МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет
імені Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

ДОПУЩЕНО ДО ЗАХИСТУ
Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____Ю. П. Кондратенко
«____»_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

ВЕБЗАСТОСУНОК З ПРОДАЖУ ОДЯГУ

Спеціальність 122 «Комп'ютерні науки»

122 – КРБ – 402.22010223

Виконав студент 4-го курсу, групи 402
_____ О. М. Скиба
«17» червня 2024 р.

Керівник: ст. викладач каф. ІПЗ
_____М. В. Фаленкова
«17» червня 2024 р.

Миколаїв – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Чорноморський національний університет ім. Петра Могили
Факультет комп'ютерних наук
Кафедра інтелектуальних інформаційних систем

Рівень вищої освіти **бакалавр**
Спеціальність **122 «Комп'ютерні науки»**
(шифр і назва)
Галузь знань **12 «Інформаційні технології»**
(шифр і назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри інтелектуальних інформаційних систем, д-р техн. наук, проф.
_____ Ю. П. Кондратенко
«____» _____2024 р.

З А В Д А Н Н Я
на виконання кваліфікаційної роботи

Видано студенту групи 402 факультету комп'ютерних наук Скибі Олегу Миколайовичу.

1. Тема кваліфікаційної роботи «Вебзастосунок з продажу одягу».

Керівник роботи Фаленкова Марина Володимирівна, ст. викладач каф. ІПЗ.

Затв. наказом Ректора ЧНУ ім. Петра Могили від «28» грудня 2023 р. № 271

2. Строк представлення кваліфікаційної роботи студентом «17» червня 2024 р.

3. Вхідні (початкові) дані до роботи: експертні оцінки технологій та методів, які сприяють успішному впровадженню та ефективному функціонуванню веб-застосунку з продажу одягу.

Очікуваний результат: вебзастосунок з продажу одягу.

4. Перелік питань, що підлягають розробці (зміст пояснювальної записки):

- аналіз сучасного стану розвитку електронної комерції в обраній сфері;
- огляд існуючих технологій та методів успішного впровадження та ефективному функціонуванню веб-застосунку;

– експертне оцінювання технологій для ефективного розвитку комерційних проектів;

– порівняльний аналіз результатів застосування обраних методів багатокритерійного прийняття рішень для розв’язання поставленої задачі.

5. Перелік графічного матеріалу: 65 рисунків, 3 таблиці та презентація.

6. Завдання до спеціальної частини: «Оцінка умов праці фахівців з ІТ-технологій у робочому приміщенні».

7. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис
Спеціальна частина з охорони праці	Алексєєва А. О., доцент кафедри екології	

Керівник роботи __ст. викладач каф. ІПЗ, Фаленкова М. В._____
(наук. ступінь, вчене звання, прізвище та ініціали)

_____
(підпис)

Завдання прийнято до виконання ___Скиба О. М._____
(прізвище та ініціали)

_____
(підпис)

Дата видачі завдання «_29_» _____січня____ 2024 р.

КАЛЕНДАРНИЙ ПЛАН
виконання кваліфікаційної роботи

Тема: Вебзастосунок з продажу одягу

№	Найменування роботи	Початок	Закінчення	Примітки
1	Подання заяви на затвердження теми та керівників КРБ	10.11.2023	15.11.2023	Виконано
2	Отримання завдання на виконання КРБ	10.01.2024	15.01.2024	Виконано
3	Складання календарного плану роботи на весь період виконання КРБ	16.01.2024	30.01.2024	Виконано
4	Отримання завдання на переддипломну практику	15.04.2024	29.04.2024	Виконано
5	Проходження переддипломної практики, збір та аналіз матеріалів до КРБ	29.04.2024	11.05.2024	Виконано
6	Розробка звіту з переддипломної практики	12.05.2024	15.05.2024	Виконано
7	Виконання КРБ: аналіз сучасного стану електронної комерції в сфері продажу одягу, огляд існуючих технологій, розробка ПЗ	13.05.2024	22.06.2024	Виконано
8	Перший попередній захист КРБ на засіданні комісії кафедри	27.05.2024	27.05.2024	Виконано
9	Доробка та остаточне оформлення КРБ	28.05.2024	09.06.2024	Виконано
10	Другий попередній захист КРБ на засіданні комісії кафедри	10.06.2024	10.06.2024	Виконано
11	Подання КРБ рецензенту	13.06.2024	13.06.2024	Виконано
11	Подання КРБ, її електронної копії та інших документів (відгуку, рецензії) до захисту	17.06.2024	21.06.2024	Виконано
12	Захист БКР перед екзаменаційною комісією (ЕК)	24.06.2024	24.06.2024	Виконано

Розробив студент __Скиба О. М._____ _____
(прізвище, ім'я, по батькові студента) (*підпис*)

Керівник роботи ст. викладач каф. ІПЗ, Фаленкова М. В. _____
(*посада, прізвище, ім'я, по батькові)* *(підпис)*

«_29_» _____01_____ 2024 р.

АНОТАЦІЯ

кваліфікаційної роботи студента групи 402 ЧНУ ім. Петра Могили
Скиби Олега Миколайовича
Тема: «Вебзастосунок з продажу одягу»

Актуальність. У сучасному світі електронна комерція, зокрема продаж одягу через інтернет, стає все більш популярною та важливою сферою бізнесу. Швидкий доступ до інформації через вебсайти сприяє зручності та швидкості процесу покупки для споживачів. Вебсайти, що пропонують продукцію модного одягу, стають важливими інструментами як для покупців, так і для продавців.

Об'єктом роботи є процес розробки вебзастосунку для продажу в інтернеті.

Предметом роботи є технології та методи, які сприяють впровадженню та функціонуванню вебзастосунку з продажу одягу.

Мета роботи полягає у створенні вебзастосунку з продажу одягу, що відповідає потребам споживачів та сприяє розвитку електронної комерції в цій сфері

Робота складається з фахового розділу і спеціальної частини з охорони праці. Пояснювальна записка складається зі вступу, трьох розділів та висновків.

У першому розділі було проведено аналіз ринку електронної комерції в галузі продажу одягу, визначення актуальності теми, перегляд застосунків-аналогів та технологій для досягнення поставленої мети.

У другому розділі було виконано модулювання вебзастосунку, проаналізовано алгоритм роботи модулю та розглянуто роботу вебзастосунку на прикладі UML діаграми.

У третьому розділі було продемонстровано роботу над проектом, реалізація необхідних функцій та виправлення зустрічних помилок у ході розробки.

В результаті розроблено вебзастосунок мовами програмування Html, Scss, JavaScript, PHP з використанням бібліотек системи Npm та додатками Gulp, EmailJS та Open Server Panel.

Кваліфікаційна робота містить 66 сторінок, 65 рисунків, 3 таблиці, 25 використаних джерел та 0 додатків.

Ключові слова: Html, Scss, JavaScript, PHP, EmailJS, Npm, Gulp, MySQL, Visual Studio Code.

ABSTRACT

To the qualification work by student of group 402 of
Petro Mohyla Black Sea National University
Skyba Oleh
Theme: «Web application for selling clothes»

Relevance. In today's world, e-commerce, in particular the sale of clothing online, is becoming an increasingly popular and important area of business. Quick access to information through websites makes the buying process more convenient and fast for consumers. Websites offering fashion products are becoming important tools for both buyers and sellers.

The object of work is the process of developing a web application for online sales.

The subject of the study is technologies and methods that facilitate the implementation and functioning of a web application for selling clothes.

The purpose of the study is to create a web application for selling clothes that meets the needs of consumers and promotes the development of e-commerce in this area.

The work consists of a professional section and a special part on labor protection. The explanatory note consists of an introduction, three sections and conclusions.

The first section analyzes the e-commerce market in the field of clothing sales, determines the relevance of the topic, and reviews analogous applications and technologies to achieve the goal.

In the second section, the web application was modularized, the module's algorithm was analyzed, and the web application was examined using a UML diagram as an example.

The third section demonstrates the work on the project, the implementation of the necessary functions and the correction of errors encountered during development.

As a result, a web application was developed in the programming languages Html, Scss, JavaScript, PHP using Npm libraries and Gulp, EmailJS and Open Server Panel applications.

The bachelor's thesis contains 66 pages, 65 figures, 3 tables, 25 references and 0 appendices.

Keywords: Html, Scss, JavaScript, PHP, EmailJS, Npm, Gulp, MySQL, Visual Studio Code.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ

ПЗ	–	Програмне Забезпечення
HTML	–	Hypertext Markup Language
JS	–	JavaScript
NPM	–	Node Package Manager
PHP	–	Hypertext Preprocessor
SCSS	–	Syntactically Cascading Style Sheet

ВСТУП

Актуальність теми. У сучасному світі електронна комерція, зокрема продаж одягу через інтернет, стає все більш популярною та важливою сферою бізнесу. Швидкий доступ до інформації через вебсайти сприяє зручності та швидкості процесу покупки для споживачів. Вебсайти, що пропонують продукцію модного одягу, стають важливими інструментами як для покупців, так і для продавців.

Об’єкт роботи – процес розробки вебзастосунку з продажу в інтернеті.

Предмет роботи – технології та методи, які сприяють впровадженню та функціонуванню вебзастосунку з продажу одягу.

Мета кваліфікаційної роботи полягає у створенні ефективного вебзастосунку з продажу одягу, що відповідає потребам споживачів та сприяє розвитку електронної комерції в цій сфері.

Для досягнення поставленої **мети** передбачається виконання наступних завдань:

- аналіз ринку електронної комерції в галузі продажу одягу;
- обговорення деталей проекту;
- розробка та імплементація вебзастосунку, який відповідає сучасним вимогам та потребам споживачів;
- розробка backend-частини;
- тестування та вдосконалення функціоналу вебзастосунку для забезпечення його надійності та ефективності;
- запуск та підтримка вебзастосунку для забезпечення його стабільної роботи та задоволення потреб клієнтів.

1 ОГЛЯД ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ВЕБЗАСТОСУНКУ ПРОДАЖУ ОДЯГУ

1.1 Основні поняття та визначення

HTML – це мова розмітки тексту, яка використовується для створення веб-сторінок (рис. 1.1). Вона описує структуру та вміст вебсторінки, включаючи заголовки, абзаци, списки, таблиці, зображення, посилання та інші елементи. HTML є основою для всіх вебсторінок і використовується в поєднанні з іншими мовами, такими як CSS та JavaScript, для створення динамічних і інтерактивних веб-сайтів.

Рисунок 1.1 – Логотип HTML5

HTML забезпечує чітку структуру контенту веб-сторінки, що робить його легко читабельним як для людей, так і для машин. Це важливо для веб-застосунків продажу одягу, адже чітка структура контенту гарантує, що користувачі зможуть легко знаходити потрібні їм товари та інформацію. HTML - це порівняно проста мова розмітки, яку легко вивчити та використовувати. Це робить її доступною для розробників з різним рівнем досвіду, що полегшує процес розробки та обслуговування веб-застосунку. Також HTML є стандартизованою мовою, яка підтримується всіма

сучасними браузерами. Це гарантує, що веб-застосунок продажу одягу буде правильно відображатися та функціонувати на будь-якому пристрої з браузером [3].

SCSS (Sass CSS) – це потужний інструмент для розробки вебсайтів, що є розширенням мови CSS (рис. 1.2). Він пропонує зручні можливості, такі як використання змінних, вкладені стилі, міксини та інші функції, що полегшують написання та підтримку CSS-коду.

Рисунок 1.2 – Логотип SCSS

За допомогою SCSS можна створювати більш структурований та організований код, що сприяє підтримці та розширенню проектів. Крім того, SCSS дозволяє використовувати вбудовані функції для роботи з кольорами, шрифтами та іншими стилями. Використання SCSS спрощує процес розробки вебсайтів, забезпечуючи більшу гнучкість і ефективність у роботі зі стилями [3].

JavaScript – це мова програмування, яка використовується для додавання інтерактивності та динамічних можливостей до веб-сторінок (рис. 1.3). Вона дозволяє вебсторінкам реагувати на дії користувача, виконувати складні обчислення, візуалізувати дані та створювати вебзастосунки з багатим функціоналом. JavaScript є однією з найпопулярніших мов програмування для веброзробки і використовується практично на всіх вебсайтах. JavaScript дозволяє реалізувати різні інтерактивні функції, які покращують користувацький досвід веб-застосунку.

Рисунок 1.3 – Логотип JavaScript

Наприклад, фільтрація товарів за категорією, ціною, кольором та брендом; додавання товарів до корзини покупок; оновлення кількості товарів у кошику; відображення зображень товарів з різних кутів; анімації та ефекти для покращення візуального сприйняття. JavaScript дозволяє динамічно оновлювати контент веб-сторінок без необхідності повного перезавантаження сторінки. Це покращує швидкість роботи веб-застосунку та забезпечує більш плавний користувацький досвід. Наприклад, оновлення загальної вартості кошику покупок при додаванні або видаленні товарів. JavaScript дозволяє створити більш зручні та інтерактивні форми для введення даних користувачів, наприклад, форми для реєстрації, оформлення замовлення або відгуків [4].

PHP – це серверна мова програмування, яка використовується для створення динамічних вебсторінок та вебзастосунків (рис. 1.4). Вона дозволяє виконувати серверну логіку, обробляти форми, взаємодіяти з базами даних, генерувати HTML-код, керувати сесіями користувачів і багато іншого. PHP часто використовується разом з HTML, SCSS та JavaScript для створення повнофункціональних веб-сайтів і веб-застосунків. PHP є популярною мовою серед розробників завдяки своїй простоті у вивченні, гнучкості та широкому спектру можливостей.

Рисунок 1.4 – Логотип PHP

Вона має велику кількість готових бібліотек і фреймворків, які значно спрощують процес розробки. PHP також підтримується багатьма веб-серверами та операційними системами, що робить її універсальною та доступною. Для вебзастосунків продажу одягу PHP є ідеальним вибором, оскільки дозволяє створювати складні функціональні можливості, такі як управління каталогом товарів, кошиком покупок, обробка замовлень, автентифікація користувачів та багато іншого. Крім того, PHP може легко інтегруватися з різними базами даних, що забезпечує зберігання та обробку великого обсягу інформації про товари та клієнтів. PHP є мовою з відкритим кодом, що означає, що вона безкоштовна для використання і має активну спільноту розробників, які постійно вдосконалюють її та створюють нові інструменти і ресурси. Це робить PHP доступною для розробників з різним рівнем досвіду і бюджетів, що сприяє швидкій та ефективній розробці веб-застосунків [5].

EmailJS – це сервіс, який дозволяє відправляти електронні листи безпосередньо з JavaScript-коду, без необхідності використання серверної логіки (рис. 1.5). Він спрощує процес інтеграції функціоналу надсилання електронної пошти в вебзастосунки, дозволяючи розробникам фокусуватися на клієнтській частині додатка. EmailJS працює шляхом використання спеціальних шаблонів електронних листів і підтримує інтеграцію з популярними поштовими сервісами, такими як Gmail, Outlook, Yahoo та іншими. Це забезпечує високу надійність і швидкість доставки повідомлень.

Рисунок 1.5 – Логотип EmailJS

Для використання EmailJS достатньо створити обліковий запис, налаштувати шаблони листів та підключити сервіс до вашого вебзастосунку за допомогою JavaScript SDK. Цей сервіс ідеально підходить для різноманітних задач, таких як відправка форм зворотного зв'язку, підтвердження реєстрації, сповіщення про замовлення та інші автоматизовані повідомлення. EmailJS дозволяє налаштовувати динамічні шаблони листів, що робить можливим персоналізацію повідомлень для кожного користувача. Для вебзастосунків продажу одягу EmailJS може бути надзвичайно корисним. Він забезпечує швидку і просту інтеграцію функціоналу відправки повідомлень, таких як підтвердження замовлення, оновлення статусу доставки та інші важливі сповіщення. Це покращує взаємодію з користувачами та підвищує рівень обслуговування клієнтів. EmailJS є хмарним сервісом, що означає, що не потрібно турбуватися про налаштування і обслуговування серверної інфраструктури для надсилання електронних листів. Він також забезпечує високу безпеку даних і підтримує шифрування повідомлень, що є важливим для захисту конфіденційної інформації користувачів [6].

NPM (Node Package Manager) – це менеджер пакетів для JavaScript, який використовується для встановлення, оновлення та видалення бібліотек JavaScript (рис.

1.6). NPM має величезний репозиторій пакетів, які надають широкий спектр функцій для веброзробки, таких як фреймворки, компоненти, інструменти та бібліотеки. NPM полегшує процес розробки вебсайтів, дозволяючи використовувати готові бібліотеки та компоненти замість написання коду з нуля.

Рисунок 1.6 – Логотип NPM

NPM дозволяє використовувати готові бібліотеки JavaScript, які надають різні функції, необхідні для вебзастосунку продажу одягу. Це заощаджує розробникам час та зусилля, адже їм не потрібно писати весь код з нуля. Завдяки великій кількості готових бібліотек в NPM, розробники можуть зосередитися на власній логіці веб-застосунку, а не на написанні базових функцій. NPM дозволяє легко оновлювати бібліотеки до останніх версій, що гарантує використання найновіших функцій та виправлення помилок [7].

Gulp – це інструмент автоматизації завдань для веброзробки, який використовується для автоматизації повторюваних завдань, таких як компіляція SCSS, мінімізація JavaScript, об'єднання файлів та розгортання вебсайтів (рис. 1.7).

Рисунок 1.7 – Логотип Gulp

Gulp дозволяє економити час та підвищувати продуктивність розробників, автоматизуючи рутинні завдання та оптимізуючи процес розробки вебсайтів. Gulp автоматизує багато рутинних завдань, таких як компіляція, мінімізація та об'єднання файлів, що значно економить час розробників. Завдяки автоматизації завдань, Gulp дозволяє розробникам зосередитися на більш важливих аспектах вебзастосунку, таких як написання коду, тестування та дизайн. Gulp може використовуватися для автоматизації таких завдань, як перевірка коду, що може допомогти у виявленні та виправленні помилок на ранніх стадіях розробки. Gulp гарантує, що всі завдання виконуються послідовно та однаково, що важливо для забезпечення стабільності та надійності веб-застосунку.[8]

Visual Studio Code (VS Code) - це потужний та легкий текстовий редактор, який широко використовується для розробки веб-застосунків, включаючи ті, що працюють з одягом (рис. 1.8). VS Code пропонує широкий спектр функцій, включаючи автодоповнення коду, вбудовану підтримку Git, інтеграцію з розширеннями та плагінами, а також можливості налаштування зовнішнього вигляду та розширення функціональності.

Рисунок 1.8 – Логотип VS Code

Завдяки своїм зручним інструментам для розробки, VS Code дозволяє розробникам створювати та підтримувати веб-застосунки для продажу одягу з високою ефективністю та продуктивністю. Його швидкість роботи та надійність роблять його вибором багатьох професіоналів у галузі веброзробки.

Open Server Panel – це програмне забезпечення, яке використовується для створення локального серверного середовища на Windows (рис. 1.9). Воно дозволяє розробникам легко налаштовувати та керувати локальними вебсерверами, що спрощує процес розробки, тестування та налагодження вебзастосунків. Open Server Panel включає в себе популярні серверні програми, такі як Apache, Nginx, MySQL, PostgreSQL, PHP, та інші, забезпечуючи повнофункціональне серверне середовище.

Рисунок 1.9 – Логотип Open Server Panel

Open Server Panel забезпечує зручний інтерфейс для керування серверами та базами даних. Розробники можуть легко запускати та зупиняти сервери, змінювати конфігурації, створювати та керувати базами даних без необхідності вручну редагувати конфігураційні файли. Це значно спрощує процес розробки та дозволяє зосередитися на написанні коду. Для веб-застосунків продажу одягу Open Server Panel є незамінним інструментом. Він дозволяє створити локальне середовище, яке точно відтворює умови реального сервера, що забезпечує більш точне тестування та налагодження. Це особливо важливо для перевірки функціональності, продуктивності та безпеки веб-застосунку перед його запуском у продакшн. Open Server Panel підтримує різні версії PHP та інших серверних програм, що дозволяє розробникам тестувати свої веб-застосунки в різних умовах і забезпечувати їх сумісність з різними конфігураціями. Крім того, він має вбудовані інструменти для моніторингу продуктивності та налагодження, що робить його потужним інструментом для веб-розробки.[9]

1.2 Огляд застосунків-аналогів

Під час аналізу вимог до програмного забезпечення для веб-застосунку продажу одягу, важливим етапом є порівняння з аналогами. Це допомагає визначити, чи існують на ринку програмні засоби, які можуть задовольнити потреби користувачів, або які можуть бути використані в якості основи для створення нового програмного забезпечення. Такий аналіз допомагає визначити критерії функціональності, інтерфейсу користувача, швидкодії, масштабованості та інших параметрів [1].

У контексті веб-застосунку для продажу одягу було проведено порівняльний аналіз аналогів, щоб виявити основні функції та потенційні недоліки. Це дозволить зрозуміти, які аспекти потребують уваги для подальшого вдосконалення розроблюваного застосунку. Було розглянуто наступні застосунки-аналоги.

Asos – це популярний онлайн-магазин одягу, який пропонує широкий вибір одягу, взуття та аксесуарів для чоловіків, жінок та дітей. На сайті Asos можна знайти

як одяг відомих брендів, так і одяг власних брендів Asos. Asos пропонує безкоштовну доставку та повернення товару, а також різні способи оплати. Сайт доступний англійською мовою, а також мовами інших країн, що робить його доступним для широкої аудиторії [10].

Таблиця 1.1 – Опис веб-застосунку Asos

Назва	**Asos**
Функції	- додавання товарів до кошика; - оформлення замовлення; - доставка; - широкий вибір фільтрів для пошуку товарів; - персональні рекомендації; - соціальні мережі.
Недоліки	- складна навігація; - перевантаження інформації; - повільна продуктивність сайту; - відсутність локалізації для всіх країн.
Переваги	- широкий вибір товарів; - конкурентні ціни; - зручний інтерфейс користувача; - можливість оплати різними способами; - доступність на різних мовах.
Посилання	https://www.asos.com/

Zalando – це популярний онлайн-магазин одягу, взуття та аксесуарів, який працює в 17 країнах Європи. На сайті Zalando можна знайти широкий вибір товарів від понад 2 500 брендів, включаючи як відомі бренди, так і молоді дизайнерські бренди. Zalando пропонує безкоштовну доставку та повернення товару, а також різні способи оплати. Сайт доступний мовами країн, де він працює, що робить його доступним для широкої аудиторії [11].

Таблиця 1.2 – Опис веб-застосунку Zalando

Назва	**Zalando**
Функції	- додавання товарів до кошика; - оформлення замовлення; - доставка; - повернення товару; - широкий вибір фільтрів для пошуку товарів; - персональні рекомендації; - соціальні мережі.
Переваги	- широкий вибір товарів; - зручний інтерфейс користувача; - можливість оплати різними способами; - доступність на різних мовах; - активний контент у блозі та соціальних мережах.
Недоліки	- складна навігація; - погана оптимізація продуктивності сайту; - проблеми з адаптивом; - надмірна складність деяких функцій.
Посилання	https://www.zalando.com/

Farfetch - це онлайн-платформа, що діє як глобальний маркетплейс для люксового одягу. Вона відрізняється від звичайних інтернет-магазинів тим, що не володіє власним запасом товарів, а співпрацює з бутиками та брендами по всьому світу. Це дозволяє Farfetch пропонувати широкий асортимент продукції від понад 3500 брендів, включаючи всесвітньові люксові марки, незалежні дизайнерські будинки та популярні streetwear бренди [12].

Таблиця 1.3 – Опис веб-застосунку Farfetch

Назва	**Farfetch**
Функції	- додавання товарів до кошика; - оформлення замовлення; - доставка; - багатомовний інтерфейс; - персональні рекомендації.

Кінець таблиці 1.3

Переваги	- широкий вибір люксових брендів та дизайнерів; - унікальні та ексклюзивні товари; - зручний інтерфейс користувача; - багатомовна підтримка.
Недоліки	- складність навігації та інтерфейсу; - проблеми з оптимізацією; - проблеми з поверненням коштів; - недостатня адаптивність.
Посилання	https://www.farfetch.com/

На рисунку 1.10 було розглянуто головну сторінку вебзастосунку Asos.

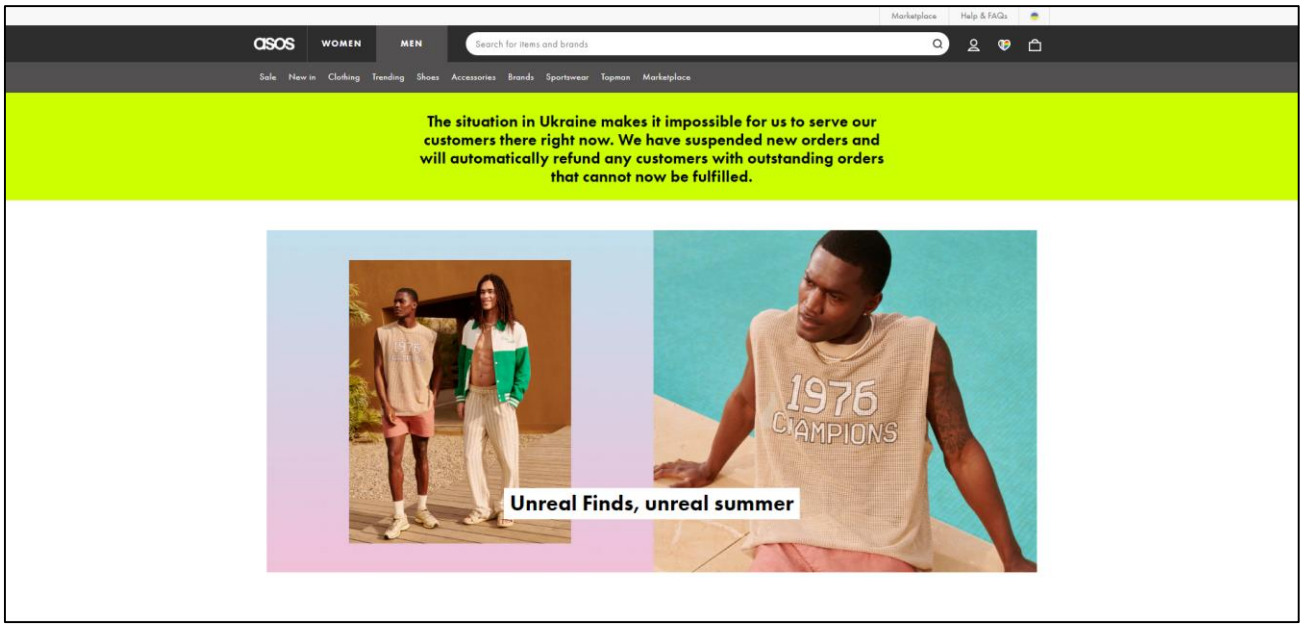

Рисунок 1.10 – Зовнішній вигляд головної сторінки Asos

На рисунку 1.11 було розглянуто сторінку каталогу вебзастосунку Asos.

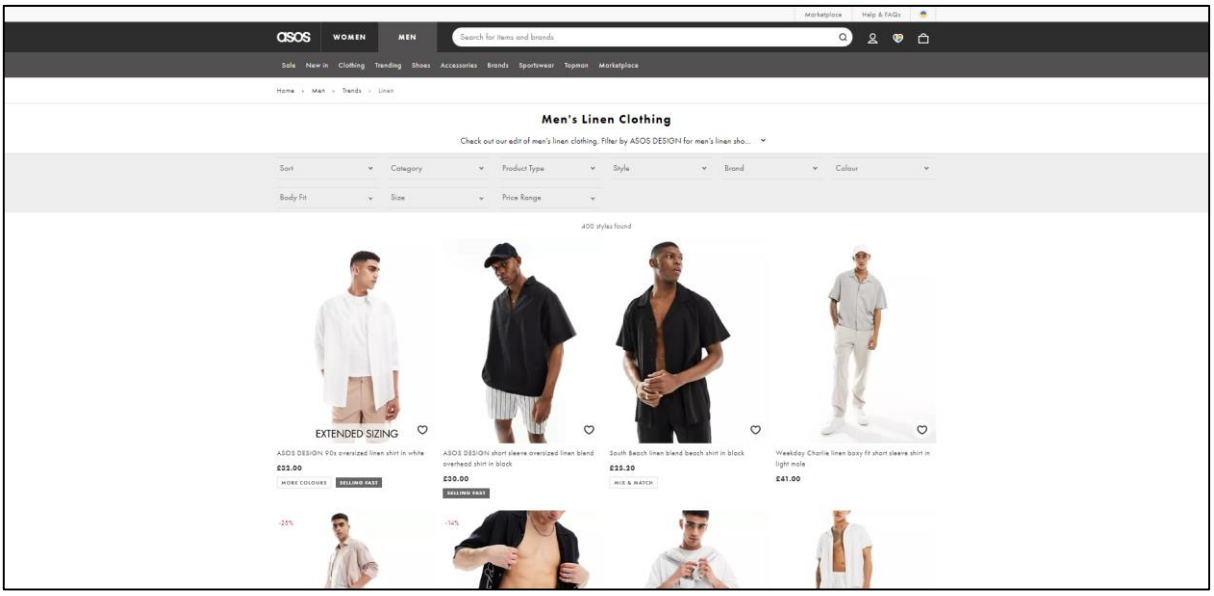

Рисунок 1.11 – Зовнішній вигляд сторінки каталогу сайту Asos

На рисунку 1.12 було розглянуто сторінку товару вебзастосунку Asos.

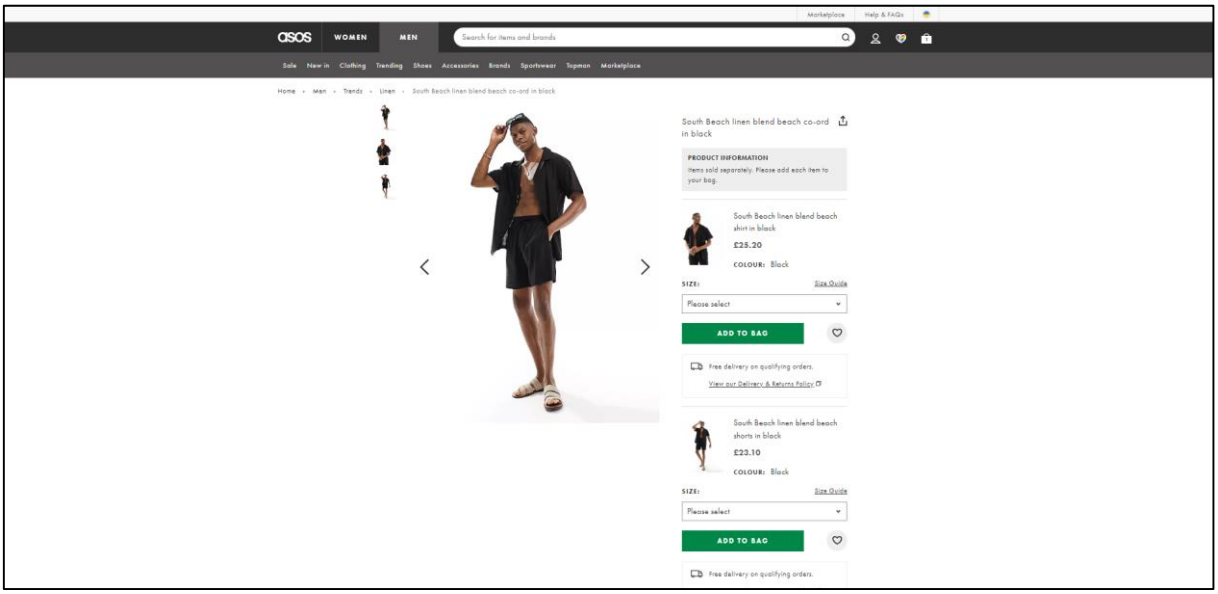

Рисунок 1.12 – Зовнішній вигляд сторінки товару сайту Asos

На рисунку 1.13 було розглянуто сторінку кошика вебзастосунку Asos.

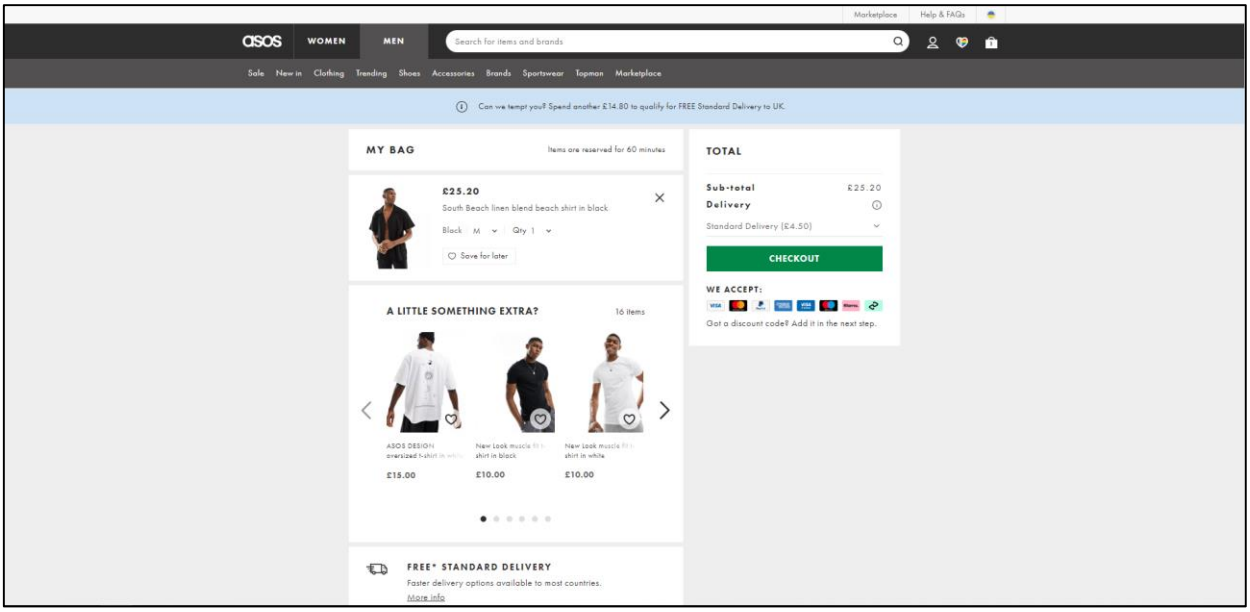

Рисунок 1.13 – Зовнішній вигляд сторінки кошика сайту Asos

На рисунку 1.14 було розглянуто сторінку каталогу вебзастосунку Zalando.

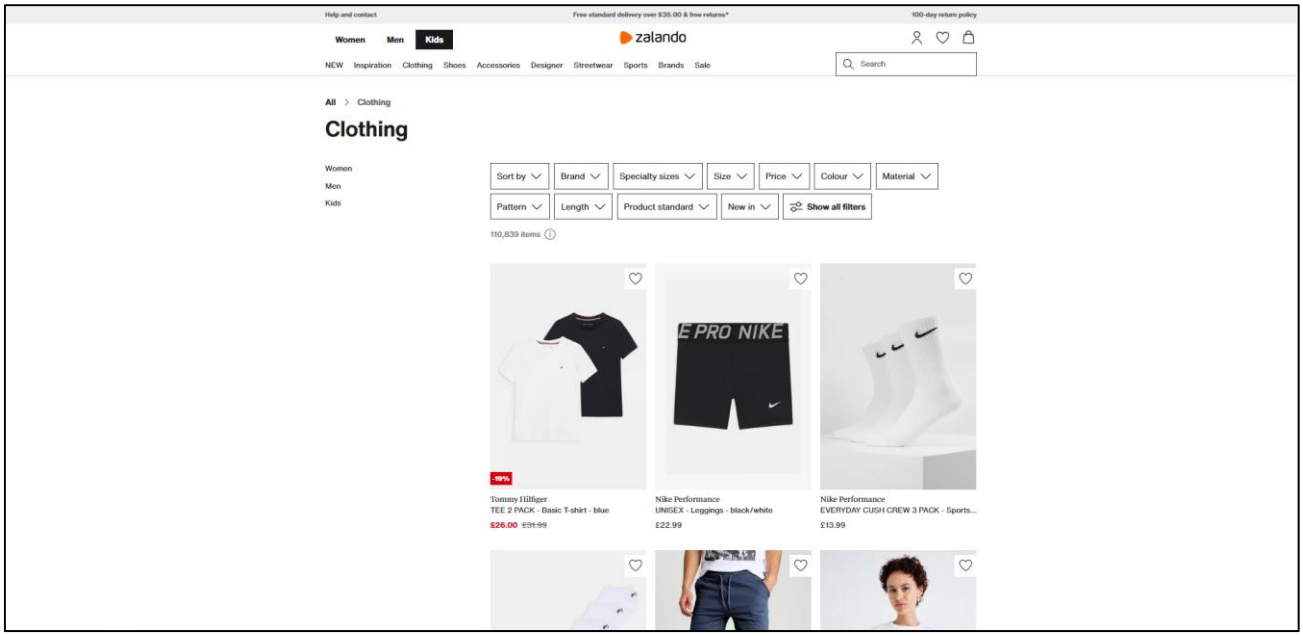

Рисунок 1.14 – Зовнішній вигляд сторінки каталогу сайту Zalando

На рисунку 1.15 було розглянуто сторінку товару вебзастосунку Zalando.

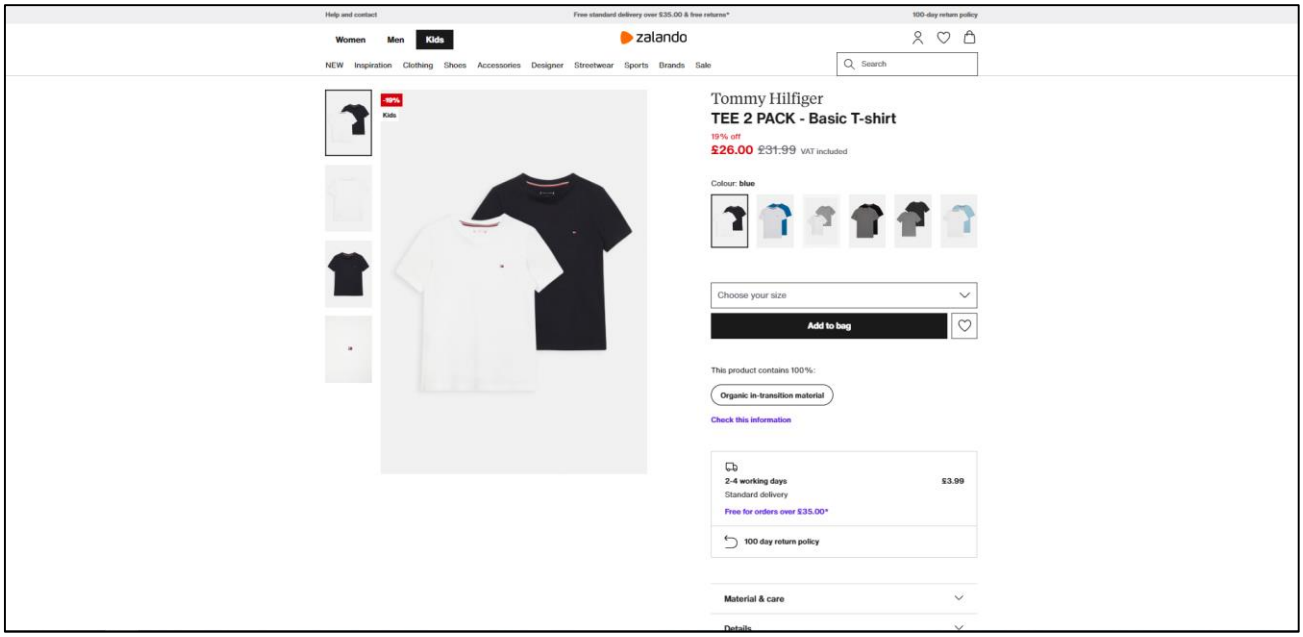

Рисунок 1.15 – Зовнішній вигляд сторінки товару сайту Zalando

На рисунку 1.16 було розглянуто секцію інформації вебзастосунку Farfetch.

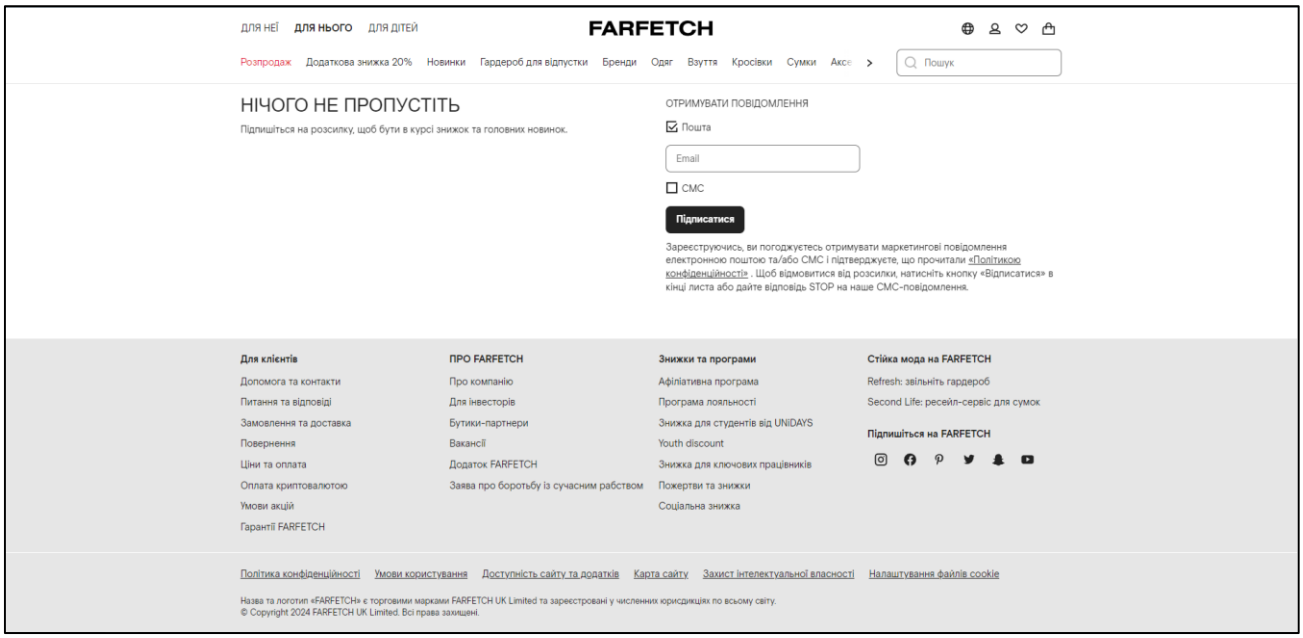

Рисунок 1.16 – Зовнішній вигляд секції інформації сайту Farfetch

На рисунку 1.17 було розглянуто секцію інформації про товар вебзастосунку Farfetch.

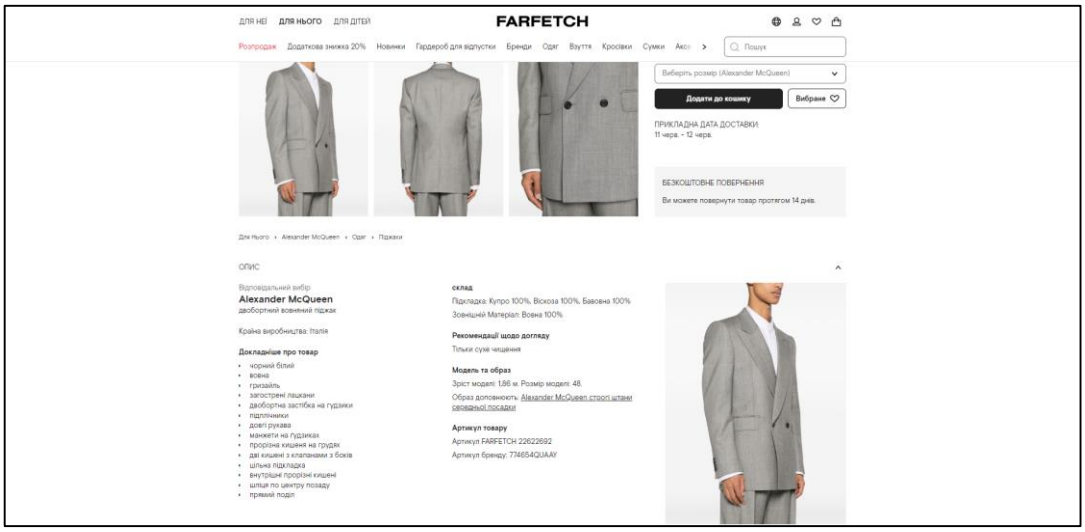

Рисунок 1.17 – Зовнішній вигляд секції інформації про товар сайту Farfetch

Представлені вище інтернет-магазини більш-менш схожі між собою за функціоналом та наявності деяких сторінок. Існує ще багато інтернет-магазинів одягу які будуть схожі з попередніми, але наведених трьох сайтів-прикладів буде достатньо для загального представлення функціоналу схожих сайтів.

Фактори на які було звернуто особливу увагу:

– доступність для широкої аудиторії;
– актуальність інформації;
– простота використання інтерфейса;
– доступність на будь-якій платформі.

1.3 Огляд технологій, методів для розробки вебзастонку продажу одягу

Розробка веб-застосунків для продажу одягу вимагає використання сучасних технологій та ефективних підходів для створення зручного та привабливого для користувачів середовища. Ось розширений огляд деяких існуючих технологій,

методів та підходів у цій сфері.

HTML, SCSS, JavaScript, PHP: чотири основних технології, які часто використовуються в розробці веб-застосунків. HTML (HyperText Markup Language) використовується для створення структури сторінок, SCSS (Sassy CSS) - для стилізації та оформлення, JavaScript – для надання інтерактивності та функціональності, таких як динамічне оновлення вмісту та взаємодія з користувачем, PHP – для виконання серверної логіки, обробляти форми, взаємодіяти з базами даних, генерувати HTML-код, керувати сесіями користувачів і багато іншого.

Npm та Gulp: ці інструменти допомагають в автоматизації процесів розробки. Npm (Node Package Manager) дозволяє керувати залежностями та встановлювати необхідні пакети, а Gulp – автоматизує завдання, такі як компіляція SCSS в CSS, злиття та мінімізація JavaScript файлів, що прискорює процес розробки.

EmailJS: сервіс, який дозволяє відправляти електронні листи безпосередньо з JavaScript-коду, без необхідності використання серверної логіки. Він спростить процес інтеграції функціоналу надсилання електронної пошти в вебзастосунки, дозволяючи розробникам фокусуватися на клієнтській частині додатка.

Респонсивний дизайн: у зв'язку з різноманітністю пристроїв, на яких може переглядатися вебзастосунок, важливо забезпечити його респонсивність. Це означає, що дизайн має пристосовуватися до розміру та орієнтації екрану користувача, забезпечуючи зручне використання незалежно від пристрою. Для досягнення цього можна використовувати CSS-фреймворки, такі як Bootstrap або Foundation, які надають готові компоненти та стилі для розробки респонсивних інтерфейсів.

SEO-оптимізація: щоб забезпечити високий рівень трафіку та видимості в пошукових системах, важливо оптимізувати вебзастосунок для пошукових запитів. Це включає в себе використання відповідних метатегів, оптимізацію швидкості завантаження сторінок, використання правильних ключових слів у контенті та структуру URL-адрес для поліпшення індексації веб-сторінок пошуковими системами.

Аналітика та відстеження даних: для ефективного управління та вдосконалення веб-застосунку важливо використовувати аналітичні інструменти та системи відстеження даних. Вони допомагають збирати і аналізувати інформацію про відвідувачів, їх поведінку та уподобання, що дозволяє приймати обгрунтовані рішення для покращення досвіду користувача та ефективності продажів. Популярні інструменти аналітики включають Google Analytics, який надає різноманітні звіти та аналітику щодо відвідувачів та їх дій на сайті.

Інтеграція з платіжними системами: для забезпечення зручних та безпечних платежів важливо інтегрувати веб-застосунок з популярними платіжними системами, такими як PayPal, Stripe, а також забезпечити безпеку та конфіденційність особистих даних користувачів під час оплати. Це може бути досягнуто за допомогою API цих систем, що дозволяють безпечно обмінюватися даними між веб-застосунком та платіжною системою. Також важливо враховувати вимоги щодо дотримання стандартів безпеки, таких як PCI DSS, для захисту конфіденційності платіжних даних користувачів.

Висновки до розділу 1

У першому розділі було розглянуто широкий спектр технологій та методів для розробки та підтримки вебзастосунків продажу одягу. Використання технологій HTML, SCSS, JavaScript, PHP, EmailJS, Npm та Gulp дозволяє ефективно створювати та підтримувати вебзастосунки, забезпечуючи їхню функціональність та зручність для користувачів. Також було розглянуті інтернет-магазини аналоги, визначені їхні основні переваги та недоліки. Визначено, що більшість являються комерційними розробками і недоступні ні у вигляді програмного коду, ні у вигляді формального опису алгоритмів. Сформульовані задачі досліджень дипломної роботи та основні вимоги щодо створення вебзастосунку.

2 МОДЕЛЮВАННЯ ВЕБЗАСТОСУНКУ З ПРОДАЖУ ОДЯГУ

2.1 Алгоритм роботи сценаріїв

Веб-застосунок з продажу одягу є простим та інтуїтивно зрозумілим для будь-якого користувача. Цей веб-застосунок не матиме обмежень і буде доступний кожному. Клієнти зможуть користуватися ним на телефоні, ноутбуці або персональному комп'ютері.

Основна мета веб-застосунку – підвищення продажів товарів в Інтернеті. Основні функціональні можливості модуля включають:

- пошук інформації про товари;
- перегляд інформації про товари;
- додавання товарів у кошик;
- здійснення замовлення;
- рекомендації щодо підбору аксесуарів до товару;
- авторизація;
- реєстрація.

Блок-схема є найпоширенішим типом графічних моделей, що описують процеси або алгоритми, де різні кроки зображуються блоками різної форми, об'єднаними між собою лініями, які вказують напрям виконання. Наприклад, прямокутник відображає певну дію, ромб – умову або формулу, паралелограм – введення або виведення даних тощо. Блок-схема допоможе схематично представити процес роботи модуля. Варто зазначити, що веб-застосунок є безкоштовним, тому кожен охочий зможе здійснювати певні дії та отримувати бажаний результат.

Алгоритм додавання товару до кошика та оформлення замовлення (рис. 2.1).

1) переходимо на сторінку бажаного товару;
2) обираємо необхідні опції кольору, розміру та кількість замовлення товару;

3) натискаємо кнопку «Add to Cart» що дозволить додати товар до кошика;

4) переходимо до кошика. За бажанням можна залишити примітки до замовлення, після чого натискаємо кнопку «Checkout»;

5) після переходу до «Checkout» необхідно ввести контактні данні та інформацію про доставку. На цьому етапі якщо всі поля заповнені правильно і інформація пройшла валідацію, користувач буде перенаправлений на етап оплати. В іншому випадку він отримає повідомлення про поля, які потрібно заповнити повторно.

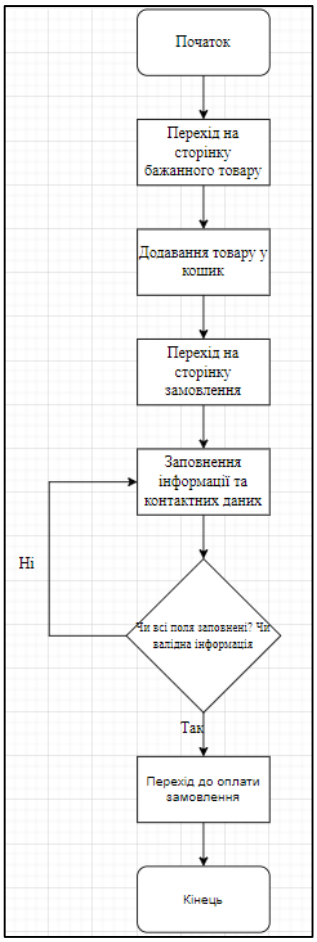

Рисунок 2.1 – Блок-схема алгоритму додавання товару до кошика та оформлення замовлення

Алгоритм створення облікового запису (рис. 2.2).

1) переходимо на сторінку реєстрації;
2) заповнюємо наведені поля даними (ім’я, прізвище, e-mail, пароль);
3) натискаємо кнопку «Create». Якщо всі поля було заповнено вірно, на пошту користувача, яка була вказана, находить лист з активацією акаунту. Якщо хоча б одне х полів не заповнено чи заповнено некоректно, або такий акаунт вже існує за вказаною поштою, буде виведено відповідну помилку.

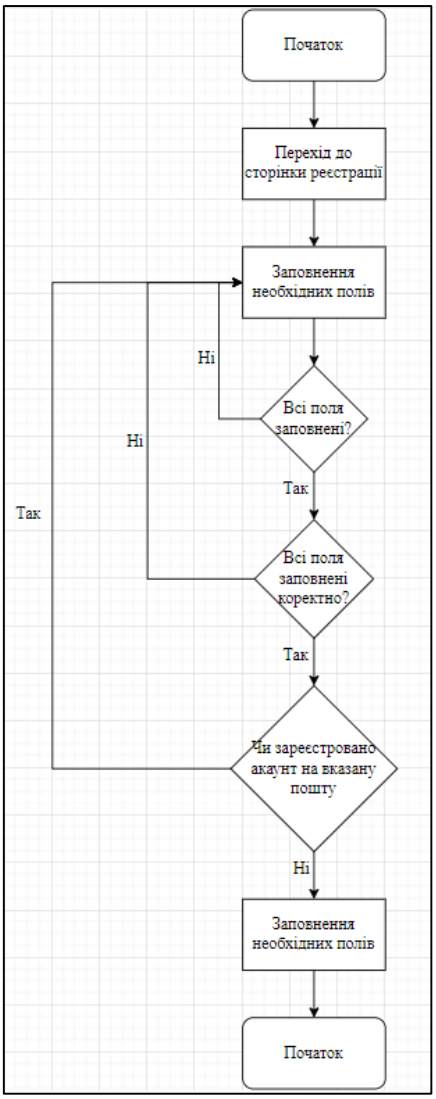

Рисунок 2.2 – Блок-схема алгоритму створення облікового запису

Алгоритм відновлення паролю акаунту (рис. 2.3).

1) переходимо на сторінку відновлення паролю;
2) вказуємо електронну адресу, до якої прив’язаний акаунт. Якщо поле заповнено некоректно або зазначеної адреси немає в базі зареєстрованих користувачів, виведеться повідомлення про помилку. Якщо введена електрона адреса є в базі зареєстрованих користувачів, на неї буде надіслано лист із посиланням для відновлення паролю;
3) переходимо за посиланням у листі;
4) вказуємо новий пароль та підтверджуємо його у відповідному полі;
5) натискаємо кнопку «Змінити пароль». Якщо паролі в обох полях співпадають, пароль для даного акаунту буде змінено. Якщо паролі не співпадають, з’явиться повідомлення про помилку.

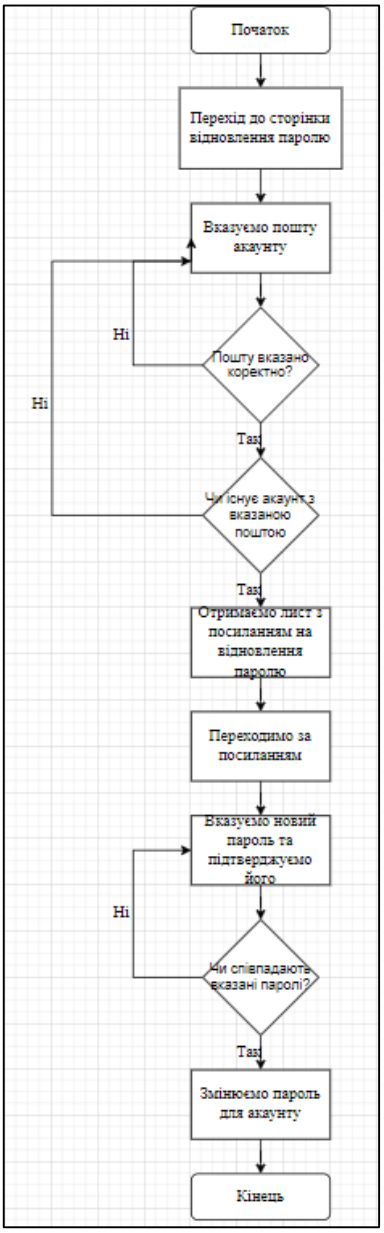

Рисунок 2.3 – Блок-схема алгоритму відновлення паролю для облікового запису

Алгоритм авторизації до акаунту (рис. 2.4).

1) переходимо на сторінку авторизації;

2) вказуємо пошту та пароль. Якщо всі поля заповнені, заповнені коректно, вказана пошта є у базі зареєстрованих користувачів та пароль до цього акаунту вказано вірно, тоді користувача буде авторизовано у систему. Інакше буде виведено відповідне повідомлення про помилку.

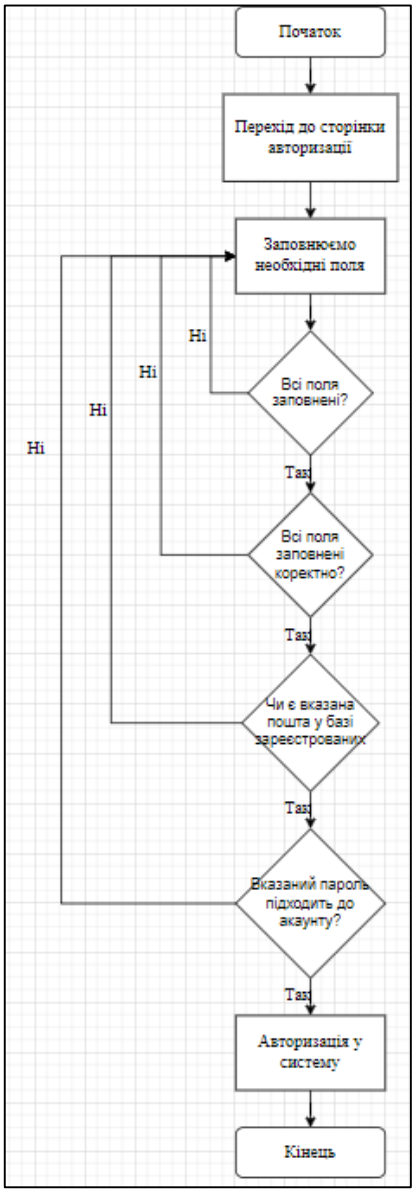

Рисунок 2.4 – Блок-схема алгоритму авторизації

Алгоритм пошуку товарів (рис. 2.5).

1) натискаємо на значок пошуку;

2) вводимо у поле пошуку відповідний запит. Якщо є товари, що відповідають запиту, перші чотири з них будуть відображатись одразу, ще до натискання кнопки пошуку;

3) натискаємо на кнопку пошуку. Після цього будуть відображені всі товари, що відповідають запиту. Якщо жодного товару не знайдено, з’явиться повідомлення про відсутність відповідних товарів.

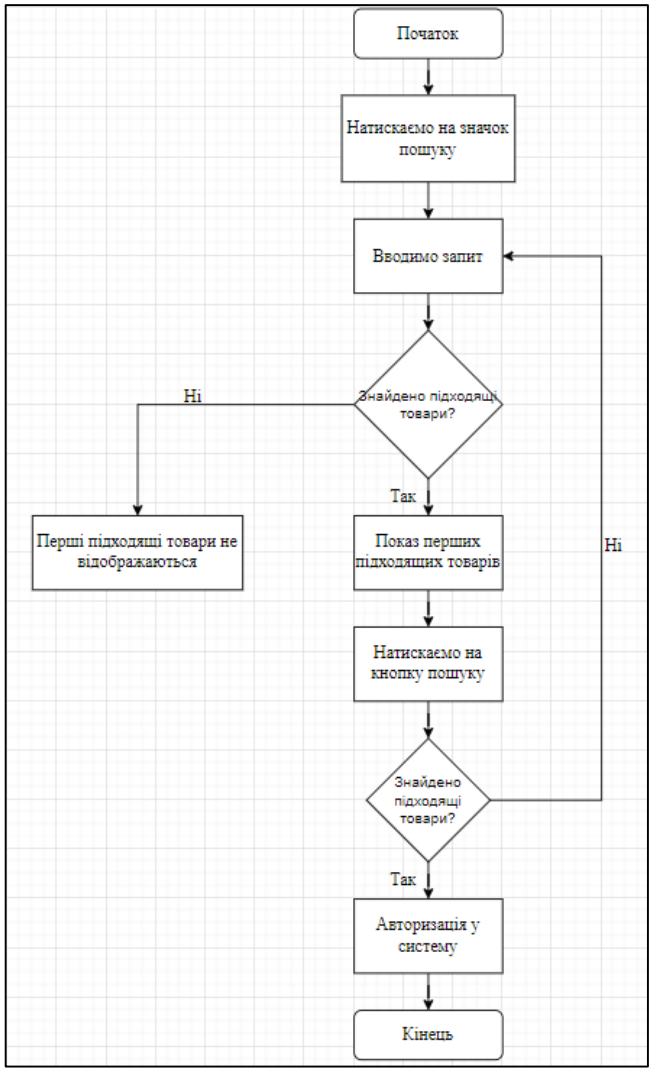

Рисунок 2.5 – Блок-схема алгоритму пошуку товарів

Розробка блок-схеми здійснюється на етапі документування або розробки програмного забезпечення. Це робиться для уникнення значних ризиків, пов'язаних з основними помилками під час проектування та розробки програмного забезпечення. Іноді створення блок-схем є необхідним, оскільки існують візуальні мови програмування. Крім того, блок-схеми використовуються для верифікації алгоритмів.

2.2 Робота веб-застосунку на прикладі UML діаграми

Unified Modeling Language (UML) – це уніфікована графічна мова для об’єктного моделювання у сфері розробки програмного забезпечення. UML має широкий спектр застосувань: системне проектування, відображення організаційних структур та моделювання бізнес-процесів.

UML має власний синтаксис і правила оформлення моделей. Використовуючи графічну нотацію UML, можна детально візуалізувати систему, сприяти її покращенню та уточненню під час розробки, а також об’єднати всі компоненти в єдину структуру.

Відносини в UML описують, як дві або більше сутностей можуть взаємодіяти під час виконання системи. Вони використовуються для представлення зв'язків між структурними, групуючими або поведінковими елементами. Основні типи відносин включають:

– асоціації – набір посилань, що пов’язують елементи моделі UML, визначаючи кількість об’єктів, які беруть участь у відносинах;

– залежності – показують, як два або більше об’єктів залежать один від одного. Зміни в одному об’єкті можуть впливати на інші;

– узагальнення – використовується для подання спадкування;

– реалізація – зустрічається у випадку інтерфейсів, коли одна сутність визначає відповідальність, яку інша сутність реалізує.

UML містить 12 типів діаграм [13]:

– 5 типів представляють поведінкові аспекти системи;

– 4 типи діаграм представляють статичну структуру додатку;

– 3 типи представляють фізичні аспекти функціонування системи (діаграми реалізації).

Деякі з видів діаграм специфічні для певних систем і додатків. Найбільш поширені з них:

– діаграми об’єктів – демонструє частини об’єктів системи та зв’язки між ними;

– діаграми класів – демонструє логічну модель системи;

– діаграми співробітництва – демонструє взаємозв’язки об’єктів;

– діаграма активностей – демонструє бізнес-процеси об’єкта, показуючи послідовність, розгалуження та синхронізацію процесів;

– діаграма станів – демонструє стани об’єктів системи;

– діаграма послідовності – демонструє послідовність передачі повідомлень між об’єктами, виділяючи послідовність прийому та передачі повідомлень;

– діаграма розгортання – демонструє декомпозицію системи на різні типи пристроїв;

– діаграма прецедентів – демонструє функціональне призначення системи, відображає об’єкти та завдання, які вони виконують.

Діаграма прецедентів включає:

– актор – сутність, що виконує певну роль у системі, може бути зовнішньою системою, людиною або організацією. Зображується як скелет;

– система – визначає сферу застосування, зображується у вигляді прямокутника. Корисний для візуалізації великих систем;

– випадок використання – функція або дія всередині системи, зображується як овал із назвою функції;

– пакет – використовується для угруповання випадків використання, корисний у складних діаграмах.

Нижче представлена розроблена UML діаграма прецедентів, де показана взаємодія веб-застосунку з усіма частинами модуля (рис. 2.6).

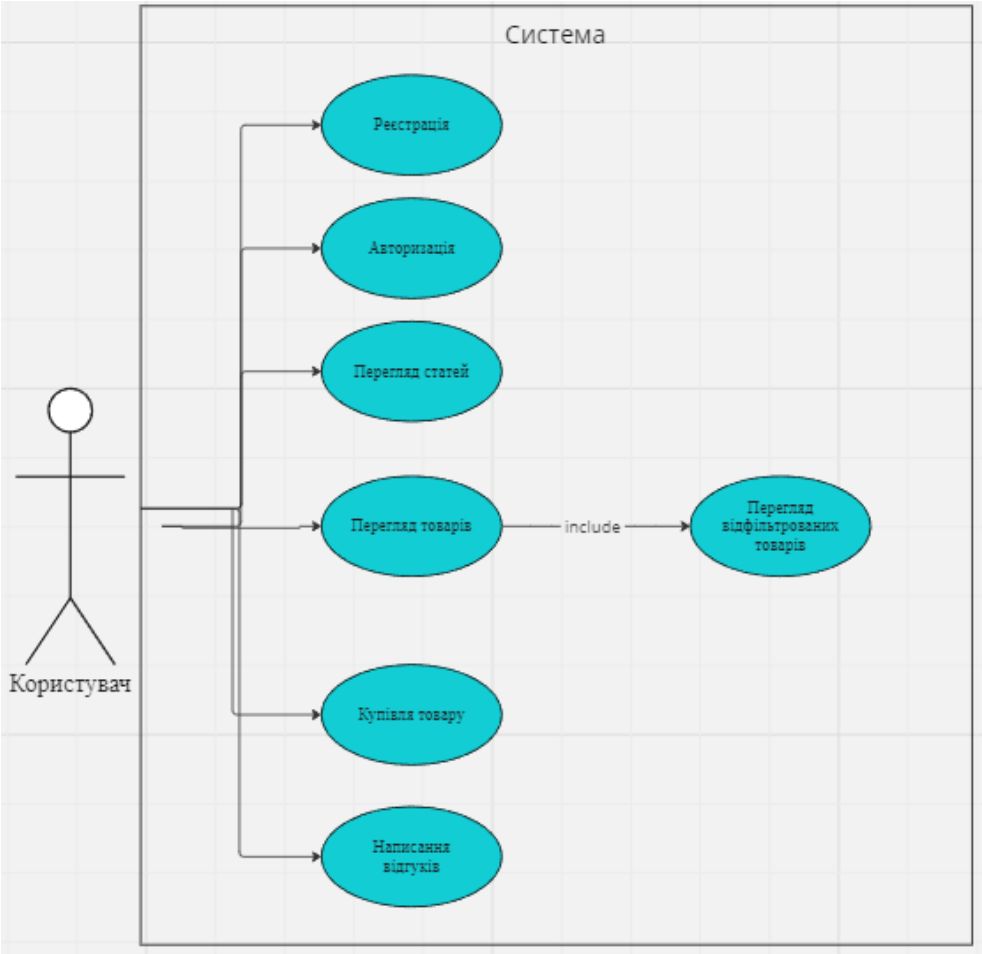

Рисунок 2.6 – UML діаграма прецендентів роботи модулю

На наступній UML діаграмі розгортання представлено архітектуру сервісів системи електронної комерції Shopify. Існують окремі служби для різних модулів електронної комерції, які взаємодіють за допомогою REST API. Вебзастосунок використовує службу шлюзу для доступу до інших служб. Мобільний додаток підключений до модуля шлюзу мобільного API, який забезпечує зв'язок з іншими сервісами (рис. 2.7).

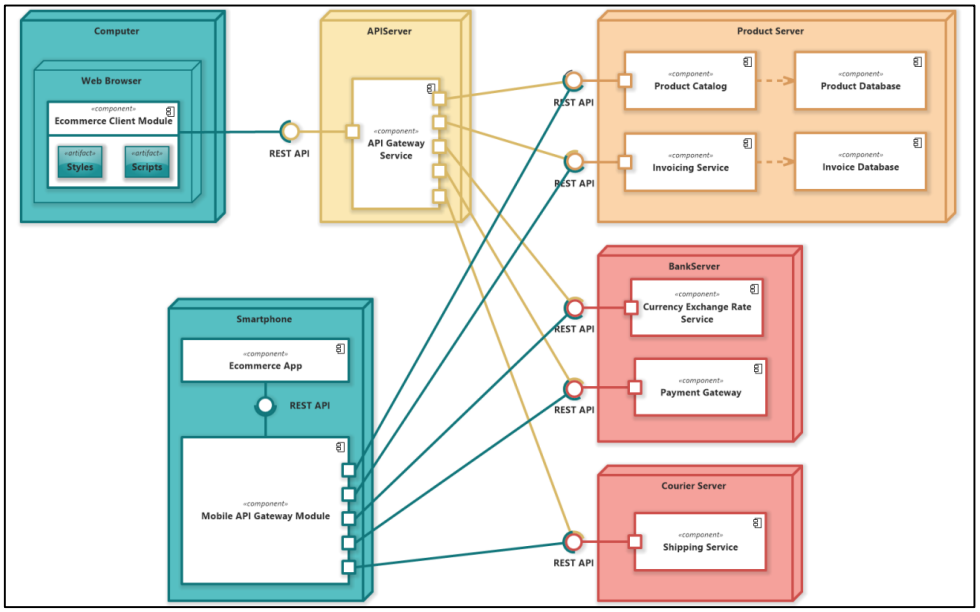

Рисунок 2.7 – UML діаграма розгортання

Платформа Shopify має певні обмеження на створення варіантів продукту – до 100 варіантів на продукт. Оскільки продукти веб-застосунку можуть мати кілька опцій, можливі ситуації, коли продукт матиме більше 100 варіантів. Тому було вирішено додавати варіанти кольору до магазину як окремі товари з іншим набором опцій. На сторінці продукту передбачена можливість перемикання не лише між опціями самого продукту, але й між варіантами кольору, розміру та кількістю (рис. 2.8).

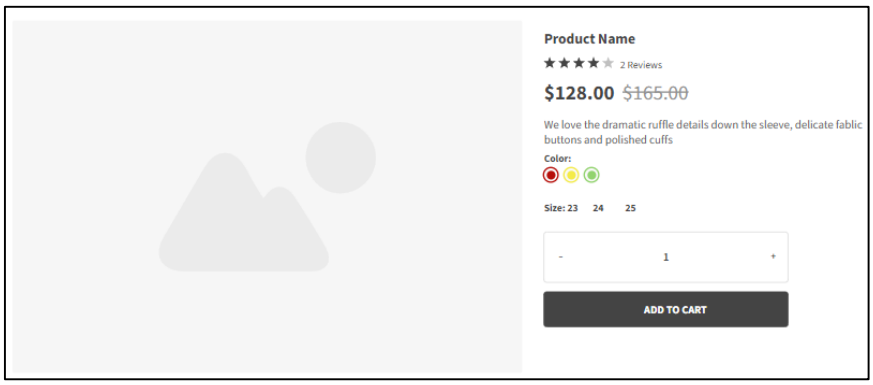

Рисунок 2.8 – Mockup форми на сторінці продукту

Висновки до розділу 2

Отже, у другому розділі визначено основні функціональні можливості модуля та описано алгоритми роботи для наступних сценаріїв: додавання товару до кошика з оформленням замовлення, створення облікового запису, процес авторизації, відновлення паролю до акаунту та пошуку товарів. Для більш точної та зрозумілої роботи системи було створено UML діаграми прецедентів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБЗАСТОСУНКУ ТА ПРОВЕДЕННЯ ТЕСТУВАННЯ

3.1 Верстка основних сторінок

На початку розробки сайту було зроблено верстку сторінок, що будуть відображатися для відвідувачів сайту, а саме:

- головна сторінка;
- каталог товарів;
- сторінка товару;
- кошик;
- сторінка реєстрації/авторизації особистого кабінету.

Зазвичай на головній сторінці розміщують важливу та цікаву інформацію, щоб залучити покупця продовжити переходити на інші сторінки. На головній сторінці можна розміщувати будь-які види секцій та у бажаному порядку. Для неї було обрано наступні секції:

- hero (рис. 3.1) – на ній було розміщено вітальний банер головної сторінки сайту;
- banners (рис. 3.2) – на ній було розміщено рекомендації про товари з різних колекцій та на яких діє знижка;
- main-products (рис. 3.3 – 3.4) – на ній було розміщено основні товари сайту, з можливістю обрати запропоновані категорії, та розгорнути більший список за допомогою кнопки “Load more”;
- shop-banners (рис. 3.5) – на ній було розміщено банери з афішами статей про рекомендовані колекції одягу.

Рисунок 3.1 – Вигляд секції hero

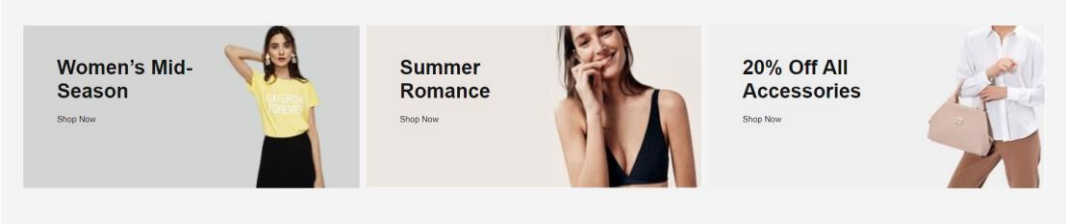

Рисунок 3.2 – Вигляд секції banners

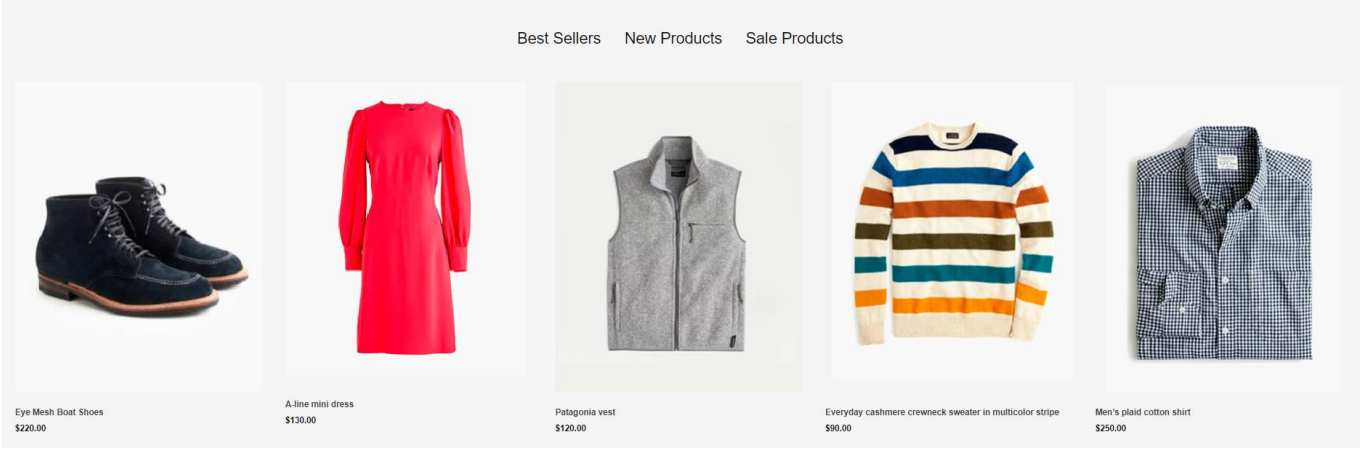

Рисунок 3.3 – Вигляд верхньої частини секції main-products

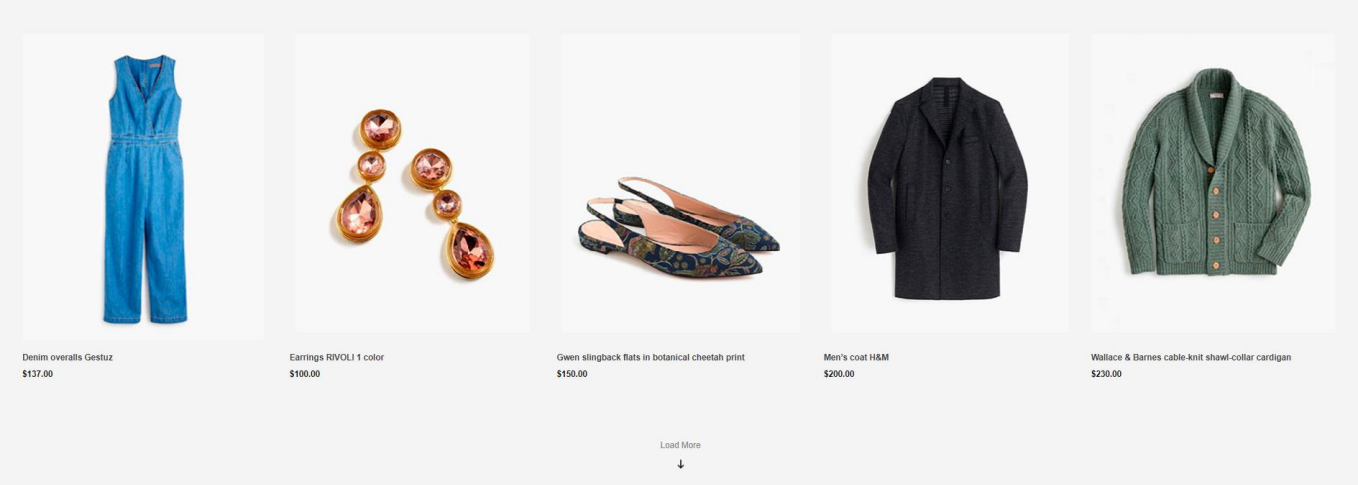

Рисунок 3.4 – Вигляд нижньої частини секції main-products

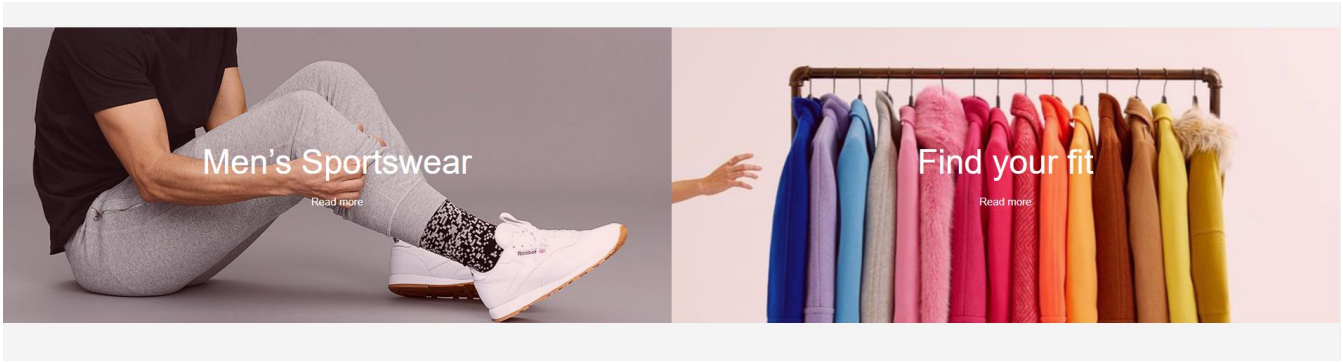

Рисунок 3.5 – Вигляд секції shop-banners

На сторінці каталогу товарів докладно показано всі товари які доступні на сайті. На сторінці доступні всі необхідні налаштування пошуку товару які б могли задовольнити покупця та знайти собі те що йому подобається. Для цього було обрано наступні секції:

- hero-catalog (рис. 3.6) – в цій секції розміщено банер на якому відображається рекомендації щодо актуальних колекцій;
- catalog-filter (рис. 3.7) – в цій секції відображається різноманітні варіанти фільтрації, за категорією, кольором та ціною;

– catalog-grid (рис. 3.8) – в цій секції відображається варіація відображення сітки товарів, сортування за ціною та останнім обраним елементом та можливість перейти на другу сторінку товарів.

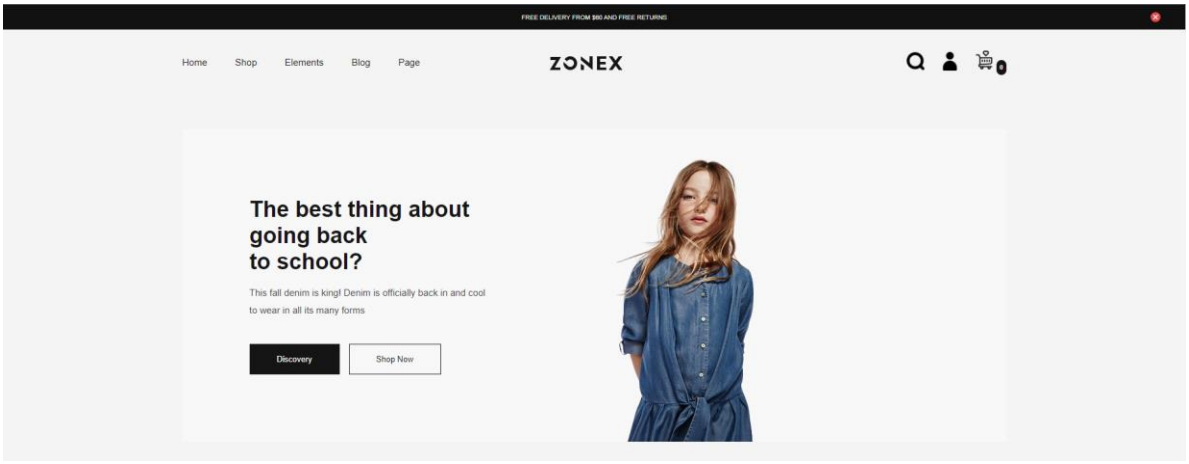

Рисунок 3.6 – Вигляд секції header-catalog

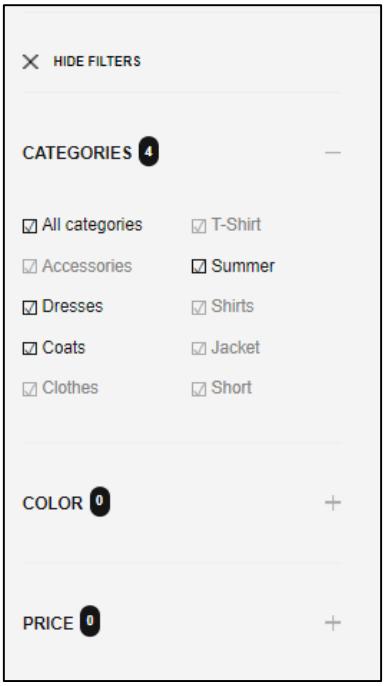

Рисунок 3.7 – Вигляд секції catalog-filter

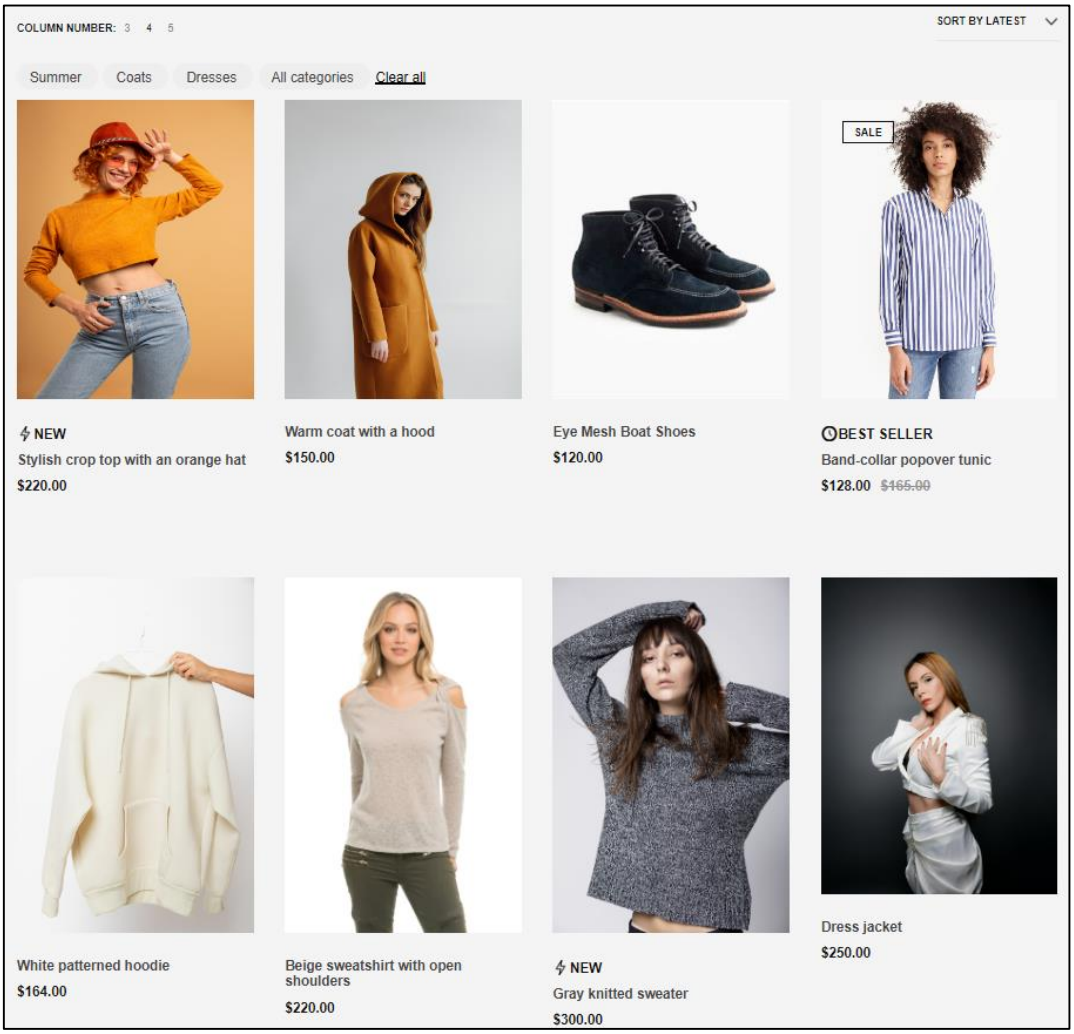

Рисунок 3.8 – Вигляд секції catalog-grid

На сторінці продукту відображається докладна інформація про обраний товар. На сторінці відображається графічний контент продукту, його ціна, назва, опис, кнопка додавання до кошику та до списку бажань (рис. 3.9). Також додано кнопки обрання розміру, кольору, та кількість замовлення бажаного товару.

Нижче головного блоку товару, розміщено додаткова інформація про товар, відгуки, та список товарів схожих на обраний (рис. 3.10).

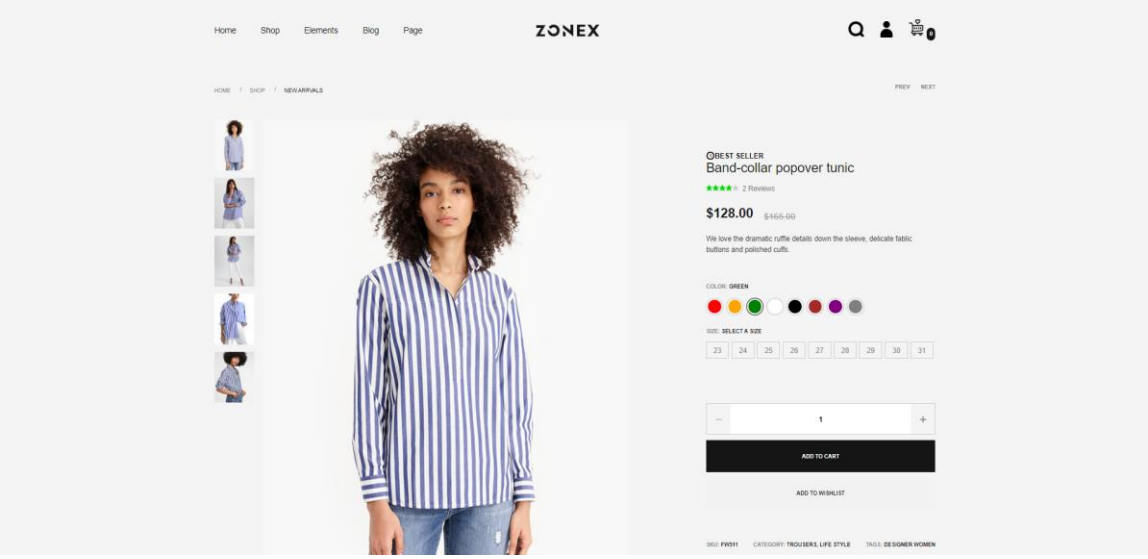

Рисунок 3.9 – Головний блок сторінки продукту

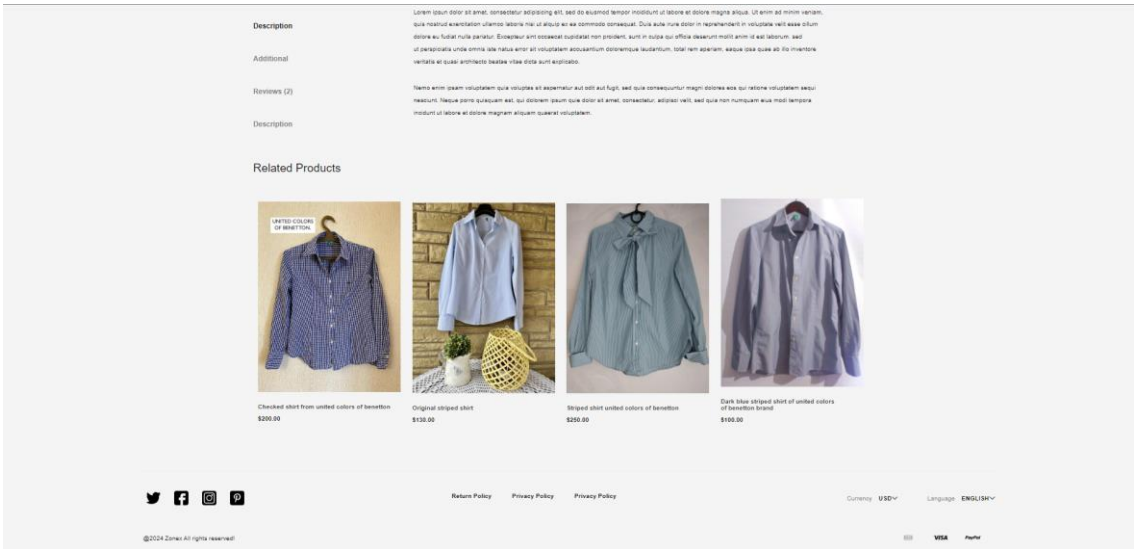

Рисунок 3.10 – Вигляд нижньої секції сторінки продукту

На сторінці кошика відображаються товари, додані до неї. Також є опис товару, які були додані, загальна кількість та сума купівлі. Нижче мається форма оформлення замовлення, та кнопки «Оплатити зараз», «Продовжити покупки» та «Зробити замовлення» (рис. 3.11).

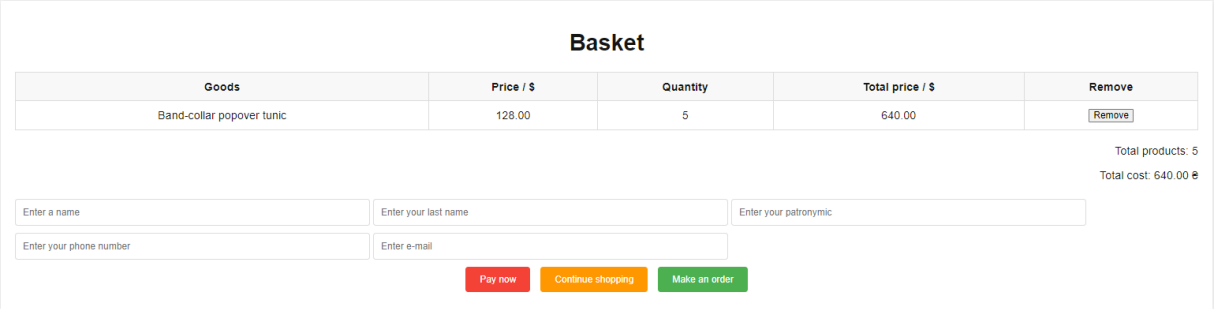

Рисунок 3.11 – Вигляд сторінки кошика

На рисунку 3.12 представлено лист про успішне оформлення замовлення.

Рисунок 3.12 – Вигляд готового шаблону листа про успішне оформлення замовлення

При розробці вебзастосунку важливим елементом також є створення сторінок реєстрації (рис. 3.13) та авторизації користувачів (рис. 3.14), щоб вони в свою чергу мали можливість зареєструвати обліковий запис і надалі використовувати його для входу в особистий кабінет, де відображатиметься інформація про поточне замовлення, історія замовлень і тп.

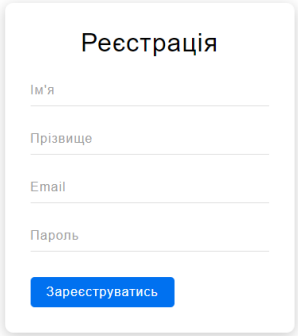

Рисунок 3.13 – Вигляд сторінки реєстрації

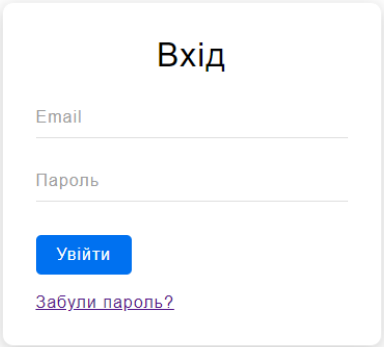

Рисунок 3.14 – Вигляд сторінки авторизації

На сайті також є унікальні секції, які відображаються на всіх сторінках, але в деяких можуть відрізнятись:

- announcement bar;
- header;
- footer.

Секція announcement bar (рис. 3.15) відображає важливу інформацію на сайті, про діючі розпродажі, або про знижку при доставці.

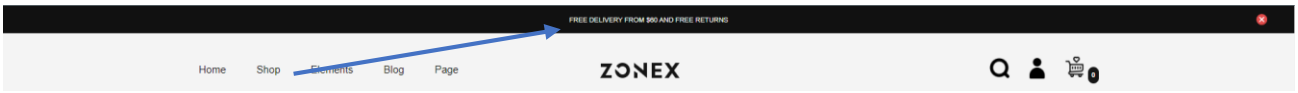

Рисунок 3.15 – Вигляд секції announcement bar

Шапка містить посилання на основні сторінки сайту (рис. 3.16).

Рисунок 3.16 – Вигляд секції header

Footer сайту містить коротку текстову інформацію, інформація про платіжні системи, а також змінити регіон або валюту (рис. 3.17 – 3.18). Також можна підписатись на розсилку новин сайту.

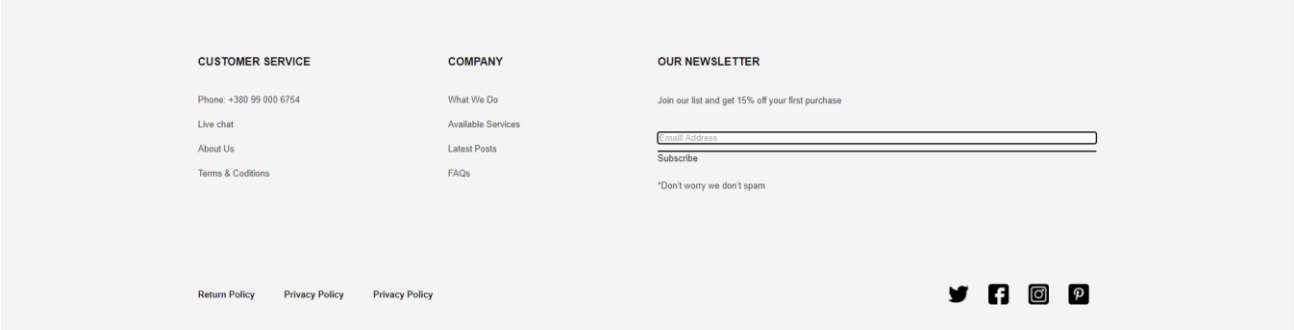

Рисунок 3.17 – Вигляд секції footer

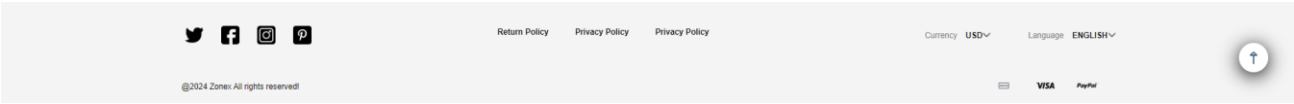

Рисунок 3.18 – Вигляд секції footer-2

3.2 Програмна реалізація функціоналу

3.2.1 Реалізація секції фільтрації товару

Після успішної верстки головних сторінок сайту, наступним завдання полягає в реалізації функціоналу. Для початку було зроблено фільтрацію товарів у секції catalog-filter.

Створюємо файл catalog-filter.html та оголошуємо в ньому основний контейнер, що буде містить всі фільтри та кнопки для приховування всіх фільтрів (рис. 3.19).

```
<div class="catalog-filters">
  <button class="hide-filters btn-reset">Hide Filters</button>
```

Рисунок 3.19 – Оголошення основного класу та кнопки

Далі виконуємо структуру кожного фільтра, в якому будуть знаходитись наступні елементи:

– : контейнер для окремого фільтра (з класом catalog-filter--open, який визначає, чи відкритий фільтр);

– : верхня частина фільтра, яка містить заголовок та кількість вибраних елементів;

– : містить заголовок фільтра та кількість вибраних елементів;

– <h3 class="catalog-filter__title">Categories</h3>: заголовок фільтра;

– 0: кількість вибраних елементів у фільтрі;

– : елемент для переключення стану (відкритий/закритий) фільтра;

– : нижня частина фільтра, що містить елементи для вибору;

– <ul class="catalog-filter__items">: список елементів для вибору.

Наступним кроком буде реалізація скрипта catalog-filter-toggle.js. Спочатку імпортуємо змінні з файлу «../_vars.js», які будуть використовуватись у коді. Робимо перевірку, щоб код виконувався лише якщо на сторінці є елемент з класом «.catalog» (рис. 3.20).

```
if (document.querySelector('.catalog')) {
```

Рисунок 3.20 – Виконання перевірки

Додаємо подію для відкривання/закривання фільтрів (рис. 3.21).

```
vars.$catalogFiltersTop.forEach(el => {
  el.addEventListener('click', (e) => {
    e.currentTarget.closest('.catalog-filter').classList.toggle('catalog-filter--open');
  });
});
```

Рисунок 3.21 – Подія відкривання закривання фільтрів

Для кожного заголовка фільтра додається обробник події ‘click’, який переключає клас catalog-filter--open, відкриваючи або закриваючи фільтр. Також додаємо обробник подій для кнопки «Hide Filters», щоб дана кнопка закривала всі фільтри, видаляючи клас catalog-filter--open з кожного фільтра (рис. 3.22).

```
vars.$hideFilters.addEventListener('click', (e) => {
  vars.$catalogFiltersTop.forEach(el => {
    el.closest('.catalog-filter').classList.remove('catalog-filter--open');
  });
});
```

Рисунок 3.22 – Закривання всіх фільтрів

Виконуємо обробник вибору елементів фільтрації (рис. 3.23). Для кожного елемента фільтра додається обробник подій «change»:

– при виборі (checked) додається клас custom-checkbox--active і створюється відповідний елемент вибору;

– при знятті вибору (unchecked) видаляється клас custom-checkbox--active і видаляється відповідний елемент вибору;

– оновлюється кількість вибраних елементів;

– якщо є активні елементи, показується блок виборів.

```
vars.$catalogFilterItems.forEach(el => {
  el.querySelector('input').addEventListener('change', (e) => {
    let checked = el.querySelector('input').checked;

    if (checked) {
      el.querySelector('.custom-checkbox').classList.add('custom-checkbox--active');
      let text = el.querySelector('.custom-checkbox__text').textContent;

      vars.$catalogChoice.insertAdjacentHTML('afterbegin', createChoiceItem(text));

    } else {
      el.querySelector('.custom-checkbox').classList.remove('custom-checkbox--active');

      let text = el.querySelector('.custom-checkbox').dataset.text;

      document.querySelector(`[data-choice-text="${text}"]`).remove();
    }

    el.closest('.catalog-filter').querySelector('.catalog-filter__quantity').textContent = el.closest('.catalog-filter__items').querySelectorAll('.custom-checkbox--active').length;

    let activeCheckboxes = document.querySelectorAll('.custom-checkbox--active');

    if (activeCheckboxes.length > 0) {
      vars.$catalogChoice.style.display = 'block';
    } else {
      vars.$catalogChoice.style.display = 'none';
    }

  });
```

Рисунок 3.23 – Обробник вибору елементів фільтрації

Останнім кроком було обробка подій на елементах вибору (рис. 3.24). Обробляє події ‘click’ на елементах вибору:

– видалення окремого вибраного елемента та скидання відповідного чекбокса;

– очищення всіх виборів при натисканні кнопки «Clear».

```
vars.$catalogChoice.addEventListener('click', (e) => {
  if (e.target.classList.contains('catalog-choice__item')) {
    e.target.remove();

    let text = e.target.textContent.trimLeft().trimRight();

    document.querySelector(`[data-text="${text}"]`).querySelector('input').checked = false;
    document.querySelector(`[data-text="${text}"]`).classList.remove('custom-checkbox--active');
  }

  if (e.target.classList.contains('catalog-choice__clear')) {
    Array.from(e.currentTarget.children).forEach(el => {
      if (!el.classList.contains('catalog-choice__clear')) {
        el.remove();
      }

      document.querySelectorAll('.catalog-filter__quantity').forEach(el => el.textContent = 0);

      vars.$catalogFilterItems.forEach(el => {
        el.querySelector('input').checked = false;
        el.querySelector('.custom-checkbox').classList.remove('custom-checkbox--active');
      });
    });

    e.currentTarget.style.display = 'none';
  }

  if (e.currentTarget.children.length === 1) {
    e.currentTarget.style.display = 'none';
  }
});
```

Рисунок 3.24 – Обробник подій на елементах вибору

Таким чином, попередні дії реалізовують функціонал для інтерактивного фільтрації каталогу на вебсторінці, забезпечуючи користувачам можливість зручного фільтрувати елементи за різними критеріями.

3.2.2 Реалізація вибору кольору та розміру продукту

Далі переходимо на сторінку товару. В першу чергу реалізуємо щоб покупець міг обрати колір бажаного товару. Для початку у файлі card-top.html оголошуємо контейнер для вибору кольору (рис. 3.25), в якому буде знаходитись:

– color-select: контейнер для вибору кольору;

– color-select__selected: показує вибраний колір. Початково встановлено "Green";

– color-select__list: список кольорових кнопок;

– color-select__item: контейнер для кожної кнопки кольору;

– color-select__btn: кнопка кольору. Встановлює колір за допомогою data-color і стилізує колір фону. Кнопка з класом color-select__btn--active показує, що цей колір вибраний (початково "Green").

```
<div class="card-info__select card-info__first-select color-select">
  <p class="color-select__selected">Color: <span>Green</span></p>
  <ul class="color-select__list">
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="red" aria-label="Choice red color" style="background-color: red;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="orange" aria-label="Choice orange color" style="background-color: orange;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset color-select__btn--active" data-color="green" aria-label="Choice green color" style="background-color: green;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="white" aria-label="Choice white color" style="background-color: white;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="black" aria-label="Choice black color" style="background-color: black;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="brown" aria-label="Choice brown color" style="background-color: brown;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="purple" aria-label="Choice purple color" style="background-color: purple;"></button></li>
    <li class="color-select__item"><button class="color-select__btn btn-reset" data-color="grey" aria-label="Choice grey color" style="background-color: grey;"></button></li>
  </ul>
</div>
```

Рисунок 3.25 – Контейнер для вибору кольору

Далі створюємо скрипт card-select.js, у якому імпортуємо змінні з файлу «../_vars.js». Після виконуємо обробку вибору кольору (рис. 3.26):

– if (vars.$colorSelect): перевірка, чи є елемент вибору кольору на сторінці;

– vars.$colorSelect.addEventListener('click', ...): додавання обробника подій на клік;

– if (e.target.classList.contains('color-select__btn')): перевірка, чи натиснута кнопка є кольоровою кнопкою;

– document.querySelectorAll('.color-select__btn').forEach(el => el.classList.remove('color-select__btn--active')): видалення класу color-select__btn--active з усіх кнопок кольору;

– let color = e.target.dataset.color: отримання значення кольору з атрибута data-color;

– e.currentTarget.querySelector('.color-select__selected span').textContent = color: оновлення тексту вибраного кольору;

– e.target.classList.add('color-select__btn--active'): додавання класу color-select__btn--active до натиснутої кнопки.

```
if (vars.$colorSelect) {
  vars.$colorSelect.addEventListener('click', (e) => {
    if (e.target.classList.contains('color-select__btn')) {

      document.querySelectorAll('.color-select__btn').forEach(el => el.classList.remove('color-select__btn--active'));

      let color = e.target.dataset.color;

      e.currentTarget.querySelector('.color-select__selected span').textContent = color;

      e.target.classList.add('color-select__btn--active');
    }
  });
}
```

Рисунок 3.26 – Обробник вибору кольору

Ця реалізація забезпечує динамічний вибір кольору для продукту на вебсторінці, дозволяючи користувачам візуально вибирати бажаний колір, а також відображати поточний вибраний колір.

Також не менш важливим елементом є вибір розміру товару (рис. 3.27). Для цього в тому ж файлі card-top.html створюємо контейнер для вибору розміру в якому будуть знаходитись блоки:

– size-select: контейнер для вибору розміру;

– size-select__selected: показує вибраний розмір. Початково встановлено "Select a size";

– size-select__list: список кнопок розміру;

– size-select__item: контейнер для кожної кнопки розміру;

– size-select__btn: кнопка розміру;

– size-select__clear: кнопка для скидання вибору розміру.

```
<div class="card-info__select size-select card-info__second-select">
  <p class="size-select__selected">Size: <span>Select a size</span></p>
  <ul class="size-select__list">
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 23 size">23</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 24 size">24</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 25 size">25</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 26 size">26</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 27 size">27</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 28 size">28</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 29 size">29</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 30 size">30</button></li>
    <li class="size-select__item"><button class="size-select__btn btn-reset" aria-label="Choice 31 size">31</button></li>
  </ul>
  <button class="btn-reset size-select__clear">
    <img src="../img/red-close.png" alt="close">
    clear
  </button>
</div>
```

Рисунок 3.27 – Контейнер для вибору розміру

Також в файлі card-select.js реалізовуємо вибір розміру (рис. 3.28):

– if (vars.$sizeSelect): перевірка, чи є елемент вибору розміру на сторінці;

– vars.$sizeSelect.addEventListener('click', ...): додавання обробника подій на клік;

– if (e.target.classList.contains('size-select__btn')): перевірка, чи натиснута кнопка є кнопкою розміру;

– e.currentTarget.querySelector('.size-select__clear').style.display = 'inline-flex': відображення кнопки "clear";

– let currentSize = e.target.textContent: отримання тексту вибраного розміру;

– e.target.classList.toggle('size-select__btn--active'): тоглування класу size-select__btn--active для натиснутої кнопки;

– if (e.target.classList.contains('size-select__btn--active')): якщо кнопка активна, додавання розміру до змінної size;

– size = size.replace(currentSize + ', ', ''): якщо кнопка неактивна, видалення розміру з змінної size;

– e.currentTarget.querySelector('.size-select__selected span').textContent = size: оновлення тексту вибраного розміру;

– if (!size): якщо жоден розмір не вибрано, встановлення тексту "Select a size";

– if (e.target.classList.contains('size-select__clear')): обробка кнопки "clear".

```
if (vars.$sizeSelect) {
  let size = '';

  vars.$sizeSelect.addEventListener('click', (e) => {
    if (e.target.classList.contains('size-select__btn')) {

      e.currentTarget.querySelector('.size-select__clear').style.display = 'inline-flex';

      let color = e.target.dataset.color;

      e.currentTarget.querySelector('.size-select__selected span').textContent = color;

      e.target.classList.toggle('size-select__btn--active');

      if (e.target.classList.contains('size-select__btn--active')) {
        let currentSize = e.target.textContent;

        size += currentSize + ', ';
      } else {
        let currentSize = e.target.textContent + ', ';

        size = size.replace(currentSize, '');
      }

      e.currentTarget.querySelector('.size-select__selected span').textContent = size;

      if (!size) {
        e.currentTarget.querySelector('.size-select__selected span').textContent = 'Select a size';
      }
    }

    if (e.target.classList.contains('size-select__clear')) {
      e.currentTarget.querySelector('.size-select__selected span').textContent = 'Select a size';
      document.querySelectorAll('.size-select__btn').forEach(el => el.classList.remove('size-select__btn--active'));
      e.target.style.display = 'none';
      size = '';
    }
  });

}
```

Рисунок 3.28 – Обробник вибору розміру

Ця реалізація забезпечує динамічний вибір розміру для продукту на вебсторінці, дозволяючи користувачам візуально вибирати бажаний розмір та відображати поточний вибраний розмір. Користувач також має можливість скинути вибір розміру.

3.2.3 Реалізація лічильника кількості товару

Ще одним етапом перед оформленням замовлення є обрати кількість бажаного продукту. Для цього потрібно реалізувати спеціальний лічильник. В тому ж файлі card-top.html оголошуємо контейнер для лічильника (рис. 3.29), в якому буде знаходитись:

- stepper: контейнер для лічильника;
- stepper__btn--minus: кнопка для зменшення кількості, початково вимкнена (має клас stepper__btn--disabled);

– stepper__input: поле введення для кількості товару, початкове значення встановлено на "1";

– stepper__btn--plus: кнопка для збільшення кількості.

```
<div class="card-info__stepper stepper">
  <button class="btn-reset stepper__btn stepper__btn--minus stepper__btn--disabled" aria-label="minus">
    <img src="../img/minus.png" alt="minus">
  </button>
  <input type="text" class="stepper__input" value="1" maxlength="5">
  <button class="btn-reset stepper__btn stepper__btn--plus" aria-label="plus">
    <img src="../img/plus.png" alt="plus">
  </button>
</div>
```

Рисунок 3.29 – Контейнер лічильника кількості товару

Далі створюємо скрипт stepper.js для обробки подій лічильника (рис. 3.30):

– if (vars.$stepper): перевірка, чи існує елемент лічильника на сторінці.

– const $stepperInput = vars.$stepper.querySelector('.stepper__input'): отримання елемента введення кількості;

– const $stepperMinus = vars.$stepper.querySelector('.stepper__btn--minus'): отримання кнопки для зменшення кількості;

– const $stepperPlus = vars.$stepper.querySelector('.stepper__btn--plus'): отримання кнопки для збільшення кількості;

– $stepperInput.addEventListener('keydown', (e) => { ... }): обробка події натискання клавіш на полі введення кількості;

– if (e.currentTarget.value <= 1): якщо кількість менше або дорівнює 1, кнопка зменшення вимикається, а кнопка збільшення вмикається;

– if (e.currentTarget.value > 99998): якщо кількість більше 99998, кнопка збільшення вимикається, а кнопка зменшення вмикається;

– $stepperPlus.addEventListener('click', (e) => { ... }): обробка події натискання кнопки збільшення кількості;

– let currentValue = parseInt($stepperInput.value): отримання поточного значення кількості;

– currentValue--: зменшення кількості на 1;

– $stepperInput.value = currentValue: встановлення нового значення в поле введення;

– $stepperPlus.classList.remove('stepper__btn--disabled'): увімкнення кнопки збільшення;

– if ($stepperInput.value <= 1): якщо значення менше або дорівнює 1, встановлюється мінімум 1, а кнопка зменшення вимикається.

```
import vars from '../_vars.js';

if (vars.$stepper) {

  const $stepperInput = vars.$stepper.querySelector('.stepper__input'),
        $stepperMinus = vars.$stepper.querySelector('.stepper__btn--minus'),
        $stepperPlus = vars.$stepper.querySelector('.stepper__btn--plus');

  $stepperInput.addEventListener('keydown', (e) => {
    if (e.currentTarget.value <= 1) {
      $stepperMinus.classList.add('stepper__btn--disabled');
      $stepperPlus.classList.remove('stepper__btn--disabled');
    } else {
      $stepperMinus.classList.remove('stepper__btn--disabled');
    }

    if (e.currentTarget.value > 99998) {
      $stepperMinus.classList.remove('stepper__btn--disabled');
      $stepperPlus.classList.add('stepper__btn--disabled');
    } else {
      $stepperPlus.classList.remove('stepper__btn--disabled');
    }
  });

  $stepperPlus.addEventListener('click', (e) => {
    let currentValue = parseInt($stepperInput.value);
    currentValue++;
    $stepperInput.value = currentValue;

    $stepperMinus.classList.remove('stepper__btn--disabled');

    if ($stepperInput.value > 99998) {
      $stepperInput.value = 99999;
      $stepperPlus.classList.add('stepper__btn--disabled');
    } else {
      $stepperPlus.classList.remove('stepper__btn--disabled');
    }
  });

  $stepperMinus.addEventListener('click', (e) => {
    let currentValue = parseInt($stepperInput.value);
    currentValue--;
    $stepperInput.value = currentValue;

    $stepperPlus.classList.remove('stepper__btn--disabled');

    if ($stepperInput.value <= 1) {
      $stepperInput.value = 1;
      $stepperMinus.classList.add('stepper__btn--disabled');
    } else {
      $stepperMinus.classList.remove('stepper__btn--disabled');
    }
  });
```

Рисунок 3.30 – Обробник подій лічильника

Ця реалізація забезпечує динамічний вибір кількості бажаного товару для замовлення продукту на вебсторінці, дозволяючи користувачам візуально вибирати і переглядати бажану та поточну кількість товару.

3.2.4 Реалізація додавання товару до кошика

Наступним кроком буде реалізація додавання товару до кошика. Для цього створюємо кнопку кнопку «Add to cart» та оголошуємо для неї скрипт обробки подій (рис. 3.31), у якому буде знаходитись:

- document.addEventListener('DOMContentLoaded', function() { ... }); – код виконується після завантаження всієї сторінки;
- const addToCartBtns = document.querySelectorAll('.add-to-cart-btn'); – знаходить всі кнопки додавання до кошика;
- addToCartBtns.forEach(btn => { ... }); – додає обробник подій для кожної кнопки;
- btn.addEventListener('click', function() { ... }); – встановлює обробник подій для кнопки;
- const productElement = btn.closest('.product'); – знаходить найближчий батьківський елемент з класом .product;
- const title = productElement.querySelector('.product-title').textContent; – зчитує назву продукту;
- const price = parseFloat(productElement.querySelector('.product-price span').textContent); – зчитує та перетворює ціну в число з плаваючою точкою;
- const quantity = parseInt(productElement.querySelector('.stepper__input').value); – зчитує та перетворює кількість товару в ціле число;
- const cartItems = JSON.parse(localStorage.getItem('cartItems')) || []; – завантажує наявні товари з кошика або ініціалізує порожній масив;

– const existingItem = cartItems.find(item => item.title === title); – перевіряє, чи існує товар вже у кошику;

– if (existingItem) { ... } else { ... } – якщо товар вже є, оновлює кількість. Якщо ні, додає новий товар;

– localStorage.setItem('cartItems', JSON.stringify(cartItems)); – зберігає оновлений кошик у Local Storage;

– alert('Товар додано до кошика'); – відображає повідомлення про успішне додавання товару.

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    const addToCartBtn = document.querySelector('.card-info__btn--tocart');
    const itemTitle = document.querySelector('.card-info__title').textContent;
    const itemPrice = parseFloat(document.querySelector('.info-price__current').textContent.replace('$', ''));
    const quantityInput = document.querySelector('.stepper__input');

    addToCartBtn.addEventListener('click', function() {
      const quantity = parseInt(quantityInput.value, 10);
      const cartItems = JSON.parse(localStorage.getItem('cartItems')) || [];

      const existingItemIndex = cartItems.findIndex(item => item.title === itemTitle);
      if (existingItemIndex > -1) {
        cartItems[existingItemIndex].quantity += quantity;
      } else {
        cartItems.push({ title: itemTitle, price: itemPrice, quantity: quantity });
      }

      localStorage.setItem('cartItems', JSON.stringify(cartItems));
      alert('Товар додано до кошика');
    });
  });
  </script>
```

Рисунок 3.31 – Скрипт обробки кнопки Add to cart

Далі переходимо до файлу cart.html та додаємо до нього скрипт прийому товару який був доданий до кошика з такими елементами (рис. 3.32):

– document.addEventListener('DOMContentLoaded', function() { ... }); – код виконується після завантаження всієї сторінки;

– const cartItems = JSON.parse(localStorage.getItem('cartItems')) || []; – завантажує наявні товари з кошика або ініціалізує порожній масив;

- const cartItemsContainer = document.getElementById('cart-items'); – отримує контейнер для товарів кошика;
- const totalItemsSpan = document.getElementById('total-items'); – отримує елемент для відображення загальної кількості товарів;
- const totalCostSpan = document.getElementById('total-cost'); – отримує елемент для відображення загальної вартості товарів;
- let totalItems = 0; – ініціалізує змінну для зберігання загальної кількості товарів;
- let totalCost = 0; – ініціалізує змінну для зберігання загальної вартості товарів;
- cartItems.forEach(item => { ... }); – перебирає всі товари в кошику;
- const itemTotalPrice = item.price * item.quantity; – обчислює загальну вартість для кожного товару;
- totalItems += item.quantity; – додає кількість товару до загальної кількості;
- totalCost += itemTotalPrice; – додає загальну вартість товару до загальної вартості;
- const row = document.createElement('tr'); – створює новий рядок таблиці для товару;
- row.innerHTML = ...` – заповнює рядок таблиці даними товару;
- const removeBtn = row.querySelector('.remove-item-btn'); - отримує кнопку видалення товару;
- removeBtn.addEventListener('click', function() { ... }); – додає обробник подій для кнопки видалення;
- cartItemsContainer.appendChild(row); – додає рядок до контейнера для товарів;
- totalItemsSpan.textContent = totalItems; – оновлює відображення загальної кількості товарів;

– totalCostSpan.textContent = totalCost.toFixed(2); – оновлює відображення загальної вартості товарів;

– function removeItemFromCart(title) { ... } – функція для видалення товару з кошика;

– const updatedCartItems = cartItems.filter(item => item.title !== title); – оновлює масив товарів, виключаючи видалений товар;

– localStorage.setItem('cartItems', JSON.stringify(updatedCartItems)); – зберігає оновлений кошик у Local Storage;

– location.reload(); – перезавантажує сторінку для оновлення відображення.

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    const cartItems = JSON.parse(localStorage.getItem('cartItems')) || [];
    const cartItemsContainer = document.getElementById('cart-items');
    const totalItemsSpan = document.getElementById('total-items');
    const totalCostSpan = document.getElementById('total-cost');

    let totalItems = 0;
    let totalCost = 0;

    cartItems.forEach(item => {
      const itemTotalPrice = item.price * item.quantity;
      totalItems += item.quantity;
      totalCost += itemTotalPrice;

      const row = document.createElement('tr');
      row.innerHTML = `
        <td>${item.title}</td>
        <td>${item.price.toFixed(2)}</td>
        <td>${item.quantity}</td>
        <td>${itemTotalPrice.toFixed(2)}</td>
        <td><button class="remove-item-btn">Видалити</button></td>
      `;

      const removeBtn = row.querySelector('.remove-item-btn');
      removeBtn.addEventListener('click', function() {
        removeItemFromCart(item.title);
      });

      cartItemsContainer.appendChild(row);
    });

    totalItemsSpan.textContent = totalItems;
    totalCostSpan.textContent = totalCost.toFixed(2);

    function removeItemFromCart(title) {
      const updatedCartItems = cartItems.filter(item => item.title !== title);
      localStorage.setItem('cartItems', JSON.stringify(updatedCartItems));
      location.reload();
    }
  });
  </script>
```

Рисунок 3.32 – Скрипт обробки отриманого товару у кошик

Останнім етапом буде відправлення листа про успішне оформлення замовлення. Для цього використаєм онлайн-сервіс EmailJS. Заходимо до сервісу, реєструємось. Після успішної реєстрації налаштовуємо службу електронної пошти, де вказуємо назву служби, ID-код та прив'язуємо пошту від якої будуть надсилатись повідомлення (рис. 3.33).

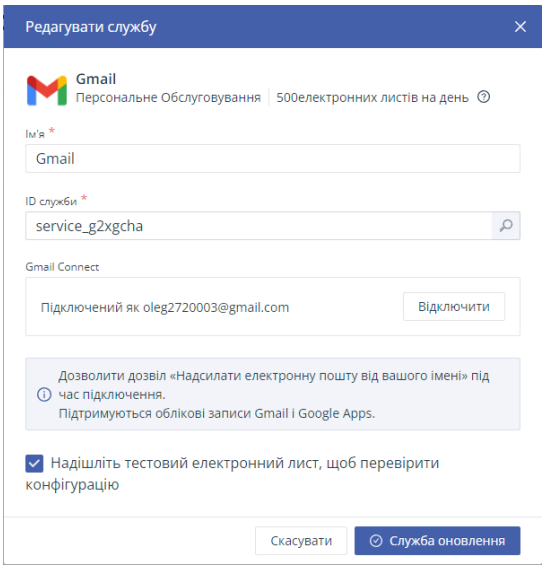

Рисунок 3.33 – Налаштування служби електронної пошти

Далі переходимо у вкладку «Шаблони електронної пошти» та пишемо шаблон для успішного підтвердження замовлення (рис. 3.34).

Рисунок 3.34 – Налаштування шаблону

Після всіх налаштувань переходимо до файлу кошика та реалізовуємо скрипт відправки листа для сервісу EmailJS (рис. 3.35):

- <script type="text/javascript" src="https://cdn.emailjs.com/dist/email.min.js"></script> – підключення бібліотеки EmailJS;
- (function(){ emailjs.init("QvNVek8Wgs3NVoHeK"); })(); – ініціалізація EmailJS з користувацьким ID;
- document.addEventListener('DOMContentLoaded', function() { ... }); – код виконується після завантаження всієї сторінки;
- const placeOrderBtn = document.querySelector('.place-order-btn'); – отримує кнопку оформлення замовлення;
- placeOrderBtn.addEventListener('click', function(event) { ... }); – додає обробник подій для кнопки оформлення замовлення;
- event.preventDefault(); – зупиняє стандартну поведінку кнопки;
- const firstName = document.querySelector('input[placeholder="Введіть ім\'я"]').value; – зчитує значення введених даних з відповідних полів;
- if (!firstName || !lastName || !middleName || !phone || !email) { alert('Будь ласка, заповніть усі поля'); return; } – перевірка, чи всі поля заповнені. Якщо ні, виводить попередження;
- const cartItems = JSON.parse(localStorage.getItem('cartItems')) || []; – завантажує наявні товари з кошика або ініціалізує порожній масив;
- const itemsList = cartItems.map(item => – ${item.title}: ${item.quantity} шт. по ${item.price} ₴).join('\n'); - формує список товарів для відправлення;
- const totalCost = cartItems.reduce((sum, item) => sum + item.price * item.quantity, 0); – обчислює загальну вартість товарів;
- const templateParams = { ... }; – формує параметри для шаблону EmailJS;
- emailjs.send('service_g2xgcha', 'template_ixoypmw', templateParams) – відправляє email за допомогою EmailJS;

– .then((response) => { ... }, (error) => { ... }); – обробка результату відправлення: успішне відправлення або помилка.

```
<script type="text/javascript" src="https://cdn.emailjs.com/dist/email.min.js"></script>
<script type="text/javascript">
  (function(){
      emailjs.init("QvNVek8Wgs3NVoHeK");
  })();

  document.addEventListener('DOMContentLoaded', function() {
      const placeOrderBtn = document.querySelector('.place-order-btn');
      placeOrderBtn.addEventListener('click', function(event) {
          event.preventDefault();

          const firstName = document.querySelector('input[placeholder="Введіть ім\'я"]').value;
          const lastName = document.querySelector('input[placeholder="Введіть прізвище"]').value;
          const middleName = document.querySelector('input[placeholder="Введіть по батькові"]').value;
          const phone = document.querySelector('input[placeholder="Введіть телефон"]').value;
          const email = document.querySelector('input[placeholder="Введіть e-mail"]').value;

          if (!firstName || !lastName || !middleName || !phone || !email) {
              alert('Будь ласка, заповніть усі поля');
              return;
          }

          const cartItems = JSON.parse(localStorage.getItem('cartItems')) || [];
          const itemsList = cartItems.map(item => `- ${item.title}: ${item.quantity} шт. по ${item.price} ₴`).join('\n');
          const totalCost = cartItems.reduce((sum, item) => sum + item.price * item.quantity, 0);

          const templateParams = {
              firstName: firstName,
              lastName: lastName,
              middleName: middleName,
              phone: phone,
              email: email,
              itemsList: itemsList,
              totalCost: totalCost
          };

          emailjs.send('service_g2xgcha', 'template_ixoypmw', templateParams)
              .then((response) => {
                  console.log('SUCCESS!', response.status, response.text);
                  alert('Замовлення успішно оформлено. Перевірте вашу пошту.');
                  localStorage.removeItem('cartItems');
                  window.location.reload();
              }, (error) => {
                  console.log('FAILED...', error);
                  alert('Виникла помилка при відправці листа. Спробуйте пізніше.');
              });
      });
```

Рисунок 3.35 – Скрипт відправки листа на пошту

3.2.5 Реалізація форми реєстрації та авторизації

Для реалізації форми реєстрації на початку було створено HTML форму (рис. 3.36), основні елементи які включають:

– поля вводу для імені, прізвище, email та паролю;

– кнопка для відправки форми;

– форма надсилає дані методом post до register.php.

```html
<body>

    <form class="form" action="register.php" method="post">

        <h1 class="form__title">Реєстрація</h1>

        <div class="form__group">
            <input class="form__input" type="text" name="name" required placeholder=" ">
            <label class="form__label">Ім'я</label>
        </div>

        <div class="form__group">
            <input class="form__input" type="text" name="surname" required placeholder=" ">
            <label class="form__label">Прізвище</label>
        </div>

        <div class="form__group">
            <input class="form__input" type="email" name="email" required placeholder=" ">
            <label class="form__label">Email</label>
        </div>

        <div class="form__group">
            <input class="form__input" type="password" name="password" required placeholder=" ">
            <label class="form__label">Пароль</label>
        </div>

        <button class="form__button" type="submit">Зареєструватись</button>

    </form>
</body>
```

Рисунок 3.36 – Форма реєстрації

Далі реалізовується PHP-скрипт який буде обробляти дані реєстрації (рис. 3.37):

- підключення до бази даних через db.php;
- отримує дані з форм реєстрації;
- хешує пароль за допомогою password_hash;
- вставляє новий запис користувача у таблицю user;
- виводить повідомлення про успіх або помилку.

```php
<?php
require_once('db.php');

$name = $_POST['name'];
$surname = $_POST['surname'];
$email = $_POST['email'];
$password = $_POST['password'];

$hashed_password = password_hash($password, PASSWORD_DEFAULT);

$sql = "INSERT INTO `users` (name,surname,email,password) VALUES ('$name', '$surname', '$email', '$hashed_password')";

if ($conn -> query($sql) === TRUE) {
    echo "Ви успішно зареєструвались";
} else {
    echo "Помилка ". $conn->error;
}

$conn->close();
?>
```

Рисунок 3.37 – Скрипт реалізації форми реєстрації

Так же створюємо HTML-форму для авторизації користувача (рис. 3.38), з наступними елементами:

- поля вводу для email та паролю;
- кнопка для відправки форми;
- форма надсилає дані методом post до login.php.

```
<body>

    <form class="form" action="login.php" method="post">

        <h1 class="form__title">Вхід</h1>

        <div class="form__group">
            <input class="form__input" type="email" name="email" required placeholder=" ">
            <label class="form__label">Email</label>
        </div>

        <div class="form__group">
            <input class="form__input" type="password" name="password" required placeholder=" ">
            <label class="form__label">Пароль</label>
        </div>

        <button class="form__button" type="submit">Увійти</button><br>

        <br><a href="forgot.php">Забули пароль?</a>

    </form>
</body>
```

Рисунок 3.38 – Форма авторизації користувача

Реалізовуємо PHP-скрипт що обробляє дані авторизації (рис. 3.39):

- підключається до бази даних через db.php;
- отримає дані з форми авторизації;
- виконує запит до таблиці users, щоб знайти користувача з введеним email;
- якщо користувач знайдений, перевіряє пароль за допомогою password_verify;
- виводить повідомлення про успіх або помилку.

```php
<?php
require_once('db.php');

$email = $_POST['email'];
$password = $_POST['password'];

$sql = "SELECT * FROM `users` WHERE email = '$email'";
$result = $conn->query($sql);

if ($result->num_rows > 0) {
    while($row = $result->fetch_assoc()) {
        // Перевірка паролю з хешованим паролем у базі даних
        if (password_verify($password, $row['password'])) {
            echo "Ласкаво просимо! " . $row['name'];
        } else {
            echo "Неправильний пароль";
        }
    }
} else {
    echo "Такого користувача не існує";
}
?>
```

Рисунок 3.39 – Скрипт обробки форми авторизації

На рисунок 3.40 продемонстровано підключення таблиці бази даних з основними полями до вебзастосунку.

```php
<?php

$servername = "localhost";
$username = "root";
$password = "";
$dbname = "registerUser";
$dbname_two = "products";

$conn = mysqli_connect($servername, $username, $password, $dbname);

if (!$conn) {
    die("Connection fialed". mysqli_connect_error());
} else {

} ?>
```

Рисунок 3.40 – Скрипт підключення бази даних до форм

Висновки до розділу 3

У третьому розділі було описано виконання розробки вебзастосунку. Реалізовано веб-застосунок з продажу одягу з використанням технологій HTML, SCSS, JavaScript, PHP, MySQL, Open Server, NPM та Gulp. Виконано верстку головних сторінок сайту, можливість обрати клієнту розмір та колір бажаного товару, сторінку кошика, відправлення листа підтвердження реєстрації та авторизації.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розроблено вебзастосунок з продажу одягу з виконанням поставлених раніше завдань. Виконано аналіз предметної області, а також досліджено методи та технології для розробки програмного забезпечення. Було виконано порівняння існуючих сайтів-аналогів магазину одягу, досліджені переваги та недоліки кожного з них. Реалізовано усі функції, що вирішують поставлені проблеми для клієнта що використовує даний вебзастосунок. Для успішної реалізації було виконано модулювання програмного забезпечення через блок-схеми UML-діаграми. Описано усі засоби розробки ПЗ та застосовані методи. Тому, усі поставлені завдання виконані успішно з дотриманням усіх поставлених вимог.

Використовуючи вебзастосунок з продажу одягу користувач може здійснити замовлення, не виходячи з дому. Враховуючи суворі сучасні реалії – це велика перевага перед фізичними магазинами одягу. Також, даний застосунок може автоматизувати процес продажу товарів, що значно зекономить час на реалізацію товару та зменшити витрати наприклад на оренду приміщення.

Розроблений застосунок має зручний та зрозумілий у використанні інтерфейс. Для досягнення мети було використано сучасні технології та методи.

Тому представлений вебзастосунок з продажу одягу зекономить час та допоможе користувачам здійснювати замовлення, не виходячи з дому та наражаючи себе на небезпеку.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Personalization in E-commerce. URL: https://www.shopify.com/enterprise/blog/ecommerce-personalization-examples (дата звернення: 01.05.2024).

2. The Role of Social Media in E-commerce. URL: https://www.locate2u.com/ecommerce/the-role-of-social-media-in-e-commerce-marketing/ (дата звернення: 01.05.2024).

3. Дакетт Д., HTML та CSS: Дизайн та розробка вебсайтів. Вілямс, 2018. 480 с. URL: http://vk.academy.lv/file/Dakett_J_HTML_i_CSS_Razrabotka_i_dizayn_web_stranic.pdf (дата звернення: 05.05.2024).

4. Фленаган Д., JavaScript: Докладний посібник. Шосте видання. Москва, 2020. 1080 с. URL: https://my-js.org/assets/files/definitive_guide-7eea8a7e5bab411a133ba9d0204c214a.pdf (дата звернення: 05.05.2024).

5. Скляр Д., Вивчаємо PHP 7. Посібник зі створення інтеративних веб-сайтів. Київ, 2017. 382 с. URL: https://library-it.com/wp-content/uploads/2021/05/izuchaem_php_7_rukovodstvo.pdf (дата звернення: 06.05.2024).

6. Документація EmailJS. Вебсайт. URL: https://www.emailjs.com/docs/ (дата звернення: 06.05.2024).

7. Документація NPM. Вебсайт. URL: https://docs.npmjs.com/ (дата звернення: 06.05.2024).

8. Gulp для початківців. URL: https://itproger.com/ua/course/gulp (дата звернення: 06.05.2024).

9. Open Server Panel для початківців. Документація. URL: https://ospanel.io/docs/ (дата звернення: 06.05.2024).

10. ASOS. URL: https://www.asos.com/us/ (дата звернення: 08.05.2024).

11. Zalando. URL: https://zalando.com/ (дата звернення: 08.05.2024).

12. Farfetch. URL: https://www.farfetch.com/ (дата звернення: 08.05.2024).

13. UML Diagram Types Guide. Learn About All types of UML Diagrams with Examples: веб-сайт. URL: https://creately.com/blog/diagrams/uml-diagram-types-examples/ (дата звернення: 10.05.2024).

14. The Role of Social Media in E-commerce. URL: https://www.locate2u.com/ecommerce/the-role-of-social-media-in-e-commerce-marketing/ (дата звернення: 01.05.2024).

15. The Importance of Mobile Optimization in E-commerce. https://www.bigcommerce.com/blog/mobile-ecommerce-seo/ (дата звернення: 04.05.2024).

16. Web Design Trends 2024. URL: https://webflow.com/made-in-webflow/2024 (дата звернення: 10.05.2024).

17. E-commerce Marketing Strategies. URL: https://www.constantcontact.com/features/ecommerce (дата звернення: 10.05.2024).

18. The User Experience in E-commerce. URL: https://www.nngroup.com/reports/ecommerce-user-experience/ (дата звернення: 10.05.2024).

19. Браун Е., Вивчаємо JavaScript. Керівництво по створенню сучасних вебсайтів. Київ, 2018. 363 с. URL: https://mykhailoeoneoff.wordpress.com/wp-content/uploads/2018/01/javascript.pdf (дата звернення: 08.06.2024).

20. Мартін Р., Чистий код. Створення, аналіз і рефакторинг. Мінськ, 2019. 464 с. URL: https://coollib.cc/b/486351-robert-sesil-martin-chistyiy-kod-sozdanie-analiz-i-refaktoring/read (дата звернення: 08.06.2024).

21. Ніксон Р., Створюємо динамічні вебсайти за допомогою PHP, MySQL, JavaScript, CSS та HTML5. Санкт-Петербург, 2016. 768 с. URL: https://booster.by/files/oeu.pdf (дата звернення: 08.06.2024).

22. Кантелон М., Хартер М., Головайчук Т., Райлих Н., Node.js в дії. Москва, 2014. 548 с. URL: https://library-it.com/frameworks/node-js-v-dejstvii/ (дата звернення: 08.06.2024).

23. Сімпсон К., You Don’t Know JS: Scope & Closures. Токіо, 2015. 100 с. URL: https://www.litres.ru/kyle-simpson/you-don-t-know-js-scope-closures-6832865/?lfrom=236997940 (дата звернення: 09.06.2024).

24. Ерик А., Вейл Е., CSS: The Definitive Guide. Бостон, 2018. 1088 с. URL: https://www.rulit.me/books/css-the-definitive-guide-download-648878.html (дата звернення: 09.06.2024).

25. Фрейн. Б., Розробка сайтів для любих браузерів та пристроїв. Харків, 2014. 304 с. URL: https://library.kre.dp.ua/Books/2-4%20kurs/%D0%9C%D0%BE%D0%B2%D0%B8%20%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D1%83%D0%B2%D0%B0%D0%BD%D0%BD%D1%8F/HTML5_i_CSS3_Razrabotka_saytov_dlya_lyubykh_brauze.pdf (дата звернення: 09.06.2024).