе дві технології, які на мою думку, найкраще підходять для цього, а саме React^[3] та Laravel^[1]. Завдяки цим технологіям був розроблений інтернет-магазин мобільних телефонів та аксесуарів.

React – це фреймворк, або іншими словами бібліотека JavaScript, яка використовується для розробки користувацьких інтерфейсів, тобто frontend частину.

В React використовуються компоненти, які можна розділити на два види: класові компоненти та функціональні компоненти.

Кожен компонент має свій стан та структуру, а також можуть обмінюватися даними з іншими компонентами, що полегшує процес розробки.

Завдяки тому, що в React підтримується використання JSX (JavaScript XML) я маю змогу легко комбінувати JavaScript та HTML-подібний синтаксис, що дозволяє нам легко створювати компоненти та використовувати їх в додатку.

В React існує велика кількість бібліотек, які полегшують та пришвидшують процес розробки. Прикладом може слугувати бібліотека «react-router», яка використовується для маршрутизації, а бібліотека «Axios^[12]» дозволяє легко взаємодіяти з сервером.

Laravel – це фреймворк, який написаний мовою програмування PHP. Завдяки цьому фреймворку значно спрощується процес розробки веб-додатків та надається доволі широкий набір функціональних можливостей.

Його перевага полягає в тому, що використовуючи зручний синтаксис можна визначати маршрути для URL-адрес, групувати ці маршрути, інтегрувати параметри, а також налаштовувати фільтри та middleware (проміжний шар програмного забезпечення, який дозволяє обробляти HTTP-запити перед тим, як вони досягнуть маршруту, або ж після його виконання) для обробки запитів.

Фреймворк Laravel має вбудовану систему керування базами даних, яка дозволяє легко інтерактивно взаємодіяти з базою даних. Це дає змогу зручно створювати, видаляти, оновлювати та отримувати дані з бази даних.

Тобто, React та Laravel це одні з найкращих фреймворків для розробки веб-додатків, які спрощують процес створення веб-додатків, та які дають можливість створювати масштабні проекти, які відповідають усім сучасним вимогам. Завдяки цим технологіям я зможу розробити функціональний та максимально зручний вебсайт для продажу мобільних телефонів та аксесуарів.

					ІІЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						17

ВИСНОВКИ ДО РОЗДІЛУ 1:

В першому розділі я визначив поставлені задачі на виконання кваліфікаційної роботи, розділили їх на блоки, слідуючи яким зможемо успішно реалізувати інтернет-магазин мобільних телефонів та аксесуарів.

В другій частині першого розділу я розглянув одні з найпопулярніших інтернет-магазинів України (Comfy, Hotline, Алло та Цитрус). Аналіз перелічених вебсайтів дав нам змогу визначити вимоги до дизайну та інтерфесу нашої майбутньої системи, а також я визначив вимоги до функціоналу інтернет-магазину.

Крім того, я визначив, що найкращою архітектурою для розробки вебсайту на обрану тему буде клієнт-серверна архітектура. Вона забезпечує поділ клієнтської та серверної частини, що дозволяє з легкістю реалізовувати функціонал інтернет-магазину.

В кінці розділу я розглядав найкращі засоби для розробки функціонального та зручного інтернет-магазину, на якому клієнти зможуть з легкістю знаходити потрібні їм товари та замовляти їх. Я обрав React для розробки клієнтської частини тому, що сьогодні це один з найпопулярніших фреймворків, які використовують для frontend частини, через його зручність та простоту використання. А для серверної частини я обрав фреймворк Laravel, який написаний мовою програмування PHP.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНТЕРНЕТ-МАГАЗИНУ ПРОДАЖУ МОБІЛЬНИХ ТЕЛЕФОНІВ

2.1 Визначення варіантів використання та структури системи

Інтернет-магазин створюється для того, щоб в подальшому його можна було використовувати для продажу мобільних телефонів та аксесуарів, що дасть змогу збільшити кількість клієнтів та впізнаваність бренду.

Для початку визначимо вимоги користувачів.

Внутрішній користувач – Адміністратор:

- Редагування вмісту сайту;
- Редагування вмісту сайту;
- Перегляд усієї необхідної інформації на сайті

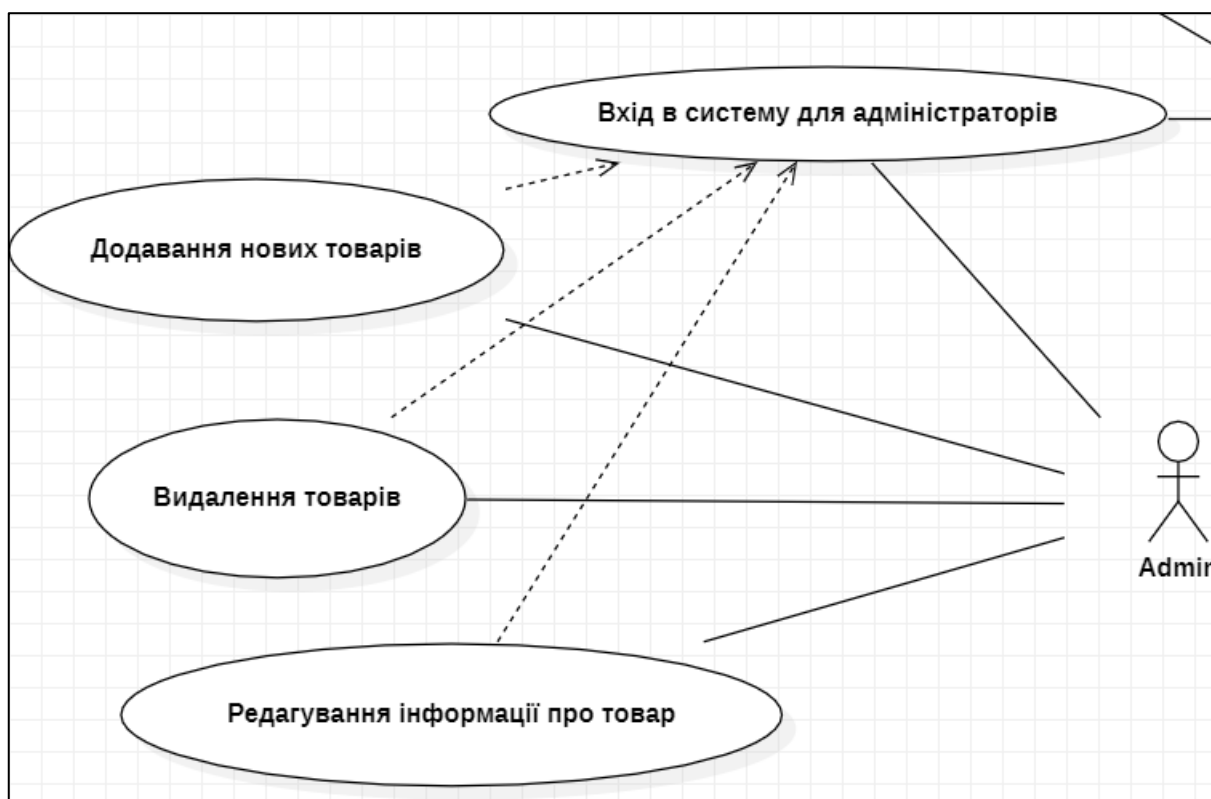


Рисунок 2.1 – Діаграма варіантів використання для Адміністраторів

Внутрішній користувач – Зареєстрований користувач:

- Перегляд вмісту сайту;

- Перегляд детальної інформації про товар;
- Пошук товарів за назвою;
- Додавання товару до списку бажаних;
- Додавання товару в кошик;
- Здійснення замовлення;

Зовнішній користувач – Гість:

- Перегляд вмісту сайту;
- Реєстрація свого акаунту;
- Авторизація.

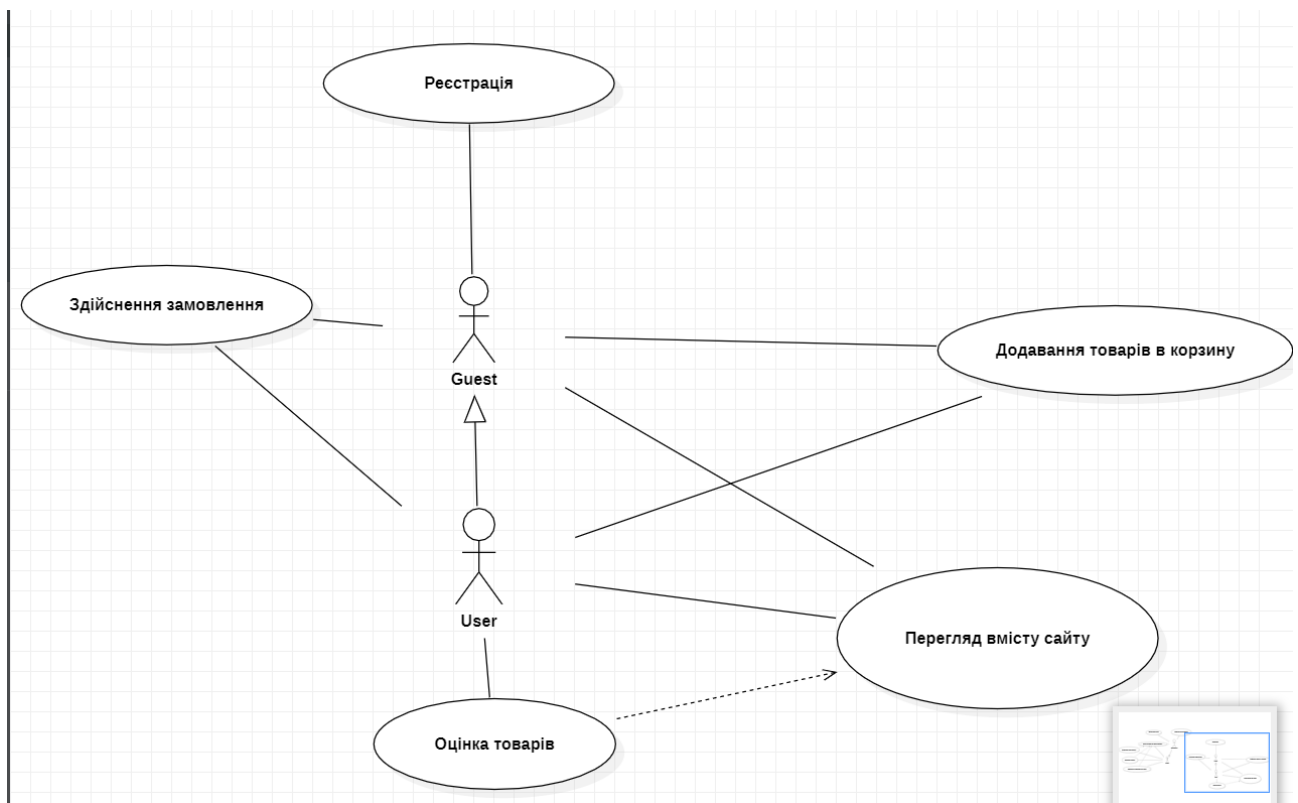


Рисунок 2.2 – Діаграма варіантів використання для Гостей та Зареєстрованих користувачів

Далі розглянемо вимоги до вебсайту:

функціональні вимоги:

- **Реєстрація та авторизація^[2]:** користувачі повинні мати змогу зареєструвати новий акаунт, або увійти у вже наявний акаунт.
- **Валідація:** усі дані, які вводить користувач повинні проходити валідацію;

- **Відображення результатів виконання операцій:** при виконанні будь-яких операцій (додавання товару в кошик, видалення товару зі списку бажаного та інше) користувач повинен отримувати повідомлення про статус виконання цієї операції (успішно, або не успішно).

- **Категоризація:** усі товари, які наявні на сайті повинні належати до певної категорії, щоб користувач міг з легкістю їх знайти.

- **Повноцінність інформації:** користувачі повинні бачити детальну інформацію про кожен товар та усі їх характеристики.

Системні вимоги:

- Система, на якій буде використовуватися сайт повинна підтримувати браузері нових версій та мати вихід в інтернет.

- Сервер, на якому буде розміщуватися веб-додаток повинен підтримувати такі технології як: PHP, React та MySQL.

Не функціональні вимоги:

Простота:

- Час, для освоєння вебсайту не повинен перевищувати 1-єї години, а для досвідчених користувачів 30 хвилин;

- Інтерфейс повинен бути максимально простим та зрозумілим;

- Час відповіді сайту на запити користувача не повинен перевищувати 2-5 секунд.

Враховуючи те, що веб-додаток поділений на frontend та backend частини варто окремо розглянути компоненти цих частин.

Frontend:

Основні компоненти, які містить ця частина поділені наступним чином:

Cart – компонент який відповідає за кошик, відображення вміст кошика та здійснення замовлення;

Common – компонент, який відображає статичні сторінки;

Favourite – компонент, який відповідає за список бажаних товарів, які додає користувач;

Home – компонент, в якому описуються основні елементи головної сторінки (категорії, слайдерів, меню та інші.)

Notification – компонент, який слугує для відображення повідомлень;

Others – компонент, в якому описані основні статичні сторінки, такі як Про нас, Співпраця та Повернення/Обмін;

Product Details – один з найважливіших компонентів, який передає детальну інформацію про товар.

Якщо ж говорити про **backend** частину, то в ній містяться контролери цих компонентів, їх моделі, а також таблиці для міграції.

Контролери:

CategoryController – контролер, в якому описаний метод отримання списку усіх категорій;

ContactController – цей контролер відповідає за записування даних з форми зв'язку в базу даних;

FavouriteController – відповідає за додавання та видалення товарів у списку бажаного, а також за виведення даних списку бажаного;

NotificationController – відповідає за виведення нових повідомлень;

ProductCartController – контролер, який відповідає за додавання товару в кошик, видалення з неї та за виконання певних функцій з товарами, які знаходяться в кошику;

ProductDetailsController – цей контролер відповідає за виведення детальної інформації про товар;

ProductListController – забезпечує виведення списку товарів по категорії, підкатегорії, або по пошуковому рядку;

ReviewController – відповідає за виведення коментарів до товарів;

SiteInfoController – виведення інформації про сайт;

SliderController – відповідає за виведення слайдера на головній сторінці;

VisitorController – відстежує коли користувач заходив на сайт.

Auth/Forget/Reset/User Controller – усі ці контролери відповідають за авторизацію, відновлення пароля, вхід та виведення даних про користувача.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						22

Також, на backend частині є моделі даних. Вони потрібні для здійснення різних взаємодій з даними.

І ще одним важливим елементом є компоненти Routes. В них прописуються шляхи до певних методів, які знаходяться в контролерах та надалі дають змогу взаємодіяти з ними на frontend частині.

2.2 Розробка бази даних вебсайту

База даних є важливим елементом для кожного онлайн-додатку. В ній зберігаються усі необхідні дані, які надалі можуть змінюватися в залежності від взаємодії користувачів з сайтом.

В інтернет-магазині мобільних телефонів та аксесуарів я використав phpMyAdmin для керування базою даних.

Першим кроком при розробці бази даних було її створення та підключення до проєкт. Для цього я створив базу даних в самому phpMyAdmin, а після цього виконав певні налаштування в файлі .env для того, щоб під'єднати базу даних.

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=apimayak
DB_USERNAME=root
DB_PASSWORD=
```

Рисунок 2.3 – Фрагмент з файлу .env, який відповідає за підключення до бази даних

Кожне поле відповідає за певний параметр.

DB_CONNECTION=mysql – цей рядок вказує тип підключення до бази даних, в нашому випадку це MySQL.

DB_HOST=127.0.0.1 – вказує адресу бази даних. Через те, що сервер знаходиться на локальному пристрої тому представлена IP-адреса 127.0.0.1

DB_PORT=3306 – зазначає порт, через який і відбувається підключення до бази даних.

DB_DATABASE=apimayak – вказується назва бази даних, яка зазначена в phpMyAdmin.

DB_USERNAME=root – вказується логін користувача для авторизації в phpMyAdmin

DB_PASSWORD= - вказує пароль, який повинен ввести користувач, щоб отримати доступ до бази даних.

Після того, як база даних була підключена я створював моделі для компонентів виконувачи просту команду в терміналі «php artisan make:model Назва моделі -m»

Ця команда створює потрібну нам модель, а “-m” допомагає відразу створити таблицю для міграції, яку я надалі зможу змінити, додавши до неї усі необхідні нам поля та зазначити типи цих полів.

Наступним кроком я налаштовував створену модель. Я додав до неї рядок «protected \$guarded = []»;», який означає що всі поля таблиці є доступними для масового заповнення.

Після цього у файлі міграції, який був створений я додавав потрібні мені поля для таблиці та відповідно вказував тип цих полів. Для більшості полів я використовував значення «string». Використання значення «string» має багато переваг, завдяки цьому я можу зберігати будь-яку текстову інформацію, рядки, URL-адреси чи JSON-об'єкти. Також робота з такими полями є доволі простою та зрозумілою. Для полів, які мають приймати доволі великий обсяг тексту я обра значення «text». Тип «text» дозволяє зберігати текст з відповідним форматуванням, розділовими знаками, переносами та іншими структурними елементами. Ще однією перевагою типу «text» є менше використання пам'яті порівняно з типом «string», якщо зберігаються довгі текстові значення. Крім того, їх легше редагувати ніж поля типу «string» бо нам не потрібно розділяти їх на менші частини.

```

public function up(): void
{
    Schema::create('product_details', function (Blueprint $table) {
        $table->id();
        $table->integer('product_id');
        $table->string('image_one');
        $table->string('image_two');
        $table->string('image_three');
        $table->string('image_four');
        $table->string('short_description');
        $table->string('color');
        $table->string('memory');
        $table->text('long_description');
        $table->string('product_brand');
        $table->string('product_type');
        $table->string('product_ram');
        $table->string('builtin_memory');
        $table->string('slot_expansion');
        $table->string('simcard_type');
        $table->string('product_processor');
        $table->string('product_number_of_cores');
        $table->string('product_frequency');
        $table->string('product_fast_charge');
        $table->string('product_battery');
        $table->string('product_diagonal');
        $table->string('product_resolution');
        $table->string('product_matrix_type');
        $table->string('product_refresh_rate');
        $table->string('product_display_features');
        $table->string('product_brightness');
        $table->string('product_main_camera');
        $table->string('product_video');
        $table->string('product_front_camera');
        $table->string('product_communication_standards');
        $table->string('product_wifi');
        $table->string('product_gps');
        $table->string('product_nfc');
        $table->string('product_interface_connector');
        $table->string('product_height');
    });
}

```

Рисунок 2.4 – Фрагмент коду з файлу, який використовується для міграції (див. додаток А)

Після виконання цих кроків потрібно виконати міграцію до бази даних наступною командою «php artisan migrate». Ця команда виконує міграцію усіх таблиць, яких ще немає в БД, тобто створює таблицю в phpMyAdmin, яку я можу надалі використовувати для записування даних, отримання цих даних, або їх змінювати.

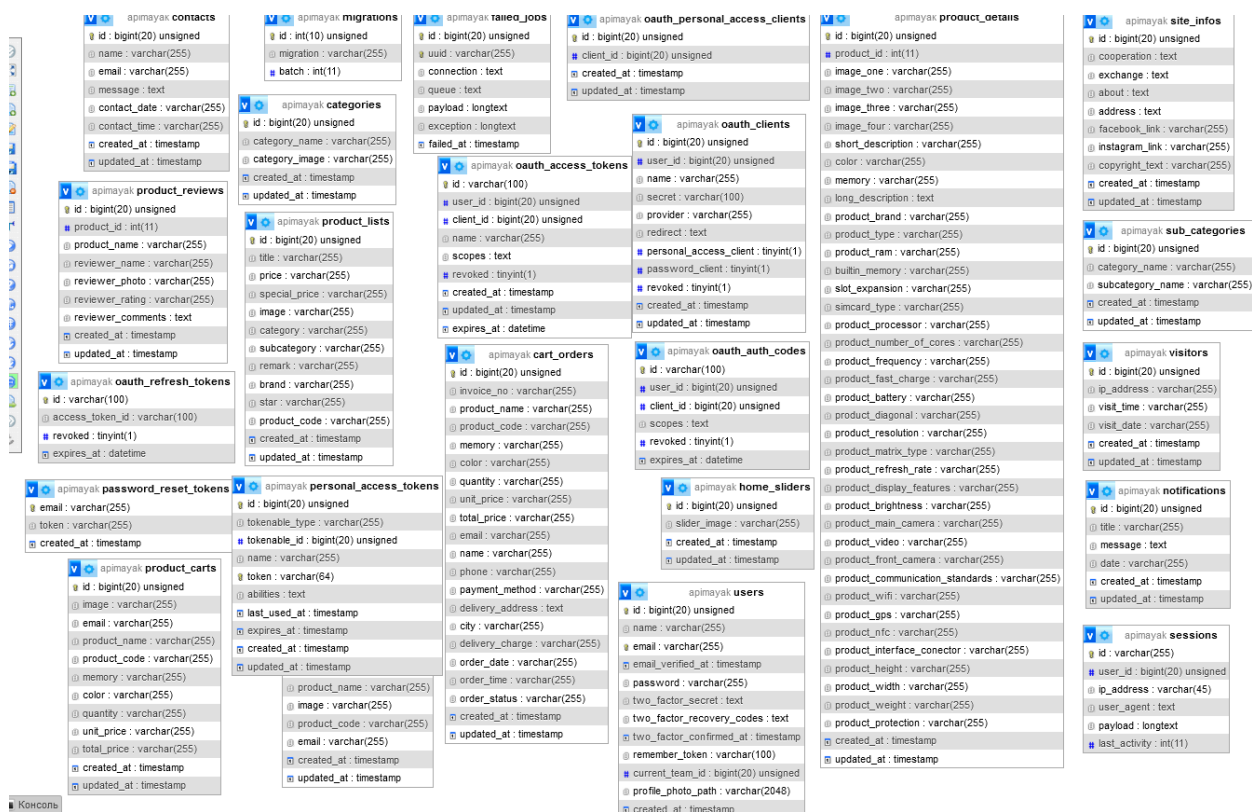


Рисунок 2.5 – Загальний вигляд діаграми бази даних

На мою думку, це доволі легкий спосіб створення бази даних, який не займає багато часу, а головне є зручним.

2.3 Проектування та реалізація алгоритмів роботи вебсайту

Основним елементом, який описує процес використання ресурсу є діаграма діяльності в якій описані різні процеси дій користувачів.

На зображенні 2.6 наведеному нижче показана загальна діаграма діяльності усіх користувачів. Тобто це діаграма, яка показує яким чином користувачі зможуть взаємодіяти з нашим веб-ресурсом залежно їхньої ролі та прав доступу до інтернет-магазину мобільних телефонів та аксесуарів.

На діаграмі можна побачити, що користувачі в яких роль «Адміністратор» мають доступ до адмін-панелі, інші ж користувачі, в яких роль «Гість», або «Користувач» зможуть користуватися лише frontend частиною. Ця функція реалізована за допомогою клієнт-серверної архітектури, яка дозволяє «розділити» наш веб-додаток на дві різні частини, які будуть пов'язані між собою.

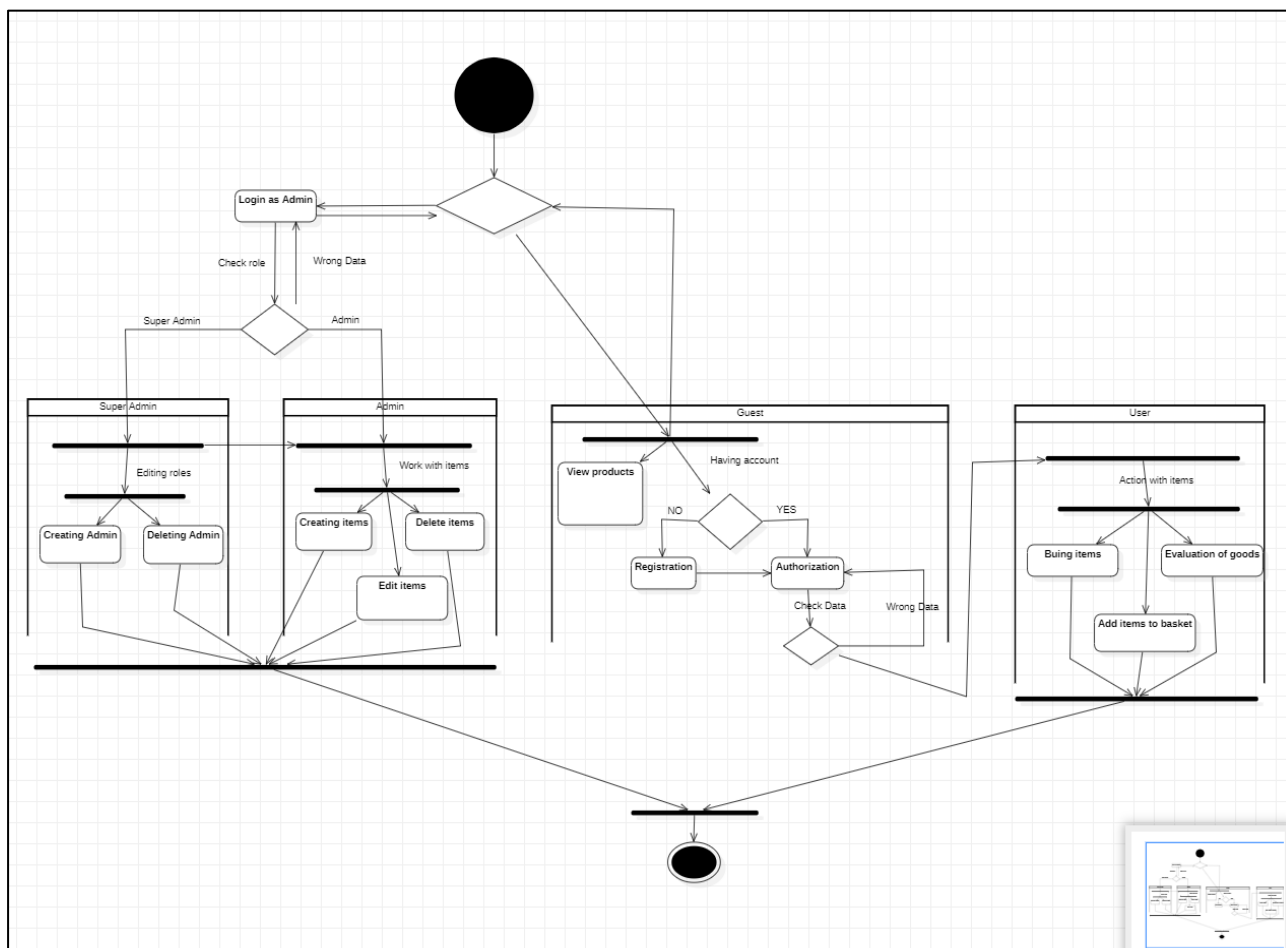


Рисунок 2.6 – Загальна діаграма діяльності дій користувачів.

На рисунку 2.6 показана загальна діаграма діяльності усіх користувачів.

На ній можна побачити, що на сайті існує кілька ролей. Головна роль належить адміністратору тому, що це людина, яка відповідає за наповнення сайту, за коректне відображення усіх елементів, відстежування приймання заявок та багато іншого.

Також на діаграмі є роль «гість». Ця роль автоматично присвоюється користувачу, який не зареєстрований, або не авторизований на сайті. В такого користувача є доступ лише до перегляду вмісту сайту та можливість зв'язку зі службою підтримки.

Роль «User» відповідає користувачу, який зареєструвався в системі та авторизувався, такий користувач може використовувати увесь функціонал сайту, робити замовлення, додавати товар до списку улюблених та інше.

Розглянемо діаграми діяльності для окремих користувачів.

Діаграма діяльності для користувача в якого є доступ Адміністратора:

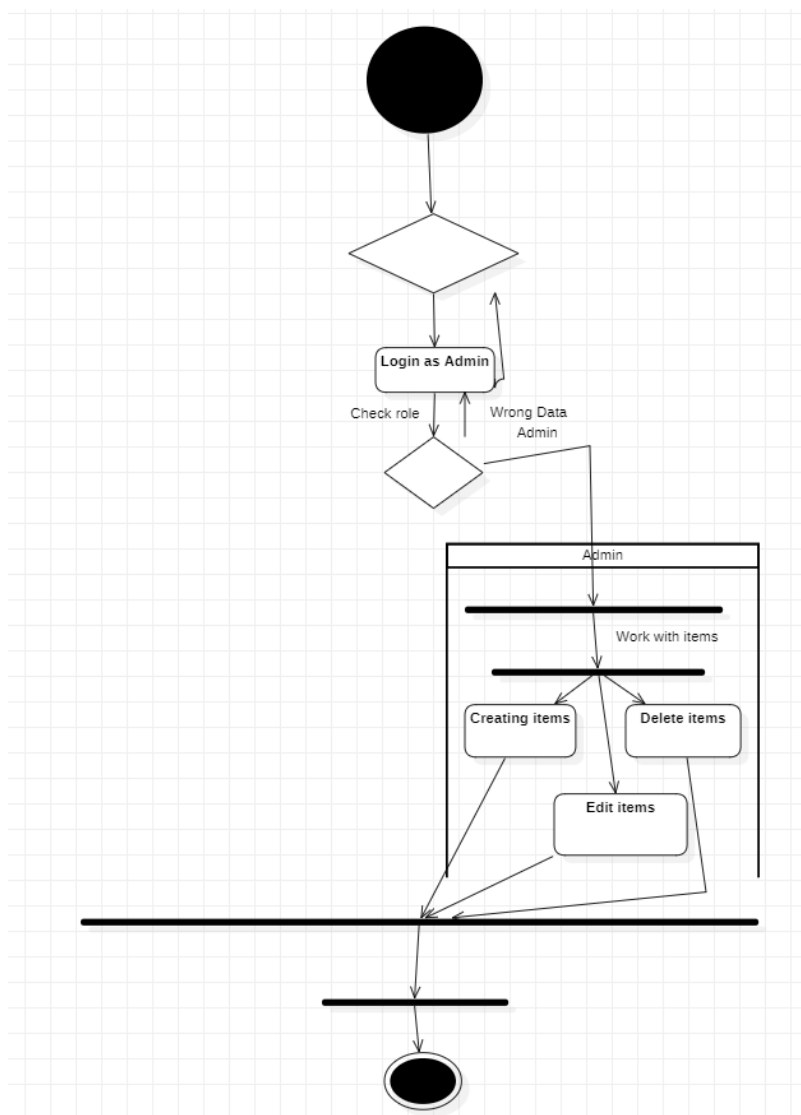


Рисунок 2.7 – Діаграма діяльності для Адміністратора

На діаграмі видно, що адміністратор має доступ до різних елементів керування сайту. Він може додавати нові товари, додавати категорії, підкатегорії та інші елементи. Також, адміністратор може видаляти товари, категорії, підкатегорії, або ж редагувати інформацію, яка міститься на сайті (слайдери, список товарів, список категорій та списки підкатегорій, які закріплені під категоріями). Крім того, адміністратор може переглядати статус опрацювання заявок, які залишили користувачі, а також, він може опрацьовувати ті заявки, які ще не були опрацьованими. Тому роль адміністратора вважається однією з найголовніших, вона забезпечує функціонування всієї системи.

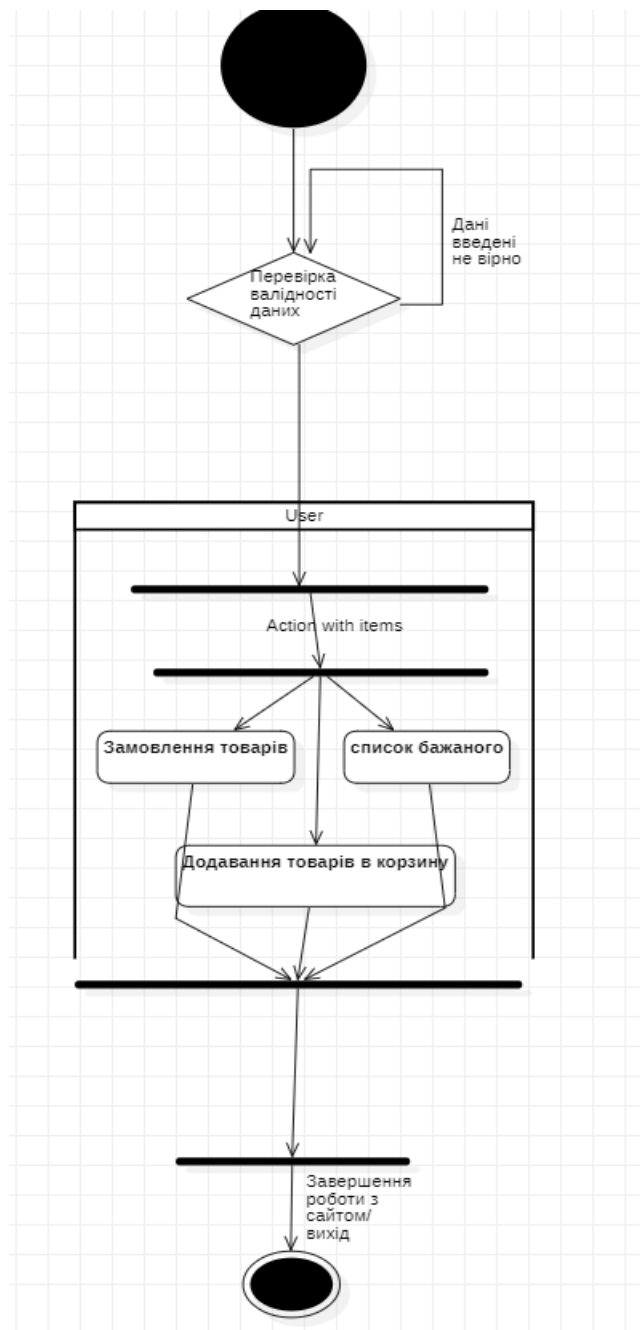


Рисунок 2.8 – Діаграма діяльності для зареєстрованого раніше користувача (User)

На діаграмі видно, що користувач, який не має акаунту, або який не авторизувався може переглядати увесь вміст сайту, але додати товар до списку бажаного, чи до кошика він не зможе, відповідно він не зможе зробити замовлення.

Для того, щоб користувач мав доступ до цих функцій він повинен бути авторизований. Після цього він зможе додавати товар до кошика та списку бажаних.

При розробці вебсайту була створена діаграма класів, яка наведена на рисунку 2.9.

На діаграмі класів представлено 6 основних класів таких як:

- Admin – відповідає за клас адміністраторів;
- User – відповідає за користувачів та містить наступні поля: id, login, email, firstName, lastName, password та інші;
- Product – використовується для зберігання інформації про товар та містить поля: id, name, description, price, image та інші;
- Image – відповідає за збереження інформації про фото товарів;
- Characteristic – відповідає за детальні характеристики товарів;
- Response – містить наступні поля (id, text, rate)

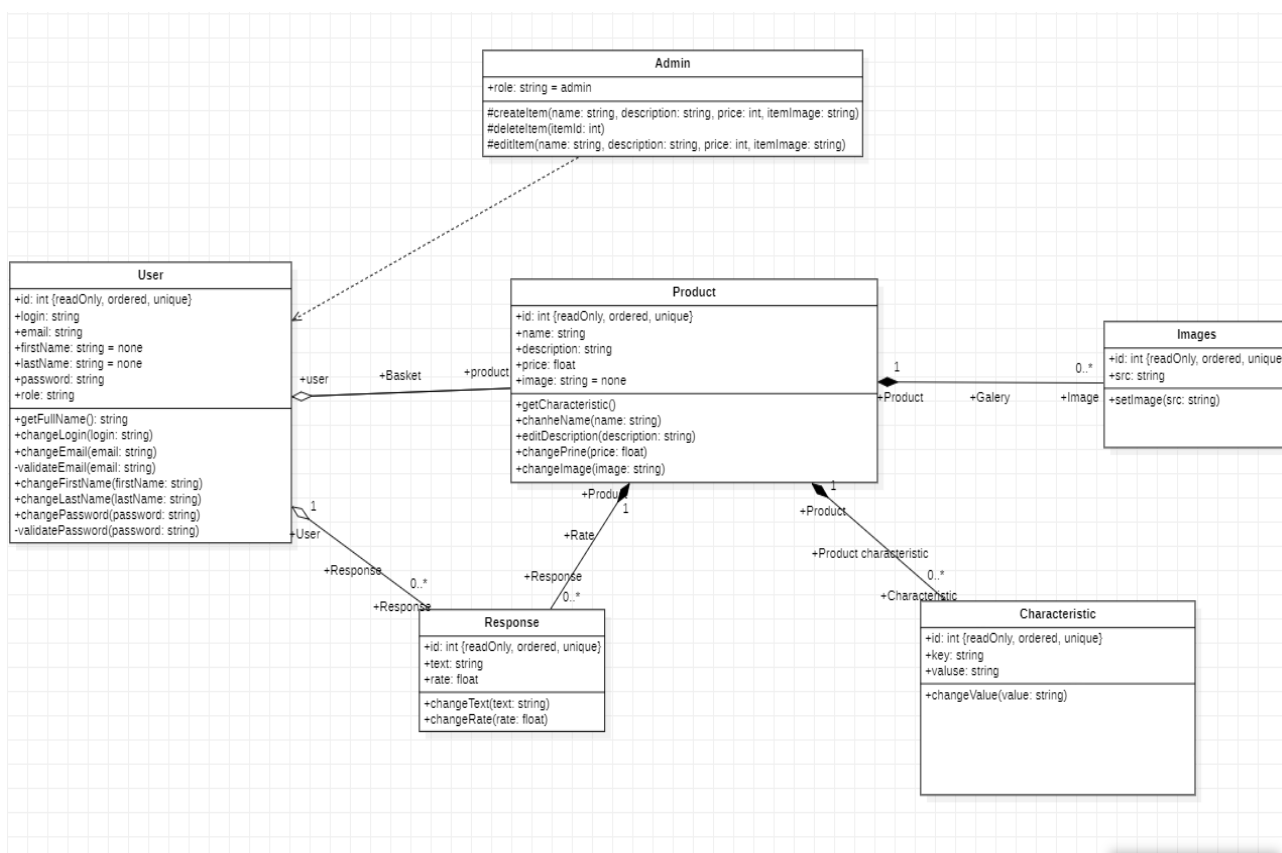


Рисунок 2.9 – Діаграма класів

На діаграмі 2.9 можна ознайомитися з усіма класами, які реалізовані в проєкті та переглянути які поля в них містяться. Це дасть змогу використати ці класи при реалізації певних функцій та компонентів.

2.4 Реалізація функціонала інтернет-магазину мобільних телефонів та аксесуарів.

Для опису функціонала нашого ПЗ потрібно зрозуміти, що наша система поділена на дві частини frontend та backend. Ці системи пов'язані тому, якщо у нас буде реалізований певний метод на backend, але при цьому він не буде поєднаний з frontend то користувач не зможе скористуватися цим функціоналом.

Для того, щоб все працювало правильно та я мав доступ з компонентів React до компонентів Laravel нам потрібно прописувати відповідні роути для наших методів.

В проєкті є кілька місць де вони прописуються. Перше це файл api.php, який відповідає за роути, які використовуються для взаємодії з користувачами, другий файл це web.php, який використовується для взаємодії з адміністраторами, тобто виведення потрібної інформації в адмін-панелі, редагування, або видалення даних використовуючи адмін-панель.

```
//////////////////// LOGIN API //////////////////////////////////////

// Issue Access Token
Route::post('/oauth/token', '\Laravel\Passport\Http\Controllers\AccessTokenController@issueToken')->middleware(['api', 'throttle']);

//Login Route
Route::post('/login', [AuthController::class, 'Login']);

//Register Route
Route::post('/register', [AuthController::class, 'Register']);

//Forget Password Route
Route::post('/forgetpassword', [ForgetController::class, 'ForgetPassword']);

//Reset Password Route
Route::post('/resetpassword', [ResetController::class, 'ResetPassword']);

//User Profile Route
Route::get('/user', [UserController::class, 'User'])->middleware('auth:api');

////////////////////

//Get Visitor
Route::get('/getvisitor', [VisitorController::class, 'GetVisitorDetails']);

//Post Contact
Route::post('/postcontact', [ContactController::class, 'PostContactDetails']);

// SiteInfo
Route::get('/allsiteinfo', [SiteInfoController::class, 'AllSiteinfo']);

//All Category
Route::get('/allcategory', [CategoryController::class, 'AllCategory']);

//Product List By Remark
Route::get('/productlistbyremark/{remark}', [ProductListController::class, 'ProductListByRemark']);
```

Рисунок 2.10 – Фрагмент з файлу api.php

В кожному роуті я прописую шлях, по якому повинен викликатися метод «'/user'», контролер (UserController), в якому описаний наш метод та відповідно назва самого методу (User).

В React частині також є файл, який відповідає за роутинг. Конкретно в моєму проєкті він має назву AppUrl.jsx. Даний файл забезпечує взаємодію між методами, які я використовую в серверній частині та методами, які використовуються в користувацькій частині.

```
class AppUrl{
  static BaseURL = "http://127.0.0.1:8000/api"
  static VisitorDetails = this.BaseURL+"/getvisitor"
  static PostContact = this.BaseURL+"/postcontact"
  static AllSiteInfo = this.BaseURL+"/allsiteinfo"
  static AllCategoryDetils = this.BaseURL+"/allcategory"
  static ProductListByRemark(Remark){
    return this.BaseURL+"/productlistbyremark/" + Remark;
  }
  static ProductListByCategory(category){
    return this.BaseURL+"/productlistbycategory/" + category;
  }
  static ProductListBySubCategory(category, subcategory){
    return this.BaseURL+"/productlistbysubcategory/"+category+"/"+subcategory;
  }
  static AllSlider = this.BaseURL+"/allslider"

  static ProductDetails(code){
    return this.BaseURL+"/productdetails/" + code;
  }

  static NotificationHistory = this.BaseURL+"/notification"

  static ProductBySearch(searchkey){
    return this.BaseURL+"/search/" + searchkey;
  }

  static SimilarProduct(code){
    return this.BaseURL+"/similar/" + code;
  }

  static ReviewList(code){
    return this.BaseURL+"/reviewlist/" + code;
  }
}
```

Рисунок 2.11 – Фрагмент файлу AppUrl.jsx

Файл представлений вище слугує для представлення класу AppUrl, в якому прописані статичні поля та методи для встановлення URL-адрес інтернет-магазину. Головна мета цього файлу полягає в забезпеченні легкого та зручного доступу до різних ендпоінтів API.

Для кращого розуміння розглянемо метод `ProductListByRemark`. Він генерує URL, для того, щоб я мав змогу отримати список продуктів із заданим значенням «Remark».

Цей файл дозволяє керувати URL-адресами API, що полегшує взаємодію з ними. Використовуючи цей клас, я можу швидко отримати доступ до потрібного ендпоінту та зробити відповідний запит до сервера.

Крім перерахованого вище у нас є контролери, в яких описані конкретні методи, тому розглянемо їх по черзі.

CategoryController:

Контролер, який містить метод `AllCategory`. Він відповідає за повернення списку усіх наявних категорій.

ContactController:

Контролер містить метод `PostContactDetails`, який відповідає за внесення даних про повідомлення і саме повідомлення до бази даних.

FavouriteController:

В даному контролері було реалізовано три методи:

`AddFavourite` – відповідає за додавання товару до списку бажаних товарів;

`FavouriteListItem` – виводить список усіх товарів у списку бажаних товарів;

`FavouriteRemove` – відповідає за видалення товарів зі списку бажаних.

NotificationController:

Головна функція цього контролера це виведення повідомлень

ProductCartController:

В даному контролері було створено 7 методів:

`addToCart` – додавання товарів до кошика;

`CartCount` – рахує кількість товарів доданих у кошик;

`CartList` – виводить список доданих товарів у кошику;

`CartItemPuls` – збільшення кількості певного товару у кошику;

`CartItemMinus` – зменшення кількості певного товару у кошику;

`CartOrder` – відповідає за оформлення замовлення, а точніше за зберігання даних про замовлення у кошику.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						33

ProductDetailsController:

Містить лише один метод ProductDetails, який виводить дані про конкретний товар.

ProductListController:

Містить в собі 5 методів

ProductListByRemark – виведення списку товарів в яких однакове поле «remark»

ProductListByCategory – виведення списку товарів в яких однакове поле «category»

ProductListBySubCategory– виведення списку товарів в яких однакове поле «subcategory»

ProductBySearch– виведення списку товарів відповідно до запиту користувача через функцію пошуку.

SimilarProduct– виведення списку схожих товарів на сторінці з детальним описом товарів.

SiteInfoController:

Виводить інформацію про сайт у футері.

SliderController:

Відповідає за виведення картинок в головному слайдері на головній сторінці.

VisitorController:

Відповідає за отримання ір-адреси користувача, який зайшов на сайт.

Auth/Forget/Reset/User контролери відповідають за авторизацію, зміну пароля, якщо користувач його не пам'ятає, просто зміну пароля та інформацію про користувача відповідно. Цей функціонал реалізований за допомогою пакету Laravel, який має назву «passport»^[2].

Після того як я створив певний метод в контролері я передаю його завдяки роутам, про які було написано раніше. Далі я використовую ці методи в дочірніх елементах.

Для відображення на сайті однієї сторінки я використовую один батьківський елемент, для прикладу візьмемо файл **ProductCategoryPage**. В цей файл я передаю дочірні елементи, такі як: **NavMenuDesktop**, **NavMenuMobile**, **Category** та **DesktopFooter**.

```
import React, { Component, useEffect, useState } from 'react'
import { Fragment } from 'react'
import { useParams } from 'react-router-dom'
import NavMenuDesktop from '../components/common/NavMenuDesktop'
import NavMenuMobile from '../components/common/NavMenuMobile'
import DesktopFooter from '../components/common/DesktopFooter'
import Category from '../components/productDetails/Category'
import axios from 'axios'
import AppUrl from '../api/AppUrl'

const ProductCategoryPage = () => {
  const { category } = useParams();
  const [productData, setProductData] = useState([]);

  useEffect(() => {
    window.scrollTo(0, 0);
    // alert(category);
    axios.get(AppUrl.ProductListByCategory(category)).then(response => {
      setProductData(response.data);
    }).catch(error => {
    });
  }, [category]);

  return (
    <Fragment>
      <div className='Desktop'>
        <NavMenuDesktop />
      </div>

      <div className='Mobile'>
        <NavMenuMobile />
      </div>

      <Category Category={category} ProductData={productData} />
      <DesktopFooter />
    </Fragment>
  )
}

export default ProductCategoryPage
```

Рисунок 2.12 — Приклад батьківського елемента в який передаються дочірні, на основі файлу ProductCategoryPage

Для коректного відображення цієї сторінки нам потрібно створити усі дочірні елементи та передати їх в цей файл. Передаються в самому верху програмного коду використовуючи «import».

Для прикладу дочірнього елемента візьмемо файл Category.jsx, який відповідає за виведення категорій на сторінку нашого сайту.

```
import React from 'react'
import { Fragment } from 'react'
import { Container, Row, Col, Card } from 'react-bootstrap'
import { Link } from 'react-router-dom'
import AppUrl from '../../api/AppUrl';
import axios from 'axios';
import Breadcrumb from 'react-bootstrap/Breadcrumb';

const Category = ({ Category, ProductData }) => {

  const MyView = ProductData.map((ProductList, i) => {
    if (ProductList.special_price === "no") {
      return (
        <Col className='p-0' xl={3} lg={3} md={3} sm={6} xs={6} key={i}>
          <Link to={"/productdetails/"+ProductList.id}>
            <Card className='image-box card w-100'>
              <img className='center w-75' src={ProductList.image} alt={ProductList.title} />
              <Card.Body>
                <h5 className='product-name-on-card'>{ProductList.title}</h5>
                <p className='product-price-on-card'>Ціна: ${ProductList.price}</p>
              </Card.Body>
            </Card>
          </Link>
        </Col>
      );
    } else {
      return (
        <Col className='p-0' xl={3} lg={3} md={3} sm={6} xs={6} key={i}>
          <Link to={"/productdetails/"+ProductList.id}>
            <Card className='image-box card w-100'>
              <img className='center w-75' src={ProductList.image} alt={ProductList.title} />
              <Card.Body>
                <h5 className='product-name-on-card'>{ProductList.title}</h5>
              </Card.Body>
            </Card>
          </Link>
        </Col>
      );
    }
  });

  return (
    <Fragment>
      <h2>{Category}</h2>
      <div>
        {MyView}
      </div>
    </Fragment>
  );
}
```

Рисунок 2.13 – Фрагмент коду дочірнього елемента Category.jsx

На рисунку 2.13 вище представлений приклад коду дочірнього елемента, в якому використовуються ті методи, які були створені на Backend частині. Завдяки тому, що використовується розширення jsx я можу використовувати в цьому коді JavaScript код в поєднанні з HTML-кодом, що дає можливість виводити потрібні нам елементи на екран та працювати з ними реалізуючи усі необхідні нам функції, які зможуть використовувати користувачі.

Для того, щоб усі елементи коректно відображалися на сторінці я використовую CSS-код. В проєкті є один загальний файл для CSS-коду, який має назву custom.css. В цей файл передаються інші CSS файли. Це пришвидшує процес розробки та робить його ще зручнішим. Це зменшує кількість запитів і одночасно пришвидшує відповідь від сервера. Також, це дає змогу визначити глобальні стилі, які в подальшому будуть використовуватися до всього веб-додатку. І головною перевагою цього методу є спрощення архітектури нашого додатку.

```
@import url("https://fonts.googleapis.com/css2?family=Montserrat:wght@300;600&family=Roboto:wght@500&display=swap");
@import "bigSlider.css";
@import "button.css";
@import "card.css";
@import "footer.css";
@import "mainMenu.css";
@import "navMenuDesktop.css";
@import "navMenuMobile.css";
@import "typo.css";
@import "megaMenuMobile.css";
@import "responsive.css";
@import "mainMenuAll.css";
@import "login.css";
@import "common.css";
@import "cartList.css";
@import "productDetails.css";

body{
  background-color: white;
}
```

Рисунок 2.14 – Вміст файлу custom.css

На рисунку 2.14 видно, що інші CSS файли передаються за допомогою «@import Назва файлу»

```
html {
  height: auto !important;
}
body {
  background-color: #FAFDDB !important;
  font-size: 15px;
  font-family: 'Roboto Condensed', sans-serif;
  font-weight: 300;
  color: #212121;
  position: initial !important;
  min-height: initial !important;
  top: auto !important;
}

.goog-te-banner-frame {
  top: initial;
  bottom: 0;
  display: none;
}

.BetweenTwoSection{
  margin-top: 60px;
  margin-bottom: 60px;
}

.TopSection{
  margin-top: 100px;
}

.Link{
  text-decoration: none !important;
}

.Link:hover{
  text-decoration: none !important;
}
```

Рисунок 2.15 – Css код файлу common.css

На рисунку 2.15 представлений приклад написання CSS коду, який відповідає за стилізацію елементів на сторінці. Надалі ці елементи можна використовувати прописавши до потрібного елементу, або блоку наступне «className=»Клас»». Програмний код представлених файлів див. Додаток А

ВИСНОВКИ ДО РОЗДІЛУ 2:

В другому розділі я розглянув усі етапи розробки інтернет-магазину мобільних телефонів та аксесуарів почавши з розгляду діаграм варіантів використання вебсайту для таких ролей як адміністратор (Admin) та зареєстрований користувач (User).

Далі я розглянув процес розробки бази даних, який починається із створення самої бази даних та підключенні її до серверної частини. Після цього, я розглянув процес створення таблиць, додавання в них полів та призначення доданим полям відповідних типів та назв, а потім визначили метод, який забезпечує міграцію наших таблиць до бази даних.

Також, я розглянув приклад створених моделей та контролерів та визначили, що контролери слугують для опису в них методів, які потім зможемо використовувати для реалізації функціоналу як на серверній частині, так і на клієнтській.

Я розібрав яким чином React працює разом з Laravel та що потрібно для того, щоб і серверна, і клієнтська частини працювали коректно та забезпечували швидку відповідь на запити користувачів.

Більше того, я розглянув приклад файлів, які відповідають за розміщення елементів сторінки та функціоналу, а також розглянули CSS файли, які відіграють ключову роль в стилізації елементів сторінок та всього сайту в цілому і розглянули структуру реалізації CSS файлі в нашому проєкті.

Не менш важливим є метод реалізації реєстрації та авторизації користувача за допомогою пакету фреймворку Laravel, який має назву «passport». Це доволі зручний інструмент, який дозволяє налаштувати аутентифікацію.

Головною перевагою є те, що він забезпечує високий рівень безпеки для аутентифікації API. Такий рівень безпеки забезпечується завдяки шифруванню та хешуванню для зберігання та перевірки токенів доступу. Окрім цього, це дає змогу забезпечити поділ користувачів, які мають доступ до перегляду сайту і тих, хто має доступ до повного функціоналу інтернет-магазину.

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНТЕРНЕТ-МАГАЗИНОМ МОБІЛЬНИХ ТЕЛЕФОНІВ ТА АКСЕСУАРІВ

3.1 Порядок встановлення та налаштування веб-додатку.

Для того, щоб вебсайт коректно працював і користувачі могли ним користуватися нам потрібно використовувати хостинг. Враховуючи те, що я використовую клієнт-серверну архітектуру то backend частина та база даних будуть знаходитися на одному сервері. При цьому сервер зможе швидко відповідати на запити користувача та відправляти потрібні йому дані.

Для того, щоб локально протестувати наш вебсайт достатньо встановити XAMPP Control Panel в якій я запускаю два модулі: Apache та MySQL. Після цього я відкриваю вікно термінала та переходимо у папку де розміщена наша backend частина і виконуємо наступні команди «php artisan serve» та «npm run dev». Після цього Backend частина буде запущена.

Для того, щоб запустити Frontend частину я так само через термінал переходжу в папку розміщення Frontend та виконуємо наступну команду «npm run start».

Після виконаних вище кроків наш сайт буде готовий до тестування.

3.2 Структура інтерфейсу. Інтерфейс та порядок роботи з вебсайтом

Для того, щоб користувач міг використовувати сайт йому потрібно перейти за посиланням на цей сайт, або відкрити збережену вкладку, якщо користувач раніше відвідував даний сайт, після чого він потрапить на головну сторінку сайту.

На головній інтернет-магазині сторінці користувач зможе знайти новини про останні пропозиції, або про новинки чи знижки, які зараз діють на сайті. Також, він зможе ознайомитися з іншою потрібною йому інформацією, яка буде йому корисна при знайомстві з нашим ресурсом, або з певним товаром. Окрім вище зазначеного, користувач зможе використати меню, або пошуковий рядок

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						40

для того, щоб знайти потрібний йому товар не витрачаючи на це багато часу та зусиль.

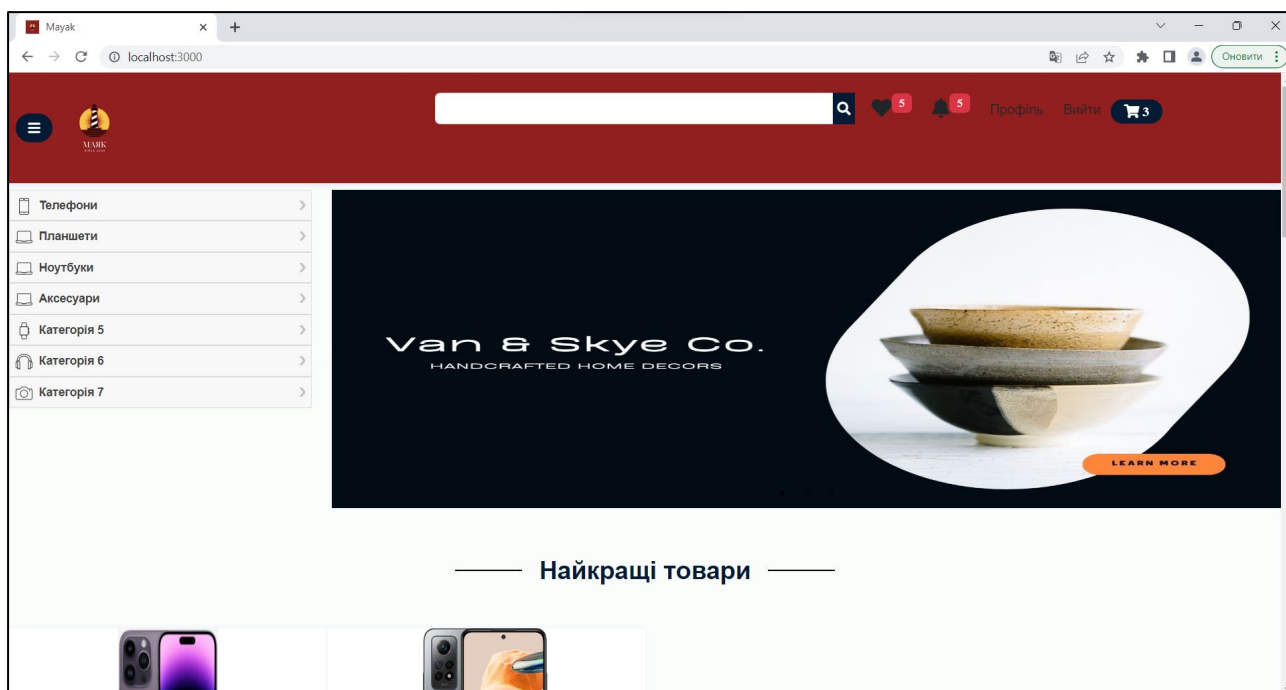


Рисунок 3.1 – Головна сторінка сайту

На головній сторінці сайту користувач може переглянути найкращі товари, новинки, популярні та товари.

Якщо користувачу потрібна конкретна модель телефону, то він може ввести назву товару в рядок пошуку, натиснути на кнопку пошуку та знайти обраний товар.

Також, користувач може знайти товар по категорії, або підкатегорії використовуючи меню з лівого боку екрана.

Для того, щоб користувачу створити новий акаунт він повинен натиснути на кнопку зареєструватися в правому верхньому куті екрану.



Рисунок 3.2 – Кнопка Реєстрації

Після цього користувач потрапляє на сторінку реєстрації де він зможе заповнити усі необхідні поля та створити власний акаунт, завдяки якому він

					ПІЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						41

зможє додавати товари у список бажаного, або робити замовлення в інтернет-магазині мобільних телефонів та аксесуарів.

Рисунок 3.3 – Сторінка реєстрації

Наступним кроком користувач повинен заповнити усі необхідні поля, а саме: Ім'я, email, пароль та повторно ввести пароль для підтвердження.

Після цього користувач потрапляє на сторінку власного профілю на які зазначено його ім'я та електрона адреса.

Рисунок 3.3 – Сторінка профілю користувача

Наступним кроком користувач може повернутися до головної сторінки сайту та переглянути потрібну йому інформацію.

Якщо користувач вже має акаунт, то ван може увійти в нього натиснувши на кнопку «Увійти», яка знаходиться в шапці сторінки. Після чого він попаде на сторінку входу.

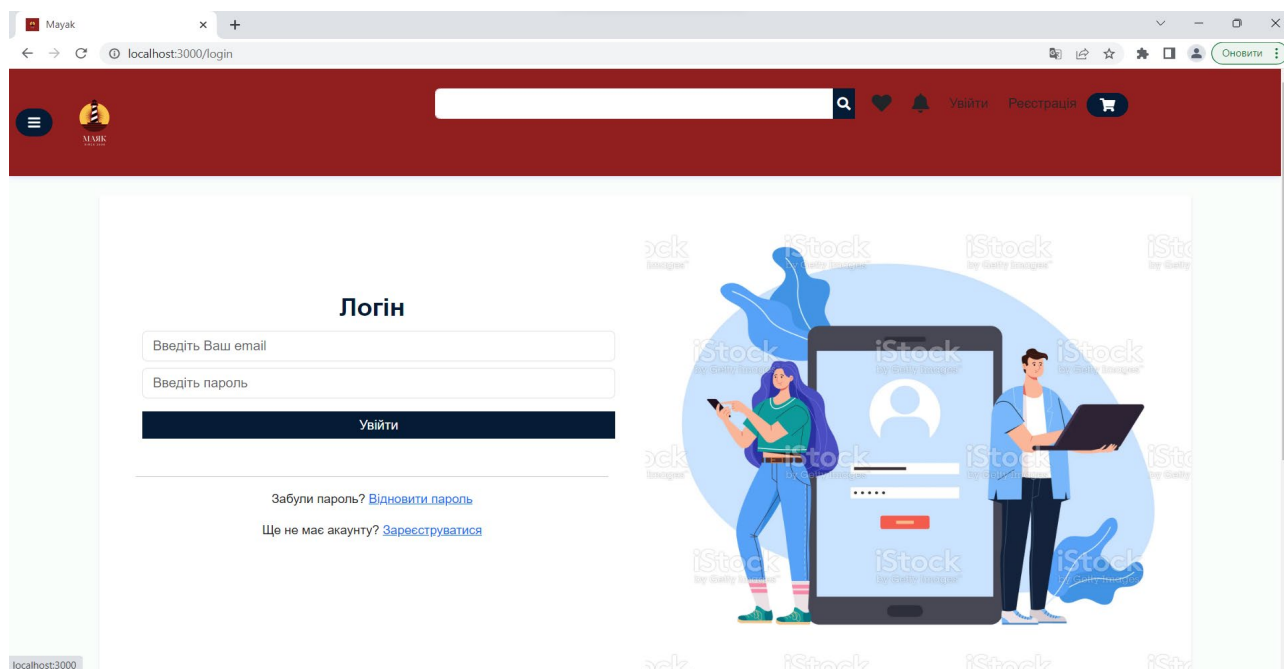


Рисунок 3.4 – Сторінку входу в акаунт

На цій сторінці користувач повинен ввести свою електронну пошту та пароль, після чого він попаде на сторінку свого профілю.

Якщо ж користувач не пам'ятає свій пароль він може його відновити. Для цього йому потрібно натиснути на кнопку «Відновити пароль».

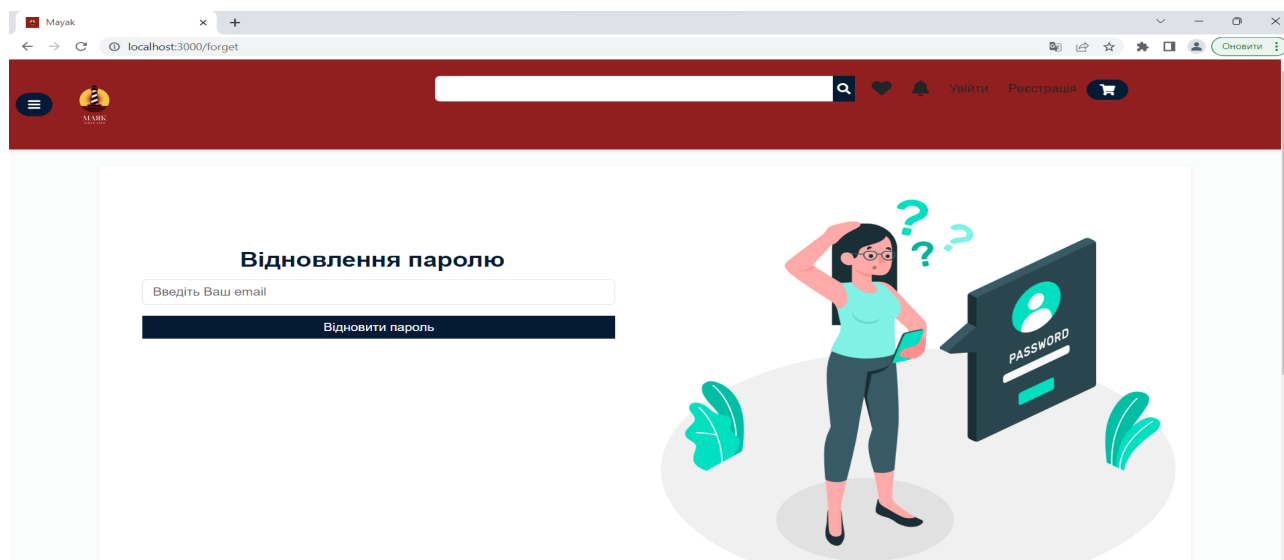


Рисунок 3.5 – Сторінка відновлення пароля

Наступним кроком користувачу потрібно буде ввести свою електронну пошту, на яку був зареєстрований акаунт і натиснути кнопку «Відновити пароль».

Коли користувач виконає попередні кроки йому на пошту прийде лист, де він побачить код для відновлення пароля та посилання, за яким він повинен буде перейти.

Але якщо користувач введе неправильну електронну адресу тоді йому виведеться повідомлення про те, що він ввів неправильну пошту.

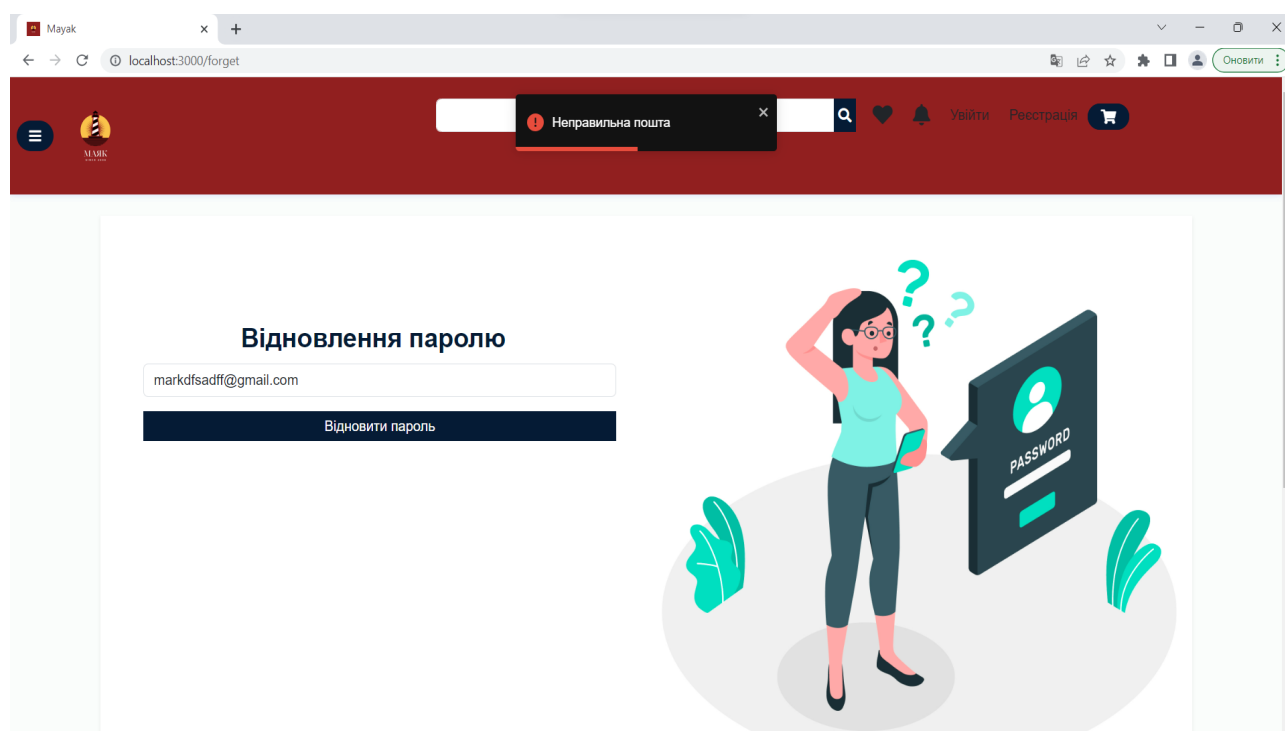


Рисунок 3.6 – Приклад введення неправильної пошти

Якщо електронна адреса користувача, яку він ввів буде вірною тоді він отримає повідомлення, яке буде відображене в верхній частині екрана, про те, що на його пошту успішно відправлено лист з кодом для відновлення пароля. Надалі йому потрібно буде використати цей код для відновлення пароля та для відновлення доступу на сайт. Якщо ж електронна пошта не співпадає ні з однією з тих, які містяться в базі даних тоді користувач не зможе відновити пароль і йому виведеться відповідне повідомлення про те, що такої адреси не існує, або що вона введена некоректно і йому потрібно буде перевірити правильність введених даних.

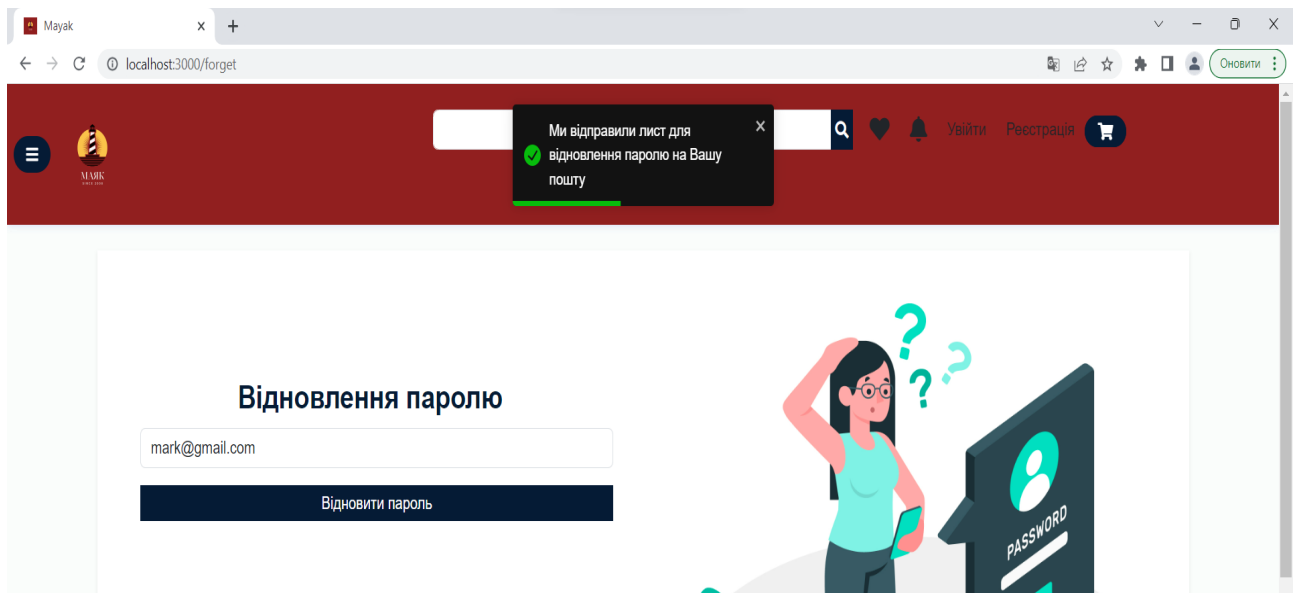


Рисунок 3.7 – Приклад введення правильного email

Для тестування коректності роботи цієї функції я використовую платформу mailtrap, яка є доволі зручною та простою у використанні, тому усі листи про відновлення пароля приходять туди.

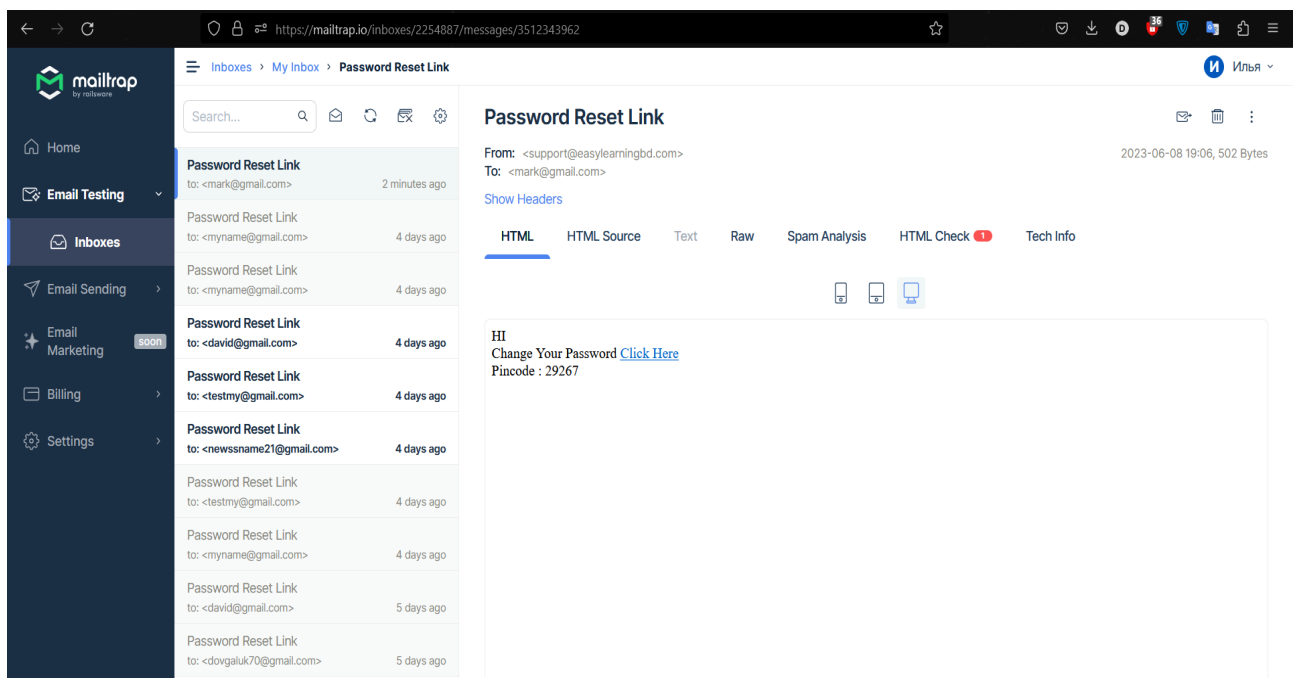


Рисунок 3.8 – Результат отримання коду для відновлення пароля

При переході за посиланням в листі користувач потрапляє на сторінку створення нового паролю, де йому теж потрібно буде заповнити форму вписавши туди усі необхідні дані. Всі поля форми є обов'язковими для заповнення.

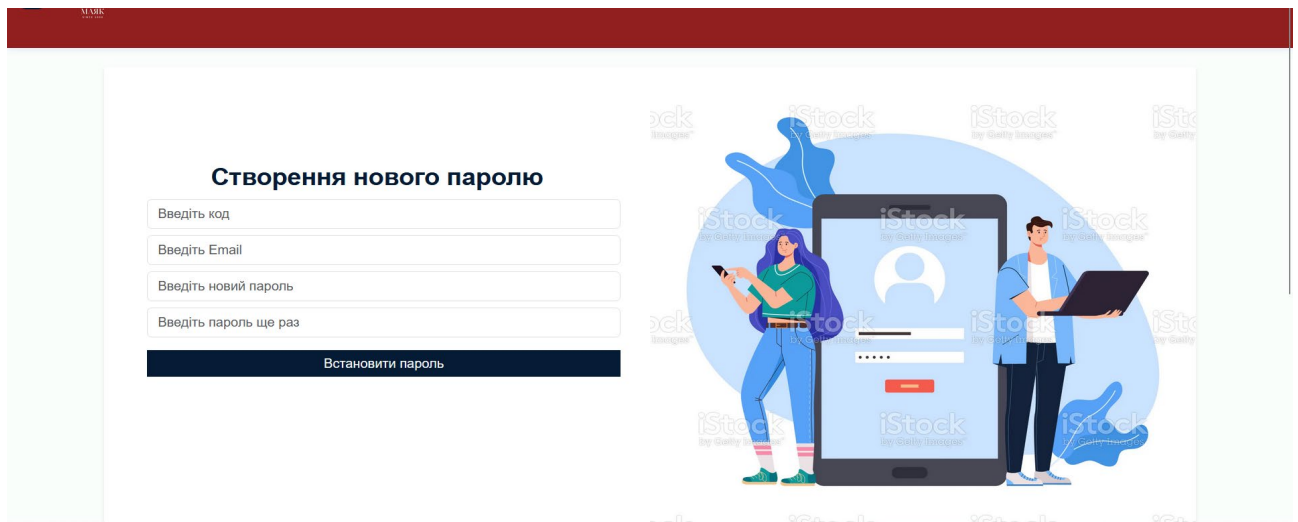


Рисунок 3.9 – Сторінка створення нового паролю

На цій сторінці користувач повинен ввести код, який він отримав в листі та заповнити усі поля, такі як:

- Код;
- Email;
- Пароль та повторити його ще раз.

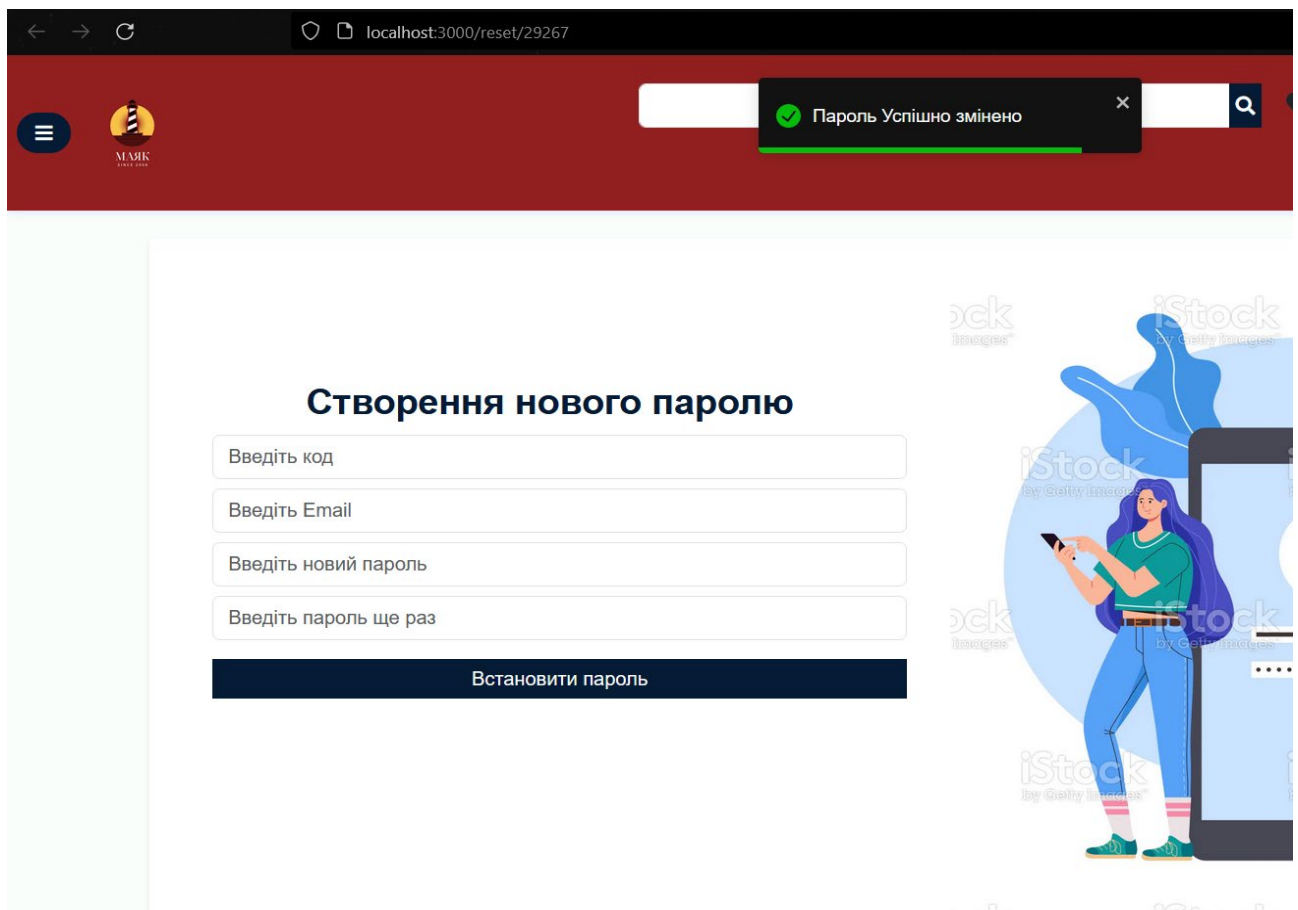


Рисунок 3.10 – Результат успішної зміни пароля

В самому низу сайту користувач може переглянути інформацію про адресу магазину та контакти, інформацію про співпрацю та про повернення/обмін товарів.

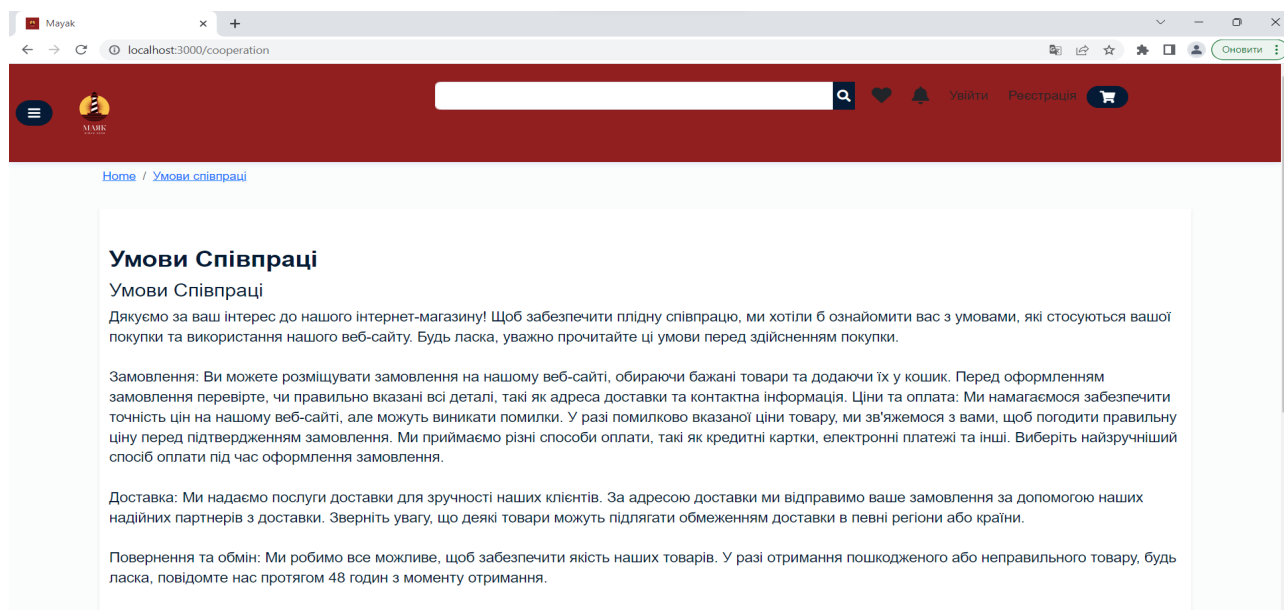


Рисунок 3.11 – Приклад сторінки з інформацією про умови співпраці

Якщо користувач перейде на сторінку «Як нас знайти», перехід на яку здійснюється в нижній частині сайту. Натиснувши на напис «Як нас знайти» він зможе побачити форму для зв'язку з нами, а також карту, на якій позначене місце розташування офлайн магазину.

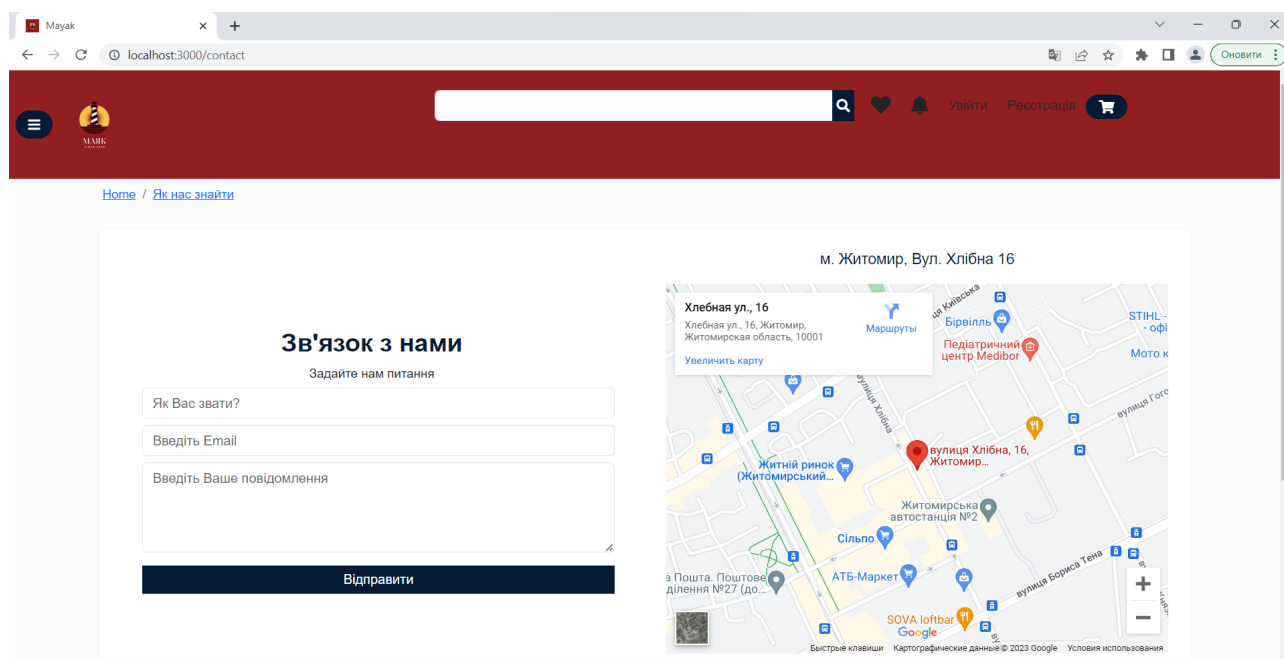


Рисунок 3.12 – Сторінка «Як нас знайти»

Якщо ж клієнт хоче зв'язатися з нами він повинен заповнити форму, яка знаходиться на цій сторінці. Користувачу необхідно заповнити усі поля у формі такі як:

- Ім'я;
- Email;
- Повідомлення.

В полі «Повідомлення» користувачу слід зазначити проблему, яку менеджер допоможемо йому вирішити, або задати питання, яке його цікавить. Усі поля у формі є обов'язковими для заповнення і якщо користувач не заповнить одне з полів він не зможе відправити своє повідомлення. Перевірити це можна натиснувши на кнопку «Відправити». Якщо одне з полів буде порожнім, то користувач отримає повідомлення про те, яке саме поле йому потрібно заповнити.

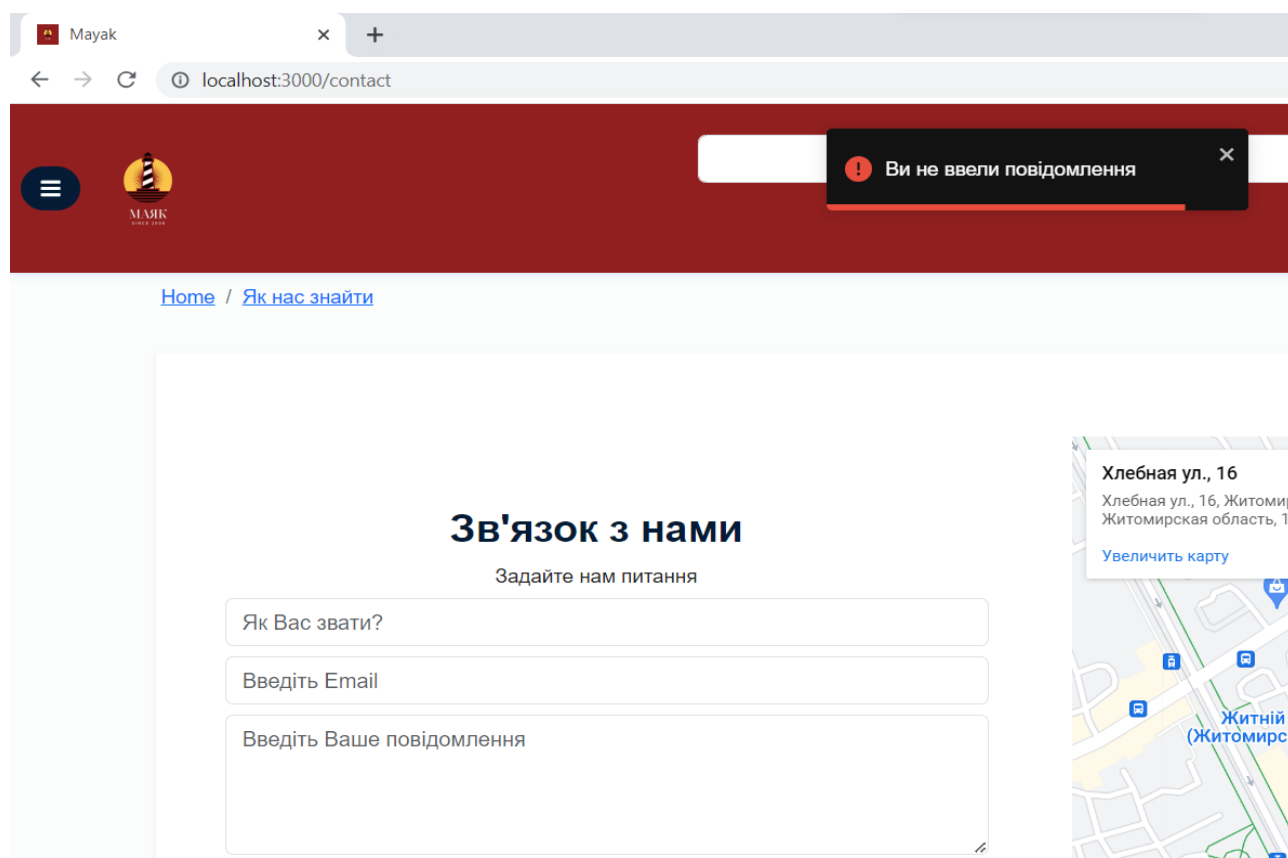


Рисунок 3.13 – Виведення помилки при спробі відправлення пустої форми

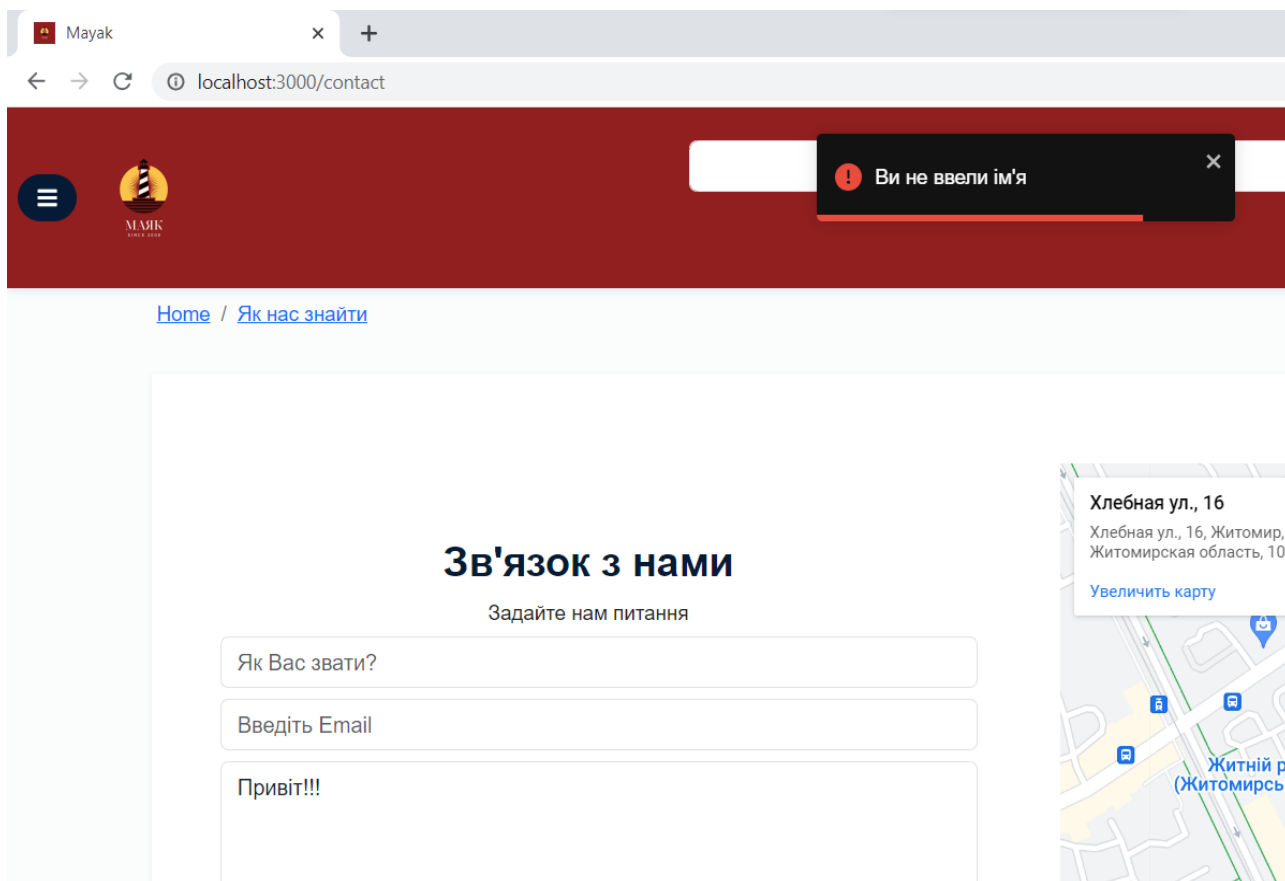


Рисунок 3.14 – Результат виведення помилки при спробі відправлення форми без заповнення поля «Ім'я»

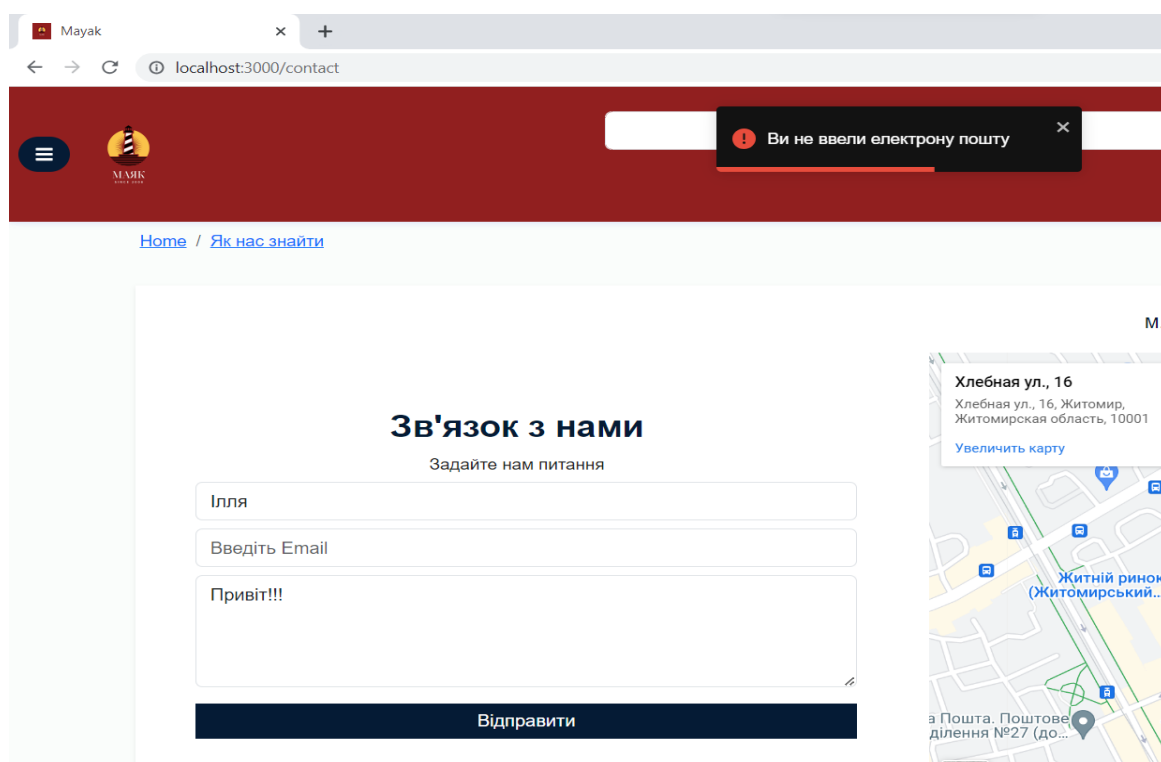


Рисунок 3.15 – Результат виведення помилки при відправці форми з незаповненим полем «email»

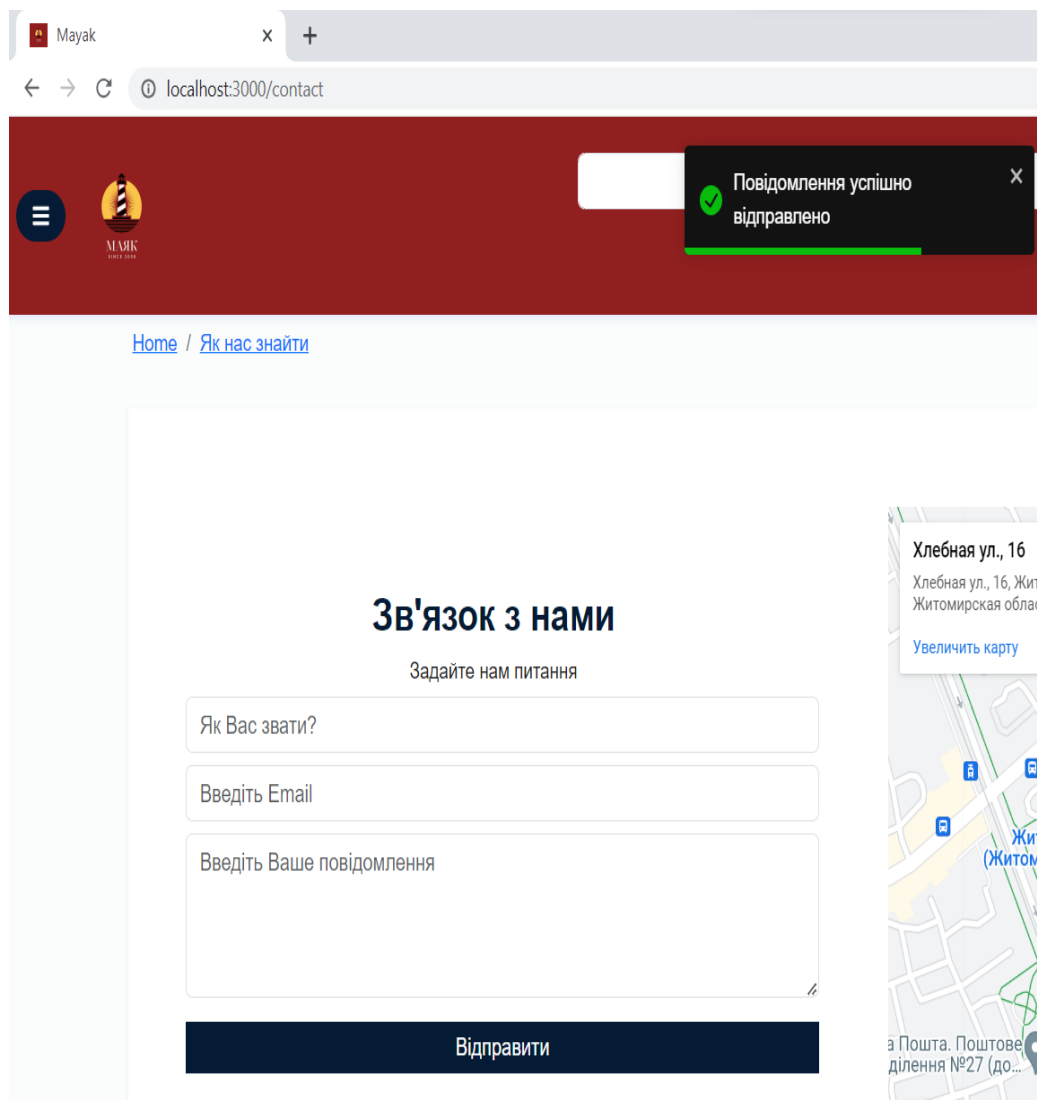


Рисунок 3.16 – Результат успішної відправки форми

Після того, як користувач відправив заповнену форму, в верхній частині екрану він може побачити спливаюче повідомлення зеленого кольору, про те, що його повідомлення успішно відправлено. Після цього це повідомлення буде опрацьоване менеджером. Якщо буде потреба то менеджер відправить зворотне повідомлення з відповіддю, або з уточненням певної інформації, на електронну пошту користувача, яка була зазначена у формі.

Наступним кроком користувач може переглянути детальну інформацію про товар, який йому сподобався. Для цього користувачу достатньо натиснути на нього. І тоді він потрапить на сторінку з детальною інформацією про обраний ним товар, зможе переглянути його фото, характеристики товару, детальний опис та відгуки інших користувачів.

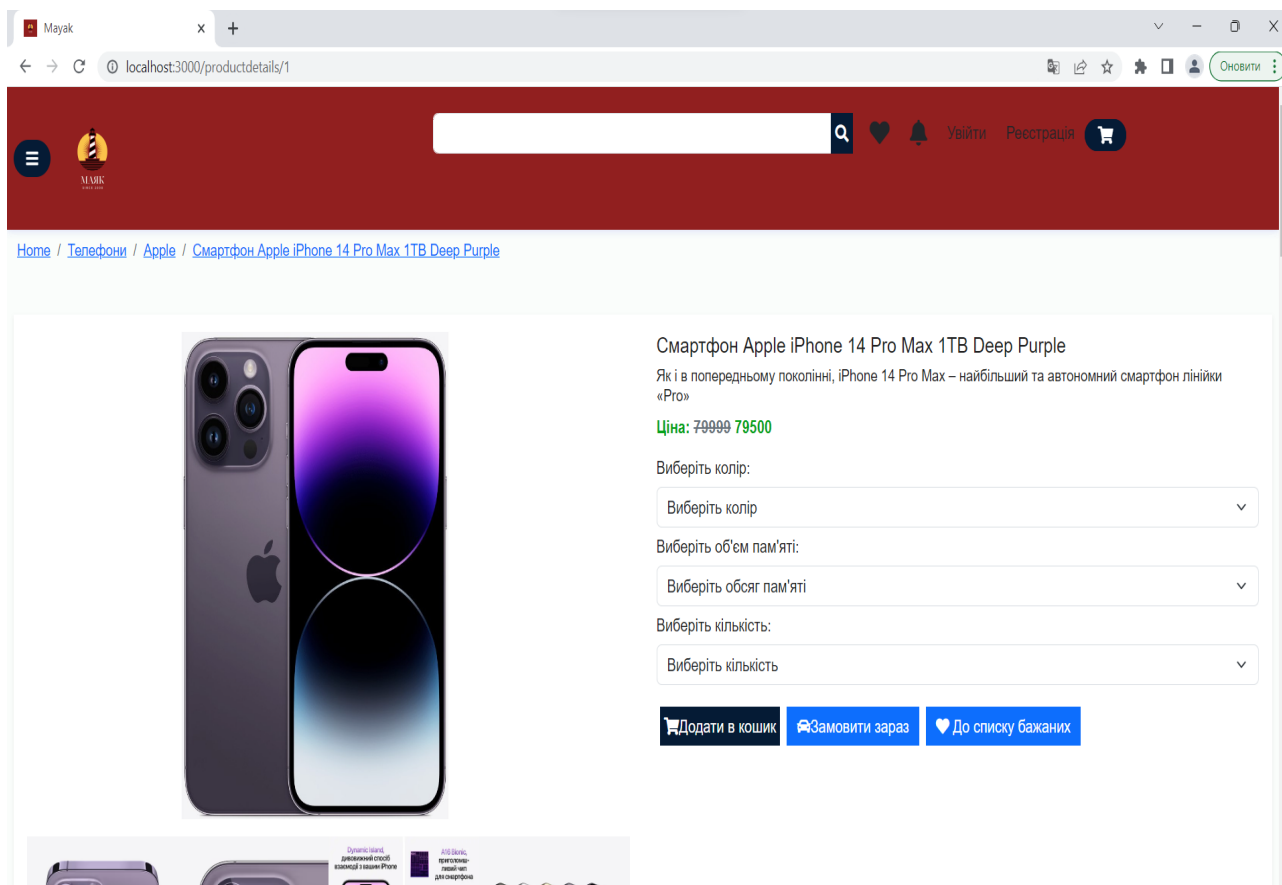


Рисунок 3.17 – Перегляд сторінки з детальною інформацією про товар

На цій сторінці користувач може переглянути фото товару, його детальний опис, ціну товару, детальні характеристики та схожі товари. Крім цього, він може додати цей товар в кошик. Для цього користувач повинен натиснути на кнопку «Додати до кошика», якщо ж потрібно додати товар до списку бажаного потрібно натиснути на кнопку «До списку бажаних» і тоді товар буде відображатися у списку бажаних товарів. Також, товар можна відразу замовити. Для виконання цієї дії необхідно натиснути на кнопку «Замовити зараз», яка знаходиться відразу під полем вибору кількості замовлення обраного товару, після цього користувач відразу потрапить на сторінку корзини, де йому потрібно буде заповнити форму, в якій вказати дані про себе (своє ім'я), місто доставки з випадаючого переліку, спосіб оплати (передоплата, або оплата після отримання), адресу доставлення. Після цього користувачу потрібно буде чекати на дзвінок менеджера для перевірки правильності вказаних даних та підтвердження замовлення і вже потім товар буде відправлено замовнику.

Характеристики

Бренд:	Apple
Лінійка:	Смартфон
ОСНАЩЕННЯ	
Оперативна пам'ять:	8
Вбудована пам'ять:	256Gb,512Gb,1Tb
Слот розширення:	
Тип SIM-карти:	nano-SIM
Процесор:	Apple A16 Bionic
Кількість ядер:	6
Частота GHz:	frequency
Швидка зарядка, Вт:	37W

Рисунок 3.18 – Фрагмент таблиці з детальною інформацією про товар.

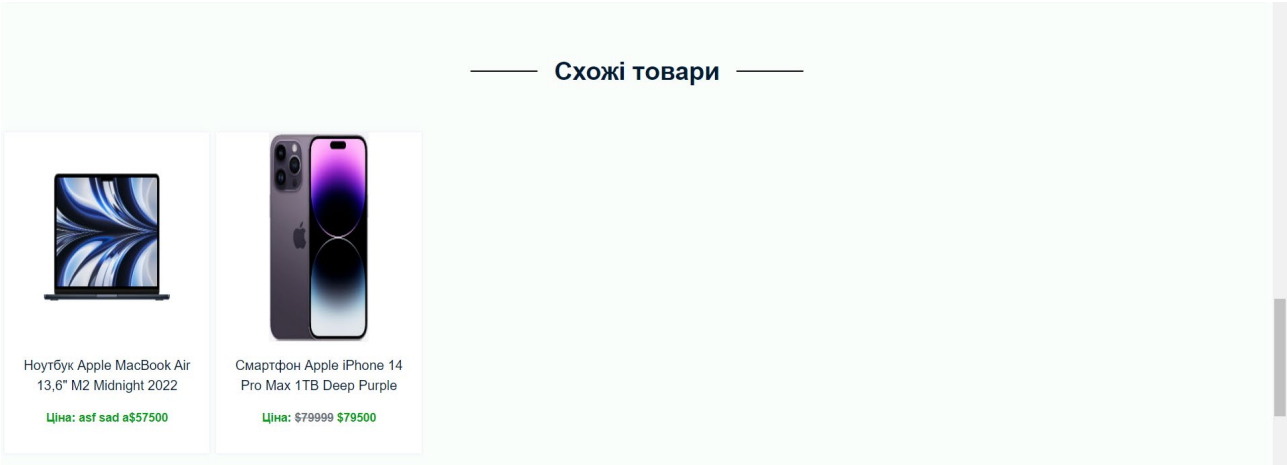


Рисунок 3.19 – Перегляд схожих товарів.

Якщо користувач хоче додати товар до списку бажаних він повинен натиснути на кнопку «До списку бажаних», після цього він отримає повідомлення про те, що товар успішно додано до списку бажаних товарів, але для того, щоб це зробити користувач повинен бути авторизованим.

Якщо користувач натисне на кнопку додати в кошик він отримає повідомлення про те, що спочатку потрібно вибрати колір, обсяг пам'яті, та кількість, але навіть заповнивши ці поля користувач не зможе зробити

замовлення, йому виведеться повідомлення про те, що він повинен спочатку авторизуватися.

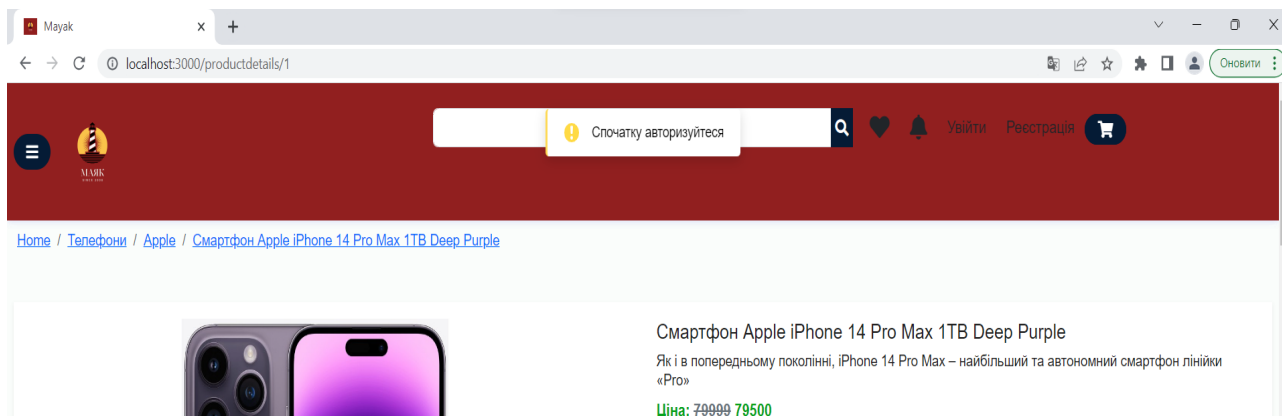


Рисунок 3.20 – Виведення попередження про те, що для додавання товару в кошик, користувач повинен бути авторизованим

Після того, як користувач додасть товар в кошик в центральній частині екрану він зможе побачити повідомлення про успішне додавання товару до кошика.

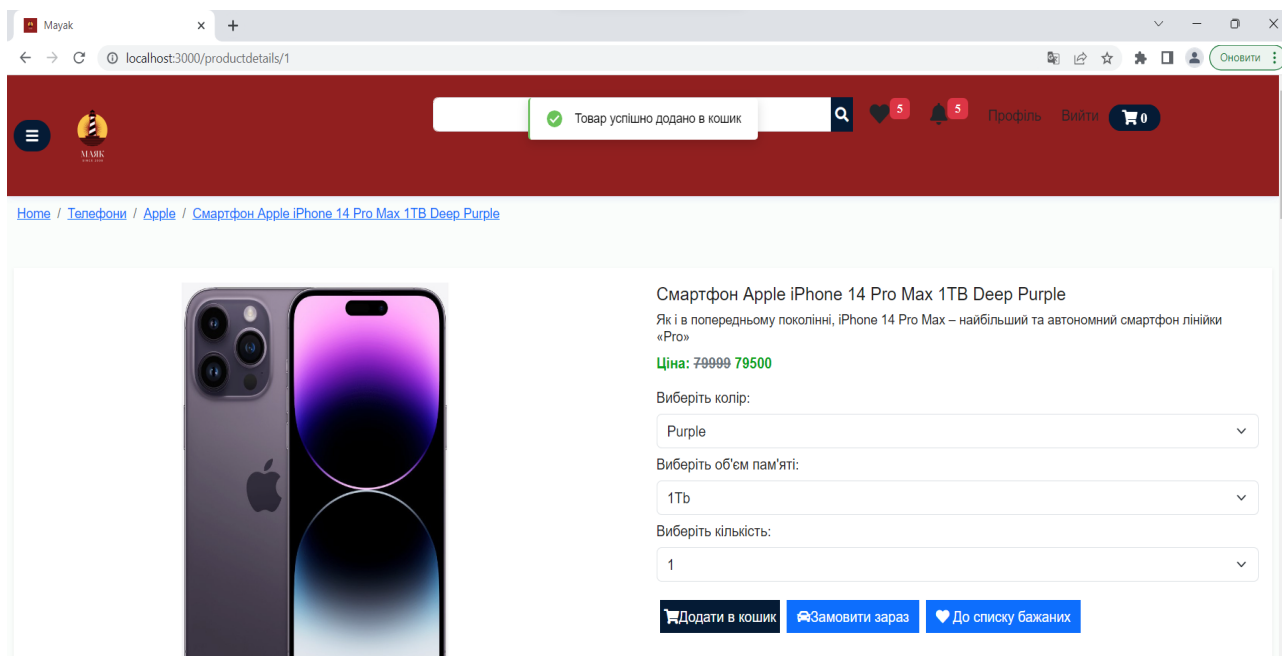


Рисунок 3.21 – Результат успішного додавання товару в кошик

Аналогічно відбудеться і з додаванням товару в список бажаного.

Якщо ж клієнт натисне на кнопку «Замовити зараз» його відразу перекине в кошик, де він відразу зможе побачити форму з інформацією про оформлення та відправку замовлення.

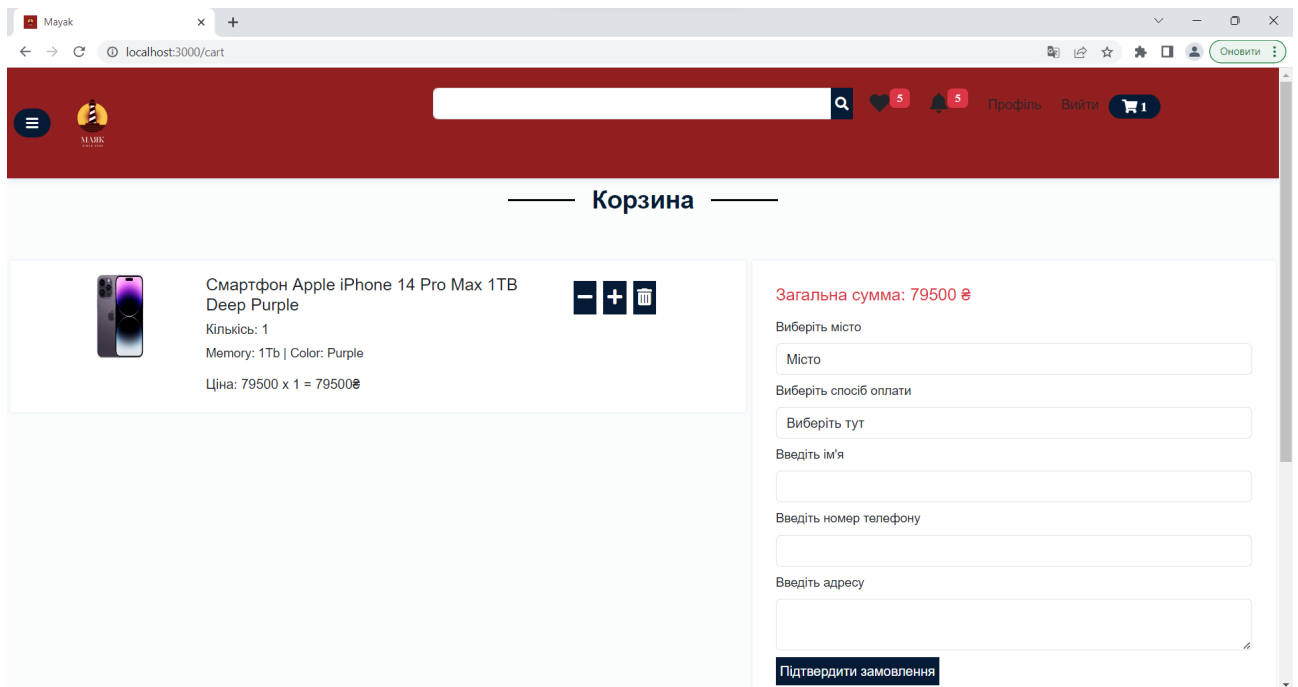


Рисунок 3.22 – Перегляд кошика з доданим товаром в попередньому кроці.

Далі користувач може заповнити форму, для того, щоб відправити замовлення на опрацювання. Коли він заповнить усі поля та натисне на кнопку «Підтвердити замовлення» він отримає повідомлення про те, що замовлення успішно прийняте.

ВИСНОВКИ ДО РОЗДІЛУ 3:

В третьому розділі я розглянув інтерфейс реалізованого інтернет-магазину мобільних телефонів та аксесуарів. Більше того, я розглянув сторінки, які є на сайті, та які може використовувати користувач. Також, я розглянув інтерфейс, з яким буде взаємодіяти користувач, перевірив його зручність та простоту у використанні.

Основним, що я розглянув в цьому розділі є тестування функціоналу сайту. Пройдено процес реєстрації заповнивши відповідну форму тестовими даними, після чого я вийшов з профілю та спробував відновити пароль, що в мене успішно вийшло і я зміг авторизуватися на сайті. Перевірено, як працює функціонал додавання товару у кошик, списку бажаного та перевірив роботу функції «замовити зараз». В кінці я заповнив форму для оформлення замовлення.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						54

ВИСНОВКИ

В ході виконання кваліфікаційної роботи я провів аналіз методів та засобів розробки інтернет-магазину. Я провів аналіз схожих ресурсів, які є одними з найпопулярніших в Україні. Це дало мені змогу визначити вимоги до мого веб-додатку, а саме те, що він має бути функціональним, зручним та простим у використанні, та мати привабливий дизайн.

Проаналізувавши сучасні вимоги до e-commerce проєктів я обрав клієнт-серверну архітектуру для розробки сайту, що дало змогу зручно реалізовувати цей проєкт.

Ми проаналізували функціональні, системні та нефункціональні вимоги. Додатково я створив діаграму варіантів використання сайту, щоб зрозуміти яким чином користувачі будуть взаємодіяти з сайтом.

Детально розглянуто процес створення веб-додатку, який загальний принцип його роботи та яким чином працюють компоненти нашого ПЗ. Додатково я розібрався втому, як взаємодіють в проєкті React, який відповідає за Frontend частину та Laravel, який відповідає за Backend частину.

Під час виконання кваліфікаційної роботи бакалавра я створив повноцінний інтернет-магазин для продажу мобільних телефонів та аксесуарів. Надалі цей проєкт можна буде покращити та налаштувати під конкретніші потреби. Це дасть змогу розвинути власний офлайн магазин, збільшити обсяги продажів, впізнаваність бренду та вийти на рівень великих інтернет-магазинів.

Також, виконання кваліфікаційної роботи дало нам змогу використати наші теоретичні знання на практиці набуті за 4 роки навчання в університеті. Це дасть нам змогу для подальшого розвитку в IT-сфері.

Реалізація e-commerce проєкту є важливим етапом у професійній та освітній підготовці. Це дало нам змогу глибше зрозуміти процес розробки інтернет-магазинів, та здобути цінний досвід, який надалі знадобиться нам в професійній діяльності.

					ІПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						55

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Laravel documentation [Електронний ресурс] – режим доступу до ресурсу:
<https://laravel.com/docs/10.x>
2. Laravel passport documentation [Електронний ресурс] – режим доступу до ресурсу:
<https://laravel.com/docs/10.x/passport>
3. React documentation [Електронний ресурс] – режим доступу до ресурсу:
<https://legacy.reactjs.org/docs/getting-started.html>
4. useNavigate [Електронний ресурс] – режим доступу до ресурсу:
<https://reactrouter.com/en/main/hooks/use-navigate>
5. PHP documentation [Електронний ресурс] – режим доступу до ресурсу:
<https://www.php.net/manual/en/function.date-default-timezone-set.php>
6. Клієнт-серверна архітектура та ролі серверів [Електронний ресурс] – режим доступу до ресурсу:
<http://surl.li/diqv>
7. React modals [Електронний ресурс] – режим доступу до ресурсу:
<https://react-bootstrap.netlify.app/docs/components/modal/>
8. React-toastify [Електронний ресурс] – режим доступу до ресурсу:
<https://fkhadra.github.io/react-toastify/introduction/>
9. Cogo-toast [Електронний ресурс] – режим доступу до ресурсу:
<https://www.npmjs.com/package/cogo-toast>
10. Laravel Jetstream [Електронний ресурс] – режим доступу до ресурсу:
<https://jetstream.laravel.com/3.x/installation.html>
11. Axios [Електронний ресурс] – режим доступу до ресурсу:
<https://www.npmjs.com/package/axios>

ДОДАТКИ

					ІІЗ.КР.Б - 121 - 23 - ІІЗ	Арк.
						57

CategoryController.php

```

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\Category;
use App\Models\SubCategory;
class CategoryController extends Controller
{
    public function AllCategory() {
        $categories = Category::all();
        $categoryDetailsArray = [];
        foreach ($categories as $value){
            $subcategory = SubCategory::where('category_name', $value['category_name'])->get();
            $item = [
                'category_name' => $value['category_name'],
                'category_image' => $value['category_image'],
                'subcategory_name' => $subcategory
            ];
            array_push($categoryDetailsArray, $item);
        }
        return $categoryDetailsArray;
    }
}

```

ContactController.php

```

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;use App\Models>Contact;

```

					ПЗ.КР.Б - 121 - 23 - ПЗ	Арк.
						58

```

class ContactController extends Controller
{
    public function PostContactDetails(Request $request){
        $name = $request->input('name');
        $email = $request->input('email');
        $message= $request->input('message');
        date_default_timezone_set("Europe/Kiev");
        $contact_time = date("h:i:sa");
        $contact_date = date("d-m-Y");
        $result = Contact::insert([
            'name' => $name,
            'email' => $email,
            'message' => $message,
            'contact_date' => $contact_date,
            'contact_time' => $contact_time,
        ]);
        return $result;
    }
}

```

FavouriteController.php

```

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\ProductList;
use App\Models\Favourites;
class FavouriteController extends Controller
{
    public function AddFavourite(Request $request){
        $product_code = $request->product_code;
        $email = $request->email;    $productDetails =
        ProductList::where('product_code',$product_code)->get();
    }
}

```

```

$result = Favourites::insert([
    'product_name' => $productDetails[0]['title'],
    'image' => $productDetails[0]['image'],
    'product_code' => $product_code,
    'email' => $email,
]);    return $result}

public function FavouriteList(Request $request){
    $email = $request->email;
    $result = Favourites::where('email', $email)->get();
    return $result;

public function FavouriteRemove(Request $request){
    $product_code = $request->product_code;
    $email = $request->email;
    $result = Favourites::where('product_code', $product_code)->where('email', $email)->get();
    return $result;
}}

```

NotificationCotroller.php

```

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\Notification;
class NotificationController extends Controller
{
    public function NotificationHistory (){
        $result = Notification::get();
        return $result;
    }
}

```

ProductCartController.php

```

<?phpnamespace App\Http\Controllers\Admin;

```

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						60

```

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\ProductCart;

use App\Models\ProductList;

use App\Models\CartOrder;

class ProductCartController extends Controller
{
    public function addToCart(Request $request){
        $email = $request->input('email');
        $memory = $request->input('memory');
        $color = $request->input('color');
        $quantity = $request->input('quantity');
        $product_code = $request->input('product_code');
        $productDetails = ProductList::where('product_code',$product_code)->get();
        $price = $productDetails[0]['price'];
        $special_price = $productDetails[0]['special_price'];
        if($special_price=="na"){
            $total_price = $price*$quantity;
            $unit_price = $price;
        }
        else{
            $total_price = $special_price*$quantity;
            $unit_price = $special_price }

        $result = ProductCart::insert([
            'email' => $email,
            'image' => $productDetails[0]['image'],
            'product_name' => $productDetails[0]['title'],
            'product_code' => $productDetails[0]['product_code'],
            'memory' => "Memory: ".$memory,

```

					ИПЗ.КР.Б - 121 - 23 - ИПЗ	Арк.
						61

```

        'color' => "Color: ".$color,
        'quantity' => $quantity,
        'unit_price' => $unit_price,
        'total_price' => $total_price,
    ]);
    return $result;
}

public function CartCount(Request $request){
    $product_code = $request->product_code;
    $result = ProductCart::count();
    return $result;
}

public function CartList(Request $request){
    $email = $request->email;
    $result = ProductCart::where('email',$email)->get();
    return $result;
}

public function RemoveCartList(Request $request){
    $id = $request->id;
    $result = ProductCart::where('id',$id)->delete();
    return $result;
}

public function CartItemPlus(Request $request){
    $id = $request->id;

    $quantity = $request->quantity;
    $price = $request->price;
    $newQuantity = $quantity + 1;
    $total_price = $newQuantity*$price;

    $result = ProductCart::where('id', $id)->update(['quantity' => $newQuantity, 'total_price' =>
    $total_price]);    return $result;

```

					III3.KP.Б - 121 - 23 - II3	Арк.
						62

```

}

public function CartItemMinus(Request $request){
    $id = $request->id;
    $quantity = $request->quantity;
    $price = $request->price;
    $newQuantity = $quantity - 1;
    $total_price = $newQuantity*$price;

    $result = ProductCart::where('id', $id)->update(['quantity' => $newQuantity, 'total_price' =>
$total_price]);

    return $result;
}

public function CartOrder(Request $request){
    $city = $request->input('city');
    $paymentMethod = $request->input('payment_method');
    $Phone = $request->input('phone');
    $yourName = $request->input('name');
    $email = $request->input('email');
    $DeliveryAddress = $request->input('delivery_address');
    $invoice_no = $request->input('invoice_no');
    $DeliveryCharge = $request->input('delivery_charge');
    date_default_timezone_set("Europe/Kiev");
    $request_time = date("h:i:sa");
    $request_date = date("d-m-Y");
    $CartList = ProductCart::where('email',$email)->get();

    foreach($CartList as $CartItem){
        $cartInsertDeleteResult = "";
        $resultInsert = CartOrder::insert([
            'invoice_no' => "Mayak".$invoice_no,
            'product_name' => $CartItem['product_name'],
            'product_code' => $CartItem['product_code'],

```

					ІІІ3.КР.Б - 121 - 23 - ІІ3	Арк.
						63

```

        'memory' => $CartItem['memory'],
        'color' => $CartItem['color'],
        'quantity' => $CartItem['quantity'],
        'unit_price' => $CartItem['unit_price'],
        'total_price' => $CartItem['total_price'],
        'email' => $CartItem['email'],
        'name' => $yourName,
        'phone' => $Phone,
        'payment_method' => $paymentMethod,
        'delivery_address' => $DeliveryAddress,
        'city' => $city,
        'delivery_charge' => $DeliveryCharge,
        'order_date' => $request_date,
        'order_time' => $request_time,
        'order_status' => "Pending",
    ];
    if($resultInsert == 1){
        $resultDelete = ProductCart::where('id', $CartItem['id'])->delete();
        if($resultDelete == 1){
            $cartInsertDeleteResult = 1;
        }
        else{
            $cartInsertDeleteResult = 0;
        }
    }
    return $cartInsertDeleteResult;
}
}

```

ProductDetailsController.php

<?php

| | | | | | | |
|--|--|--|--|--|---------------------------|------|
| | | | | | ИПЗ.КР.Б - 121 - 23 - ИПЗ | Арк. |
| | | | | | | 64 |
| | | | | | | |

```

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\ProductList;
use App\Models\ProductDetails;

class ProductDetailsController extends Controller
{
    public function ProductDetails(Request $request){
        $id = $request->id;

        $productDetails = ProductDetails::where('product_id', $id)->get();
        $productList = ProductList::where('id',$id)->get();

        $item = [
            'prodductDetails' => $productDetails,
            'productList' => $productList,
        ];

        return $item;
    }
}

```

ProductListController.php

```

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\ProductList;

class ProductListController extends Controller
{
    public function ProductListByRemark(Request $request){
        $remark = $request->remark;

        $productlist = ProductList::where('remark', $remark)->limit(8)->get();

        return $productlist;
    }
}

```

| | | | | | | |
|--|--|--|--|--|----------------------------|------|
| | | | | | III3.KP.Б - 121 - 23 - II3 | Арк. |
| | | | | | | 65 |
| | | | | | | |

```

    }

    public function ProductListByCategory(Request $request){
        $Category = $request->category;

        $productlist = ProductList::where('category', $Category)->get();

        return $productlist;
    }

    public function ProductListBySubCategory(Request $request){
        $Category = $request->category;

        $SubCategory = $request->subcategory;

        $productlist = ProductList::where('category', $Category)->where('subcategory',
        $SubCategory)->get();

        return $productlist;
    }

    public function ProductBySearch(Request $request) {
        $key = $request->key;

        $productlist = ProductList::where('title', 'LIKE', "%{$key}%")->orWhere('brand', 'LIKE',
        "%{$key}%")->orWhere('category', 'LIKE', "%{$key}%")->get();

        return $productlist;
    }

    public function SimilarProduct(Request $request) {
        $subcategory = $request->subcategory;

        $productlist = ProductList::where('subcategory', $subcategory)->orderBy('id', 'desc')->limit(6)-
        >get();

        return $productlist;
    }
}

```

ReviewController.php

```

<?php

namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;

use Illuminate\Http\Request;

use App\Models\ProductReview;

```

| | | | | | | |
|--|--|--|--|--|---------------------------|------|
| | | | | | ИПЗ.КР.Б - 121 - 23 - ИПЗ | Арк. |
| | | | | | | 66 |
| | | | | | | |

```

class ReviewController extends Controller
{
    public function ReviewList(Request $request){
        $id = $request->id;
        $result = ProductReview::where('product_id', $id)->orderBy('id', 'desc')->limit(4)->get();
        return $result;
    }
}

```

SiteInfoController.php

```

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\SiteInfo;
class SiteInfoController extends Controller
{
    public function AllSiteinfo(){
        $result = SiteInfo::get();
        return $result;
    }
}

```

SliderController.php

```

<?php
namespace App\Http\Controllers\Admin;

use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\HomeSlider;
class SliderController extends Controller
{
    public function AllSlider(){

```

| | | | | | | |
|--|--|--|--|--|---------------------------|------|
| | | | | | ИПЗ.КР.Б - 121 - 23 - ИПЗ | Арк. |
| | | | | | | 67 |
| | | | | | | |

```

        $result = HomeSlider::all();

        return $result;
    }
}

VisitorController.php

<?php
namespace App\Http\Controllers\Admin;
use App\Http\Controllers\Controller;
use Illuminate\Http\Request;
use App\Models\Visitor;
class VisitorController extends Controller
{
    public function GetVisitorDetails(){
        $ip_address = $_SERVER['REMOTE_ADDR'];
        date_default_timezone_set("Europe/Kiev");
        $visit_time = date("h:i:sa");
        $visit_date = date("d-m-Y");
        $result = Visitor::insert([

            'ip_address' => $ip_address,
            'visit_time' => $visit_time,
            'visit_date' => $visit_date
        ]);
        return $result;
    }
}

```

Create_product_details_table.php:

```

<?php
use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

```

| | | | | | | |
|--|--|--|--|--|----------------------------|------|
| | | | | | ІІІ3.КР.Б - 121 - 23 - ІІ3 | Арк. |
| | | | | | | 68 |
| | | | | | | |

```

return new class extends Migration
{
    /**
     * Run the migrations.
     */
    public function up(): void
    {
        Schema::create('product_details', function (Blueprint $table) {
            $table->id();
            $table->integer('product_id');
            $table->string('image_one');
            $table->string('image_two');
            $table->string('image_three');
            $table->string('image_four');
            $table->string('short_description');
            $table->string('color');
            $table->string('memory');
            $table->text('long_description');
            $table->string('product_brand');
            $table->string('product_type');
            $table->string('product_ram');
            $table->string('builtin_memory');
            $table->string('slot_expansion');
            $table->string('simcard_type');
            $table->string('product_processor');
            $table->string('product_number_of_cores');
            $table->string('product_frequency');
            $table->string('product_fast_charge');
            $table->string('product_battery');
            $table->string('product_diagonal');
        });
    }
}

```

```

$table->string('product_resolution');

$table->string('product_matrix_type');

$table->string('product_refresh_rate');

$table->string('product_display_features');

$table->string('product_brightness');

$table->string('product_main_camera');

$table->string('product_video');

$table->string('product_front_camera');

$table->string('product_communication_standards');

$table->string('product_wifi');

$table->string('product_gps');

$table->string('product_nfc');

$table->string('product_interface_conector');

$table->string('product_height');

$table->string('product_width');

$table->string('product_weight');

$table->string('product_protection');

$table->timestamps();

});

}

/**

 * Reverse the migrations.

 */
public function down(): void

{
    Schema::dropIfExists('product_details');

});

```