МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
ФАКУЛЬТЕТ КОМП'ЮТЕРНИХ НАУК ТА ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач випускової кафедри
_____ Аліна САВЧЕНКО
«___»_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ МАГІСТР
ЗА ОСВІТНЬО-ПРОФЕСІЙНОЮ ПРОГРАМОЮ
«ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ПРОЕКТУВАННЯ»

Тема: «Вебзастосунок продажу домашнього одягу на базі ОС "Windows"»

Виконавець: Максим РИМАРЕНКО

Керівник: к.т.н. Сергій ВОДОП'ЯНОВ

Нормоконтролер: к.т.н., доцент Олена ТОЛСТІКОВА

КИЇВ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних наук та технологій*
Кафедра *комп'ютерних інформаційних технологій*
Спеціальність *122 «Комп'ютерні науки»*
Освітньо-професійна програма *«Інформаційні технології проектування»*

ЗАТВЕРДЖУЮ:
завідувач кафедри КІТ
_____Аліна САВЧЕНКО
(підпис)

«____»_____2023 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи
Римаренко Максима Віталійовича
(ПІБ випускника)

1. Тема роботи: «Вебзастосунок продажу домашнього одягу на базі ОС "Windows"» затверджена наказом ректора № 1976/ст від 29.09.2023р.

2. Термін виконання роботи: з 02 жовтня 2023 року по 31 грудня 2023 року.

3. Вихідні дані до роботи: вебзастосунок на мові програмування HTML для продажу домашнього одягу.

4. Зміст пояснювальної записки: 1. Аналіз та поняття технології. 2. Проектування вебзастосунку. 3. Розробка та тестування вебзастосунку.

4. Перелік обов'язкового ілюстративного матеріалу: 1. Поняття роботи вебзастосунку. 2. Технічний стек. 3. Дизайн та функціонал вебзастосунку. 4. Підготовка даних. 5. Реалізація функцій вебзастосунку. 6. Реалізація функції «Перегляд та пошук товарів». 7. Реалізація функції «Додавання товарів у кошик». 8. Розробка та тестування вебзастосунку. 9. Демонстрація роботи «Головної сторінки». 10. Демонстрація роботи «Аутентифікації, авторизації та реєстрації».

6. Календарний план-графік

№ з/п	Завдання	Термін виконання	Підпис керівника
1.	Аналіз предметної області та огляд аналогів. Написання 1 розділу, аналіз та поняття технології.	02.10.2023-16.10.2023	
2.	Вибір та опис використаних технологій. Написання 2 розділу, проектування застосунку.	17.10.2023-30.10.2023	
3.	Написання 3 розділу, розробка та тестування застосунку.	31.10.2023-14.11.2023	
4.	Загальне редагування та друк пояснювальної записки.	15.11.2023-20.11.2023	
5.	Проходження нормоконтролю, перепліт пояснювальної записки.	16.11.2023-20.10.2023	
6.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації.	19.12.2023-22.12.2023	

7. Дата видачі завдання 02.10.2023 р.

Керівник кваліфікаційної роботи _____ Сергій ВОДОП'ЯНОВ
(підпис керівника)

Завдання прийняв до виконання _____ Максим РИМАРЕНКО
(підпис випускника)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи на тему: «Вебзастосунок продажу домашнього одягу на базі ОС "Windows"» містить: 89 сторінок, 23 рисунки, 20 інформаційних джерел, 1 додаток.

Об'єкт дослідження – процес створення вебзастосунку на базі ОС "Windows".

Предмет дослідження – аналіз методів та засобів розробки вебзастосунку продажу домашнього одягу.

Мета кваліфікаційної роботи – отримати готовий вебзастосунок для продажу домашнього одягу на операційній системі "Windows" .

Методи дослідження – мова програмування HTML, CSS,JavaScript, інтегроване середовище розробки VSCode**.**

Результати кваліфікаційної роботи рекомендується використовувати для демонстрації роботи вебзастосунку для продажу домашнього одягу, та в подальшій інтеграції у компанії та підприємстві.

ВЕБДОДАТОК, БЕКЕНД, ФРОНТЕНД, БЕЗПЕКА , HTML, CSS ,API, JAVASCRIPT.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СКОРОЧЕНЬ, ТЕРМІНІВ

HTML - Hypertext Markup Language

CSS - Cascading Style Sheets

JS - JavaScript

API - Application Programming Interface

URL - Uniform Resource Locator

HTTP - Hypertext Transfer Protocol

ВСТУП

У епоху стрімкого розвитку технологій та змін у споживчих уподобаннях, роль онлайн-торгівлі визначено рішучою зручністю та доступністю. Цей контекст спонукає до роздумів та досліджень у сфері створення вебзастосунків, що не лише відповідають потребам споживачів, але й встановлюють нові стандарти у віртуальних торгівлях. У цьому контексті, дана дипломна робота націлена на створення та дослідження вебзастосунку для продажу домашнього одягу на базі операційної системи "Windows".

Сучасні тенденції свідчать про те, що електронна комерція в сфері моди стає ключовим елементом способу життя сучасного споживача. Вибір домашнього одягу стає не тільки питанням стилю, але й важливим вираженням індивідуальності. Тому розробка вебзастосунку, який допомагає ефективно вибирати та придбавати домашній одяг онлайн, має особливу важливість для якісного відгуку на потреби споживачів.

Виходячи із власного досвіду та пізнання потреб ринку, вірю, що вебзастосунок має стати не просто платформою для продажу товарів, але і вражаючим інструментом, який заохочує інтерактивність та особисте зв'язку з клієнтом. Це повинно бути не лише місце для покупок, але й простір, де кожен клієнт відчуває увагу та індивідуальний підхід.

Мета кваліфікаційної роботи полягає в дослідженні, розробці та імплементації вебзастосунку для продажу домашнього одягу на базі операційної системи "Windows". Основні аспекти цієї мети включають:

1. Аналіз Тенденцій та Потреб Ринку: Дослідження сучасних тенденцій у галузі електронної комерції та специфічних потреб споживачів у сегменті домашнього одягу.

2. Проектування та Розробка Інтерфейсу: Створення інтуїтивного та ефективного інтерфейсу вебзастосунку, який враховує особливості та очікування цільової аудиторії.

3. Реалізація Функціональності: Розробка необхідної функціональності для ефективних онлайн-продажів домашнього одягу, включаючи каталог товарів, кошик покупок, систему знижок та оновлення товарів.

4. Аналіз Ефективності та Користуваності: Оцінка продуктивності та зручності вебзастосунку, зокрема, вивчення реакцій користувачів та здатності застосунку задовольняти їхні потреби.

5. Впровадження Інновацій: Внесення новаторських рішень, які забезпечать високий рівень користувацького досвіду та відмінності від інших платформ електронної комерції.

Мета кваліфікаційної роботи полягає не лише в технічному розробленні вебзастосунку, але й в створенні інноваційного середовища для ефективної та задовільної онлайн-купівлі домашнього одягу.

Об'єктом досліджень є вебзастосунок, розроблений для продажу домашнього одягу та побудований на операційній системі "Windows". В рамках кваліфікаційної роботи об'єктом аналізу та вивчення є сам процес розробки та впровадження цього вебзастосунку з метою забезпечення ефективних онлайн-продажів домашнього одягу.

Важливо розглядати вебзастосунок як цілісний продукт, що включає в себе різноманітні компоненти:

1. Інтерфейс користувача: Дизайн та функціональність веб-сторінок, що складають вебзастосунок, зокрема головної сторінки, сторінок каталогу товарів, кошика покупок та інших.

2. База даних: Зберігання та управління інформацією про товари, користувачів, замовлення та інші аспекти електронної комерції.

3. Функціональність: Реалізація основних опцій, таких як додавання товарів у кошик, обробка замовлень, керування каталогом товарів та виведення акцій та знижок.

4. Забезпечення безпеки: Заходи для захисту конфіденційності та цілісності даних користувачів та забезпечення безпеки фінансових транзакцій.

5. Взаємодія з операційною системою "Windows": Врахування особливостей та можливостей даної операційної системи для оптимізації роботи вебзастосунку.

Об'єкт дослідження охоплює як технічні аспекти розробки вебзастосунку, так і аспекти користувальницького досвіду та маркетингові аспекти, спрямовані на забезпечення успішності продажів домашнього одягу через цей онлайн-канал.

Актуальність теми про розробку вебзастосунку для продажу домашнього одягу залишається актуальною і важливою, оскільки онлайн-торгівля продовжує набувати популярності, а люди все більше вдаються до покупок через Інтернет. Зокрема, сегмент продажу домашнього одягу може мати великий попит, особливо з урахуванням змін у робочому середовищі та популярності комфортного одягу вдома.

Онлайн-магазини дозволяють підприємцям досягати широкої аудиторії, забезпечуючи користувачам можливість зручно та безпечно робити покупки. Підтримка такого бізнесу вимагає розробки ефективного вебзастосунку, що дозволяє вам ефективно взаємодіяти з клієнтами, демонструвати продукти та обробляти замовлення.

Також, з огляду на те, що технології швидко розвиваються, важливо враховувати останні тренди в розробці веб-додатків, забезпечуючи високий рівень користувацької зручності, безпеки та інноваційних можливостей.

Однак для збереження актуальності вашого проекту важливо вдосконалювати його з часом, додаючи нові функції, враховуючи відгуки користувачів та адаптуючи його до змін у сучасному електронному бізнесі.

Наукова новизна роботи полягає у декількох аспектах, що визначають її важливість та внесок у галузь дослідження електронної комерції та розробки вебзастосунків.

1. Розвиток вебзастосунків для Windows: Багато вебзастосунків в основному розробляються для платформ, таких як веб-браузери або крос-

платформені фреймворки. Дипломна робота фокусується на розробці вебзастосунку для продажу домашнього одягу специфічно для операційної системи "Windows", що може привести до розширення можливостей та оптимізації для цієї конкретної платформи.

2. Інтеграція з операційною системою: Дослідження включає в себе вивчення можливостей та взаємодії з операційною системою "Windows", зокрема використання її унікальних функціональних можливостей та інтеграція з іншими програмами або сервісами, що працюють на даній ОС.

3. Інновації в користувацькому досвіді: Розробка інтерфейсу та функціоналу, спрямованих на покращення користувацького досвіду та зручності покупок онлайн. Інновації включають в себе нові підходи до навігації, взаємодії та представлення інформації для покупців.

4. Комплексний аналіз електронної комерції в галузі моди: Розгляд аспектів електронної комерції, специфічних для сегмента моди, таких як віртуальні примірочні кімнати, персоналізовані рекомендації, знижки та акції, зокрема орієнтовані на домашній одяг.

5. Оцінка впливу та результативності: Дослідження включає в себе ефективний аналіз впливу розробленого вебзастосунку на покупців та результативності електронної комерції у сегменті домашнього одягу.

Загальна наукова новизна полягає в поєднанні технічних аспектів розробки вебзастосунку, інтеграції з платформою "Windows" та інновацій в електронній комерції для покращення якості та зручності онлайн-купівель.

РОЗДІЛ 1
АНАЛІЗ ТА ПОНЯТТЯ ТЕХНОЛОГІЇ

1.1. Технічний стек

Один з перших важливих кроків у розробці вебзастосунку - це вибір технічного стеку.

Фронтенд — це та частина вебзастосунку, яка відповідає за відображення та взаємодію з користувачем. У вебзастосунку для продажу домашнього одягу фронтенд буде відповідати за створення зручного та естетичного інтерфейсу для користувачів.

HTML, CSS і JavaScript - це три основні мови для розробки фронтенду веб-додатків. Кожна з цих мов має свої унікальні функції та використовується для різних аспектів розробки веб-сторінок та додатків.

HTML (HyperText Markup Language)

HTML використовується для створення структури та маркування вмісту веб-сторінок. Це мова розмітки, яка визначає, як повинна виглядати ієрархія елементів на сторінці. Основні поняття HTML включають:

Теги: Елементи, які визначають різні частини сторінки, такі як заголовки, абзаци, зображення, посилання та інше.

Атрибути: Додаткова інформація, яка надається тегам для зміни їх поведінки або вигляду.

```html
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.
    <title>Мій веб-сайт</title>
</head>
<body>
    <h1>Вітаємо на нашому веб-сайті!</h1>
    <p>Це перша сторінка нашого сайту.</p>
    <img src="image.jpg" alt="Зображення">
    <a href="https://www.example.com">Перейти на інший сайт</a>
</body>
</html>
```

Рис. 1.1. Приклад HTML-коду

CSS (Cascading Style Sheets)

CSS використовується для стилізації та оформлення вмісту, визначеного за допомогою HTML. Він забезпечує можливість змінювати кольори, розміри, шрифти та інші аспекти вигляду сторінки. Основні концепції CSS включають:

Селектори: Вказують, які елементи на сторінці повинні бути стилізовані.

Властивості: Визначають конкретні аспекти вигляду, такі як кольори ,розміри, рамки тощо.

Класи та ідентифікатори: Використовуються для точної адресації елементів на сторінці.

```css
body {
    font-family: Arial, sans-serif;
    background-color: #f0f0f0;
}

h1 {
    color: #333;
}

p {
    font-size: 16px;
}

img {
    max-width: 100%;
    height: auto;
}

a {
    color: #0066cc;
    text-decoration: none;
}
```

Рис. 1.2. Приклад CSS-коду

JavaScript

JavaScript використовується для надання динамічності та інтерактивності веб-сторінкам. Він дозволяє вам взаємодіяти з елементами сторінки, обробляти події, виконувати асинхронні запити та додавати розширений функціонал. Основні концепції JavaScript включають:[1]

- Змінні та константи: Для зберігання та використання даних.
- Функції: Блоки коду, які виконують конкретні завдання.
- Об'єкти: Зберігання даних та функцій в одному об'єкті.
- Події: Взаємодія з користувачем та обробка подій.

```javascript
// Отримання елементу за його ідентифікатором
var header = document.getElementById("myHeader");

// Зміна тексту у заголовку
header.innerHTML = "Новий текст заголовку";

// Додавання класу до елементу
header.classList.add("highlight");

// Обробка кліків на посиланні
var link = document.getElementById("myLink");
link.addEventListener("click", function() {
    alert("Посилання було натиснуте!");
});
```

Рис. 1.3. Приклад JavaScript-коду

Разом HTML, CSS і JavaScript створюють потужний інструментарій для розробки фронтенду веб-додатків, що забезпечує високий рівень користувацької зручності та інтерактивності.

Бекенд - це та частина вебзастосунку, яка відповідає за обробку запитів, взаємодію з базою даних, та надання необхідної інформації фронтенду. Для розробки бекенду використовуються різні технології та мови програмування. Основні аспекти бекенду включають:

Мови програмування

Вибір мови програмування залежить від конкретних вимог проекту.

- Node.js (JavaScript/TypeScript): Дозволяє використовувати JavaScript або TypeScript для розробки серверної частини. Зручно для створення асинхронних додатків.
- Python: Добре підходить для розробки швидких і чистих додатків. Фреймворки, такі як Django або Flask, полегшують розробку.
- Java: Мова з великою кількістю фреймворків, таких як Spring чи Apache Struts. Використовується для великих корпоративних проектів.

- Ruby: Розробка на Ruby часто ведеться за допомогою фреймворку Ruby on Rails, що дозволяє швидко розробляти повнофункціональні додатки.

Фреймворки

Фреймворки пропонують готові структури та інструменти для розробки, що полегшує процес та робить код більш структурованим. Наприклад:

- Express (Node.js): Легкий та гнучкий фреймворк для Node.js.
- Django (Python): Включає в себе всі необхідні компоненти для швидкої розробки, включаючи ORM для роботи з базою даних.
- Spring Boot (Java): Забезпечує швидку розробку на Java та вбудований контейнер для додатків.
- Ruby on Rails (Ruby): Скорочує кількість коду і полегшує розробку додатків.

Бази даних

Бекенд зазвичай взаємодіє з базою даних для зберігання та отримання інформації. Популярні типи баз даних:

- Relational Databases (SQL): Такі як MySQL, PostgreSQL, SQLite. Використовують мову SQL для взаємодії.
- NoSQL Databases: Такі як MongoDB, CouchDB. Не використовують традиційні таблиці, а зберігають дані у форматі документів.

Засоби розгортання та хостинг

Для розгортання та хостингу веб-додатків існують різні засоби та платформи. Вибір конкретного інструменту може залежати від ваших потреб, технічних навичок та фінансових можливостей. Ось кілька популярних засобів розгортання та хостингу:

Хмарні платформи:

Amazon Web Services (AWS): Надає широкий спектр послуг для розгортання та масштабування веб-додатків, включаючи віртуальні сервери (EC2), бази даних (RDS), а також інші хмарні рішення.

Microsoft Azure: Платформа від Microsoft, яка також надає широкий спектр хмарних послуг для розгортання та управління додатками.

Google Cloud Platform (GCP): Google пропонує інфраструктуру для хостингу та розгортання веб-додатків, а також інші корисні сервіси.

Хмарні платформи для розробників (PaaS):

Heroku: Простий у використанні сервіс PaaS, який дозволяє розгортати додатки без необхідності вручну налаштовувати інфраструктуру.

Firebase (від Google): Платформа для розробки мобільних та веб-додатків, яка включає в себе хмарні послуги, базу даних, аутентифікацію та інші інструменти.

Хостинг-постачальники:

GitHub Pages: Якщо ваш веб-додаток є статичним, GitHub Pages може служити як безкоштовне та просте рішення для розгортання.

Netlify: Надає хмарний хостинг для статичних та динамічних веб-додатків, а також автоматичні засоби CI/CD.

Vercel: Спеціалізується на хостингу веб-додатків та надає простість у використанні та інтеграцію з різними фреймворками.

Сервери власного розгортання:

Apache: Відкрите програмне забезпечення для обслуговування веб-серверів.

Nginx: Ще один популярний веб-сервер, який також може використовуватися як обертач (reverse proxy).

Docker: Платформа для автоматизації розгортання та управління контейнерами, що полегшує переносимість додатків між середовищами.

Інші інструменти:

DigitalOcean: Платформа хмарного хостингу, яка славиться простотою використання та доступністю.

IBM Cloud: Платформа від IBM, яка надає різні хмарні та когнітивні послуги.

Клієнт-серверна архітектура - це модель взаємодії між комп'ютерами у розподіленій системі, де один комп'ютер (сервер) надає послуги або ресурси, а інший комп'ютер (клієнт) використовує ці послуги чи ресурси.

Розподілена модель - це архітектурний підхід, призначений для реалізації розподіленої обчислювальної системи, де компоненти цієї системи розташовані на різних фізичних або логічних пристроях та співпрацюють через мережу для досягнення спільних цілей. Основні характеристики розподіленої моделі включають:

1.Розташування: Компоненти системи можуть знаходитися на різних фізичних машинах, серверах або навіть в різних мережах.

2.Взаємодія: Компоненти спілкуються між собою через мережу, використовуючи різні механізми передачі даних, такі як HTTP, TCP/IP тощо.

3.Спільні цілі: Компоненти працюють разом для досягнення спільних завдань чи цілей системи.

4. Незалежність: Кожен компонент може працювати незалежно та мати свою власну логіку та стан.

5.Масштабованість: Розподілена модель дозволяє легко масштабувати систему шляхом додавання нових компонентів або ресурсів.

6.Надійність: Завдяки розташуванню компонентів на різних серверах, система може залишатися надійною навіть у випадку відмови окремих частин.

7. Ефективність: Розподілена модель дозволяє оптимізувати використання ресурсів та полегшує паралельну обробку завдань.

8.Безпека: Забезпечення безпеки в розподіленій моделі включає в себе методи шифрування, аутентифікації та контролю доступу.

Розподілені системи можуть бути застосовані в різних областях, включаючи мережеві технології, веб-розробку, обчислювальні хмари, телекомунікації та інші. Вони дозволяють побудовувати більш потужні, гнучкі та масштабовані системи.

Взаємодія через мережу в розподілених системах є критичним аспектом, оскільки компоненти цих систем можуть знаходитися на різних фізичних машинах та спілкуватися між собою через мережеве середовище. Цей процес передбачає обмін даними, комунікацію та встановлення зв'язків між вузлами системи. Основні аспекти взаємодії через мережу включають:

1.Протоколи комунікації: Визначають набір правил та форматів для ефективного обміну даними між компонентами. Прикладами можуть бути HTTP, TCP/IP, SOAP, REST тощо.

2. Архітектурні стилі: Оприділяють загальний підхід до організації взаємодії. Наприклад, стилі RESTful або базовані на повідомленнях (message-passing) для взаємодії між вузлами.

3. Формати даних: Визначають, які дані передаються між компонентами та у якому форматі. Часто використовуються JSON або XML для представлення структурованих даних.

4.Алгоритми маршрутизації: Для ефективного направлення даних між різними вузлами системи, особливо у великих мережах.

5.Забезпечення надійності: Механізми для виявлення та виправлення помилок під час передачі даних.

6.Аутентифікація та авторизація: Забезпечують безпеку взаємодії, визначаючи, хто має доступ до яких ресурсів.

7.Методи взаємодії: Включають синхронний та асинхронний обмін даними, відправлення повідомлень, виклик методів та інші методи взаємодії.

8. Механізми обробки помилок: Які способи передбачені для вирішення ситуацій, коли взаємодія може бути порушена.

Взаємодія через мережу є важливою частиною розподіленої архітектури і вимагає уважного планування та реалізації для забезпечення ефективної та надійної комунікації між компонентами системи.

Розділення відповідальностей (Separation of Concerns) є принципом проектування, який визначає, що програмні системи повинні бути розділені на окремі частини або компоненти, кожен з яких відповідає за конкретну аспекту функціональності. Цей принцип сприяє полегшенню розробки, тестування та обслуговування програм, а також покращенню їх читабельності та розуміння.

Основні переваги розділення відповідальностей включають:

1. Модульність: Кожен компонент системи відповідає за конкретну задачу чи функціональність, що робить систему більш розділеною та менш залежною.

2. Підтримка змін: Зміни в одній частині системи не повинні суттєво впливати на інші, що полегшує розширення та модифікацію.

3. Відновлення помилок: Виправлення помилок та оптимізація коду можуть бути зосереджені в окремих компонентах, що полегшує обслуговування.

4. Покращена читабельність: Розділення відповідальностей сприяє створенню коду, який легко читати та розуміти, оскільки кожен компонент відповідає за конкретний аспект системи.

5. Повторне використання: Можливість використовувати окремі компоненти в інших проектах або контекстах.

6. Тестування: Модульне тестування стає більш ефективним, оскільки можна тестувати окремі компоненти незалежно один від одного.

Наприклад, в розподіленій веб-системі можна розділити відповідальності між фронтендом (інтерфейс та взаємодія з користувачем) та бекендом (логіка серверної частини, обробка даних). Це дозволяє командам спеціалізуватись на своїх областях та ефективно працювати над різними аспектами системи.

1.2. Інтеграція з операційною системою "Windows"

Інтеграція з операційною системою Windows може включати в себе різноманітні аспекти від налаштування сервера до взаємодії з різними компонентами системи.

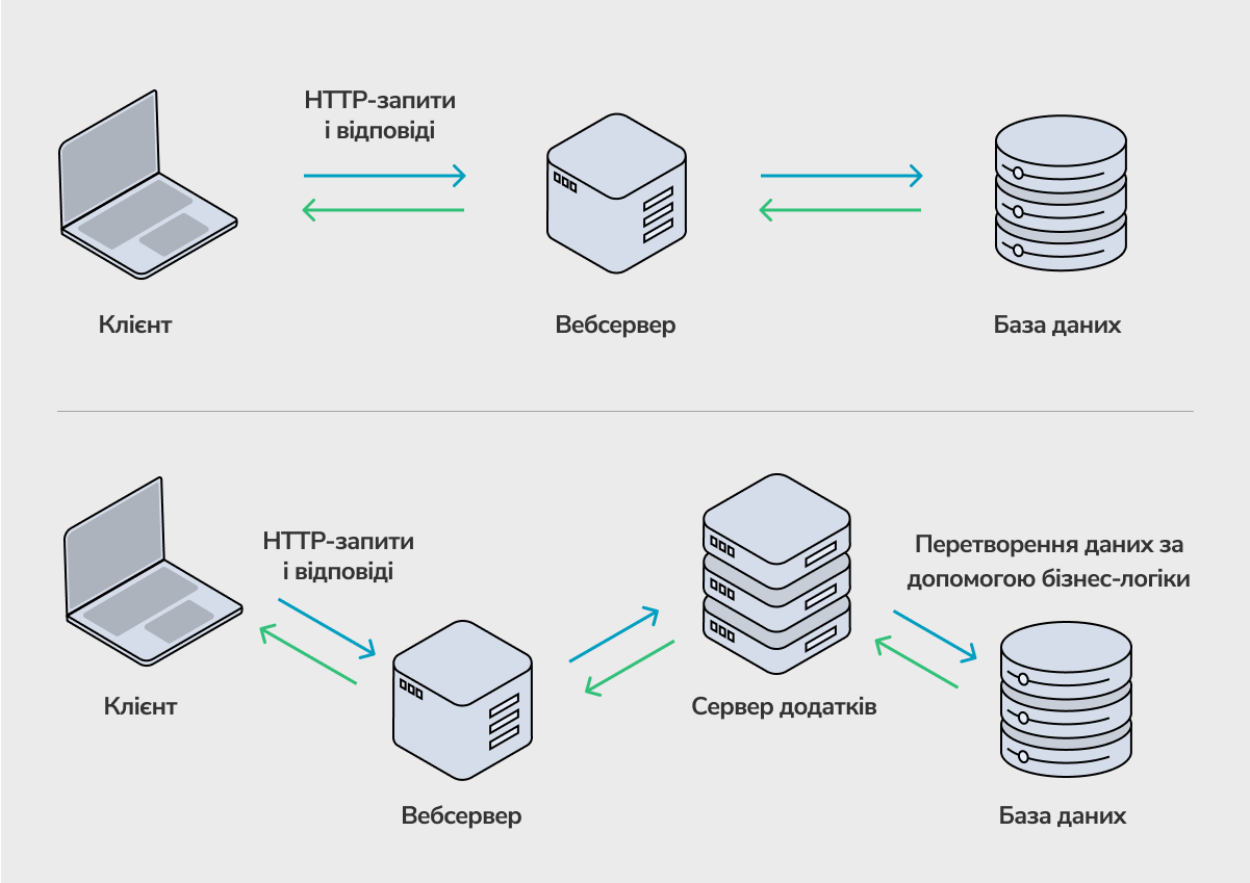

Рис. 1.4. Приклад роботи веб сервера

Вибір **веб-сервера** є ключовим етапом у розробці та розгортанні вебзастосунку. Є кілька популярних веб-серверів, які підтримують операційну систему Windows. Нижче представлено деякі з них:[5-6]

Internet Information Services (IIS):

IIS є вбудованим веб-сервером для операційних систем Windows.

Переваги:

Висока продуктивність, оптимізований для Windows.

Легко інтегрується з іншими продуктами Microsoft, такими як .NET та SQL Server.

Підтримує різноманітні технології, включаючи ASP.NET, PHP, Node.js і багато інших.

Недоліки:

Більш призначений для технологій Microsoft, інтеграція з іншими технологіями може виявитися менш зручною.[4]

Apache HTTP Server:

Apache - це один з найпоширеніших веб-серверів у світі, і він підтримує операційну систему Windows.

Переваги:

Відкритий код і безкоштовний.

Широкий вибір модулів та плагінів для розширення функціоналу.

Легко налаштовується та конфігурується.

Недоліки:

Може бути менш ефективним на Windows порівняно з Linux.

Nginx:

Nginx - це легкий та ефективний веб-сервер.

Переваги:

Висока продуктивність та ефективність.

Відмінно підходить для обробки багато з'єднань одночасно (висока конкурентність).

Легко налаштовується та відомий своєю стабільністю.

Недоліки:

Більш призначений для статичного контенту, може вимагати додаткових налаштувань для обробки динамічного контенту.

Microsoft HTTPAPI/2.0:

HTTPAPI - це компонент операційної системи Windows, який може бути використаний для обробки HTTP-запитів.

Переваги:

Вбудований компонент операційної системи, що зменшує необхідність у встановленні окремого веб-сервера.

Підтримує HTTP/2 для оптимізації швидкості завантаження веб-сторінок.

Недоліки:

Менше розширені можливості порівняно з іншими веб-серверами.

Вибір веб-сервера залежить від конкретних вимог проекту, рівня експертизи команди розробників та інших факторів. Зазвичай, IIS є популярним вибором для додатків, що використовують технології Microsoft, тоді як Apache та Nginx використовуються для більш різноманітних стеків технологій.

Системні ресурси

При розробці та розгортанні веб-додатків для операційної системи Windows, ефективне використання системних ресурсів є критично важливим для забезпечення оптимальної продуктивності та роботи вашого додатку. Ось деякі ключові аспекти, які варто врахувати:

Пам'ять (RAM):

Оптимізація використання пам'яті: Використання ефективних структур даних та оптимізація запитів може допомогти зменшити використання оперативної пам'яті.

Кешування: Використання кешування для збереження проміжних результатів та часто використовуваних даних може поліпшити продуктивність та зменшити навантаження на пам'ять.

Процесор (CPU):

Асинхронні операції: Використання асинхронних операцій дозволяє оптимально використовувати процесор та підтримувати велику кількість одночасних з'єднань.

Оптимізація коду: Ефективний та оптимізований код може зменшити навантаження на процесор та підвищити загальну продуктивність.

Диск (HDD/SSD):

Кешування даних: Використання кешування для зменшення читання/запису даних на диск та прискорення доступу до інформації.

Оптимізація запитів до бази даних: Зменшення кількості та оптимізація запитів може поліпшити швидкість доступу до бази даних.

Мережа:

Компресія даних: Використовуйте методи компресії для зменшення обсягу передачі даних між сервером та клієнтом.

Мінімізація HTTP-запитів: Об'єднання ресурсів та мінімізація HTTP-запитів можуть скоротити час завантаження сторінок.

Система Логування:

Ефективне логування: Оптимізуйте логування, зберігаючи лише необхідну інформацію та використовуючи асинхронні методи для уникнення блокування процесів.

Сервіси Windows:

Автоматичний запуск: Налаштуйте служби Windows так, щоб вони автоматично запускались під час завантаження системи.

Моніторинг ресурсів: Використовуйте засоби моніторингу для вивчення використання ресурсів та вчасного виявлення можливих проблем.

Адаптивність до Навантаження:

Масштабованість: Забезпечте масштабованість вашого додатку, щоб він міг ефективно обробляти зростаюче навантаження.

Ефективне управління системними ресурсами дозволить покращити продуктивність та стійкість вебзастосунку під час роботи під управлінням операційної системи Windows.

Аутентифікація та авторизація

Аутентифікація та авторизація є критично важливими аспектами для забезпечення безпеки та захисту конфіденційності даних у вебзастосунку. Особливо на операційній системі Windows, використання інтегрованих механізмів аутентифікації може полегшити цей процес. Ось кілька ключових аспектів:

Аутентифікація: Windows Authentication:

− використання інтегрованої аутентифікації Windows дозволяє використовувати облікові записи користувачів операційної системи для входу в систему;

− забезпечення безпеки за допомогою протоколів, таких як NTLM або Kerberos;

− налаштування IIS для використання Windows Authentication та додаткова конфігурація веб-додатка для обробки інформації про аутентифікованих користувачів.

Форма аутентифікації:

Використання форми аутентифікації для введення інформації про користувача (ім'я користувача та пароль).

Захист від атак, таких як атаки на перебор паролів, за допомогою заходів безпеки, таких як блокування облікових записів після кількох невдалих спроб.

Авторизація:

Ролі та Дозволи:

Визначення ролей користувачів, таких як "Адміністратор", "Користувач", "Гость" і встановлення відповідних дозволів для кожної ролі.

Використання системних ролей Windows або створення власних ролей для кращого контролю.

Claims-Based Авторизація:

Використання системи клеймів (claims) для передачі додаткової інформації про користувача та його права.

Встановлення зв'язків між клеймами та дозволами для управління доступом.[3]

Додаткові Аспекти:

Дводіапазонна Аутентифікація (2FA):

Використання двофакторної аутентифікації для додаткового рівня безпеки.

Включення методів, таких як OTP (одноразові паролі) або апаратні токени.

Логування та Аудит:

Запис подій аутентифікації та авторизації для подальшого аудиту та виявлення можливих загроз безпеки.

SSL/TLS Захист:

Використання протоколу HTTPS для захисту передачі конфіденційних даних між користувачем та сервером.

Сесії та Контроль Стану:

Захист сесій користувача та використання заходів безпеки, таких як токени аутентифікації для контролю стану сесій.

Рис. 1.5. Процесс авторизації

Ефективна аутентифікація та авторизація включають в себе ретельне планування, налаштування опцій безпеки операційної системи Windows та вибір відповідних механізмів для забезпечення надійності та безпеки вашого вебзастосунку.

Аналіз ефективності та безпеки

Аналіз ефективності та безпеки є ключовим етапом у розробці веб-додатків. Оцінка продуктивності та заходів безпеки дозволяє забезпечити надійну та ефективну роботу додатку. Ось кілька аспектів, які варто врахувати:

Ефективність:

Продуктивність Веб-Сервера:

Вимірювання часу відгуку сервера та швидкість обробки запитів.

Моніторинг використання ресурсів сервера, таких як CPU, пам'ять та диск.

Оптимізація Запитів та Бази Даних:

Використання кешування для часто запитуваних даних.

Оптимізація SQL-запитів та використання індексів для покращення продуктивності бази даних.

Масштабованість:

Тестування можливості додатку масштабуватися та ефективно обробляти збільшене навантаження.

Виявлення та Оптимізація Узких Місць:

Аналіз журналів та моніторинг для виявлення узких місць у роботі додатку.

Оптимізація алгоритмів та компонентів, що використовують багато ресурсів.

Безпека:

Тестування На Проникнення:

Проведення тестів на проникнення для виявлення можливих вразливостей.

Аудит заходів безпеки та виправлення виявлених проблем.

SSL/TLS Захист:

Використання протоколу HTTPS для захисту конфіденційних даних у транзиті.

Налаштування коректних параметрів шифрування та сертифікатів.

Контроль Доступу:

Перевірка та оновлення прав доступу користувачів.

Використання принципу найменших привілеїв та обмеження прав на необхідні мінімум.

Захист від Атак:

Використання механізмів захисту від атак, таких як захист від введення з зовнішніх джерел (XSS, SQL Injection).

Валідація та санітарізація введених даних.

Моніторинг Логів та Аудит:

Моніторинг логів для виявлення непередбачуваних подій та потенційних загроз.

Аудит та журналювання дій користувачів та системних подій.

Двофакторна Аутентифікація (2FA):

Впровадження двофакторної аутентифікації для збільшення рівня безпеки.

Безпека Сесій:

Використання безпечних механізмів сесій та їх належна конфігурація.

Обробка Помилок та Винятків:

Забезпечення безпечного оброблення помилок, щоб уникнути витоку конфіденційної інформації.

Ефективний аналіз ефективності та безпеки вимагає поєднання автоматизованих і ручних тестів, аудиту та моніторингу. Ретельна увага до цих аспектів дозволить створити надійний та ефективний веб-додаток.

Робота з базами даних є ключовим аспектом при розробці веб-додатків. Ось деякі важливі аспекти стосовно баз даних, які слід врахувати:

База даних

Вибір Бази Даних:

Relational Database Management System (RDBMS): Такі як MySQL, PostgreSQL, Microsoft SQL Server, Oracle.

NoSQL Databases: Такі як MongoDB, Cassandra, Redis.

Модель Даних:

Реляційна Модель: Використовує таблиці для представлення даних та визначає взаємозв'язки між ними.

NoSQL Модель: Документна, ключ-значення, стовпчаста, графова - вибір залежить від потреб проекту.

Нормалізація Даних:

Застосування нормалізації для уникнення аномалій даних та забезпечення ефективного використання простору.

Індексація:

Використання індексів для прискорення операцій пошуку та сортування даних.

Транзакції: Забезпечення атомарності, консистентності, ізоляції та стійкості (ACID) для ефективної роботи з транзакціями.

Захист Даних:

Використання різних методів захисту, таких як шифрування, для захисту конфіденційності даних.

Резервне Копіювання та Відновлення:

Регулярне резервне копіювання бази даних та тестування процедур відновлення.

Інтеграція з Додатком:

Забезпечення ефективної інтеграції бази даних із веб-додатком.

Використання ORM (Object-Relational Mapping) для спрощення взаємодії з базою даних.

Оптимізація Запитів:

Аналіз та оптимізація складних запитів для забезпечення високої продуктивності.

Масштабування Баз Даних:

Використання горизонтального (шарового) або вертикального масштабування для підтримки збільшеного обсягу даних.

Міграції та Оновлення:

Ефективне керування міграціями та оновленнями схеми бази даних.

Моніторинг та Аналітика:

Налаштовані механізми моніторингу для виявлення аномалій та оптимізації продуктивності.

Безпека Бази Даних:

Встановлення правильних прав доступу до бази даних та аудит доступу.

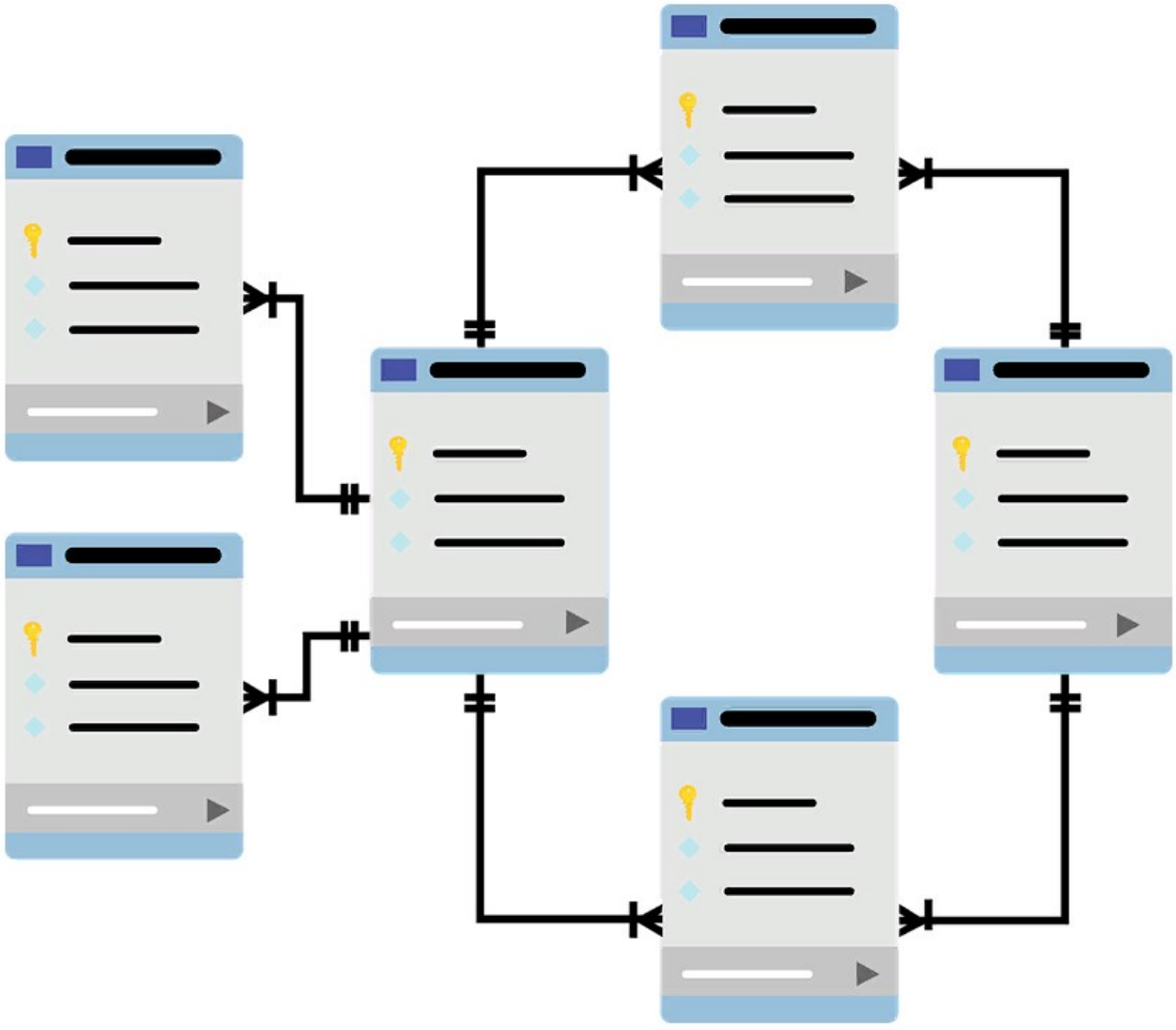

Рис. 1.6. Бази даних

Ефективна робота з базами даних включає в себе розуміння основних концепцій, вибір відповідних технологій та постійне вдосконалення для забезпечення ефективної та безпечної роботи вашого вебзастосунку.[1,2; 5-8]

Служби Windows

Служби Windows є важливою частиною інфраструктури операційної системи, яка надає можливість виконання фонових задач та служб на комп'ютері, навіть коли користувач не увійшов у систему.

Їх використання у веб-додатках може бути корисним для автоматизації операцій, які вимагають постійного моніторингу чи обслуговування. Ось кілька ключових аспектів:

Створення Служб:

Служби можна створювати, використовуючи мови програмування, які підтримують розробку під Windows, такі як C# або Python.

Визначення точки входу служби та налаштування параметрів її функціонування.

Фонові Завдання:

Використання служб для виконання фонових завдань, таких як генерація звітів, обробка даних, регулярне виконання операцій тощо.

Робота із Системними Ресурсами:

Управління ресурсами, які використовує служба, такими як CPU, пам'ять, диск, для оптимальної роботи системи.

Логування та Моніторинг:

Запис інформації в журнали для моніторингу роботи служби та виявлення проблем.

Визначення механізмів сповіщення в разі виникнення помилок чи невдалого виконання.

Автоматичний Запуск:

Налаштування служб для автоматичного запуску при старті операційної системи.

Забезпечення Високої Доступності:

Реалізація механізмів перезапуску служби в разі її аварійного завершення.

Інтеграція із Системними Завданнями:

Взаємодія з планувальником завдань Windows для регулярного виконання операцій.

Безпека та Аутентифікація:

Налаштування прав доступу до служби та використання безпечних методів аутентифікації.

Створення та налагодження служб Windows вимагає уважності до деталей та врахування особливостей використання веб-додатком. Це може бути корисно для оптимізації фонових операцій та забезпечення стабільної роботи системи.

Інтеграція з API Windows

Інтеграція з API Windows відкриває широкі можливості для взаємодії вашого вебзастосунку з операційною системою. Це дозволяє здійснювати різноманітні функції, такі як робота з файловою системою, реєстром, мережевими сервісами та іншими системними ресурсами. Нижче наведено ключові аспекти інтеграції з API Windows:

Робота з Файловою Системою:

Використання API для читання, запису, перейменування та видалення файлів і папок.

Моніторинг змін у вказаних директоріях.

Взаємодія з Реєстром:

Зчитування та запис даних в системний реєстр Windows.

Використання реєстрових подій для відстеження змін.

Мережеві Операції:

Встановлення та розрив з'єднань, робота з мережевими ресурсами.

Взаємодія з мережевими принтерами, сертифікатами тощо.

Робота з Службами та Процесами:

Керування службами Windows: запуск, зупинка, конфігурація.

Моніторинг та керування процесами в операційній системі.

Взаємодія із Зовнішніми Програмами:

Виклик зовнішніх програм із вебзастосунку через API Windows.

Передача параметрів та обробка результатів виконання.

Робота з Подіями та Повідомленнями:

Використання API для обробки системних подій та повідомлень.

Спостереження за змінами у системі та реагування на них.

Використання Бібліотек та Фреймворків:

Використання спеціалізованих бібліотек та фреймворків, які надають API для конкретних операцій (наприклад, Windows API Code Pack).

Безпека та Авторизація:

Забезпечення безпеки під час використання системних API.

Обмеження прав доступу та використання безпечних практик.

Інтеграція з API Windows розширює можливості вебзастосунку та дозволяє вам більш ефективно використовувати можливості операційної системи для реалізації різноманітних функцій.

Інсталяція та Оновлення вебзастосунку на Операційній Системі Windows

Інсталяція та оновлення веб-додатків на операційній системі Windows вимагає ретельного підходу та використання відповідних інструментів. Нижче наведено ключові аспекти цього процесу:

Створення Інсталятора:

Використання спеціальних інструментів для створення інсталятора (наприклад, Inno Setup, NSIS, WiX Toolset).

Конфігурування інсталятора для розміщення вебзастосунку, налаштування прав доступу та інших параметрів.

Підготовка Сервера:

Забезпечення наявності необхідного середовища виконання (наприклад, встановлення веб-сервера, бази даних, мов програмування).

Налаштування прав доступу до каталогів та ресурсів, які використовує веб-додаток.

Контроль Залежностей:

Вказання та вивчення всіх залежностей вебзастосунку (бібліотеки, фреймворки, мови програмування).

Забезпечення, що всі необхідні компоненти встановлені на сервері.

Автоматизація Процесу:

Використання сценаріїв або інструментів для автоматизації процесу інсталяції та оновлення.

Можливість розгортання за допомогою командного рядка або інших інструментів автоматизації (наприклад, PowerShell).

Запобігання Конфліктам:

Перевірка конфліктів із іншими програмами або службами, які можуть впливати на роботу вебзастосунку.

Вивчення можливих проблем із сумісністю на операційній системі.

Механізми Оновлення:

Розробка механізмів автоматичного оновлення вебзастосунку.

Забезпечення можливості безпечного та надійного вдосконалення на нові версії.

Інтеграція з Планувальником Завдань:

Можливість автоматичного планування процесів інсталяції та оновлення за допомогою планувальника завдань Windows.

Логування та Моніторинг:

Додавання логування подій в процес інсталяції та оновлення для подальшого моніторингу та аналізу.

Ефективна інсталяція та оновлення вебзастосунку дозволяє забезпечити користувачам швидку та безперебійну роботу програмного продукту. Такий процес також допомагає уникнути проблем сумісності та забезпечує безпеку вебзастосунку під час встановлення та оновлення.

ВИСНОВКИ ДО РОЗДІЛУ 1

Аналіз та визначення технологічних аспектів вебзастосунку для продажу домашнього одягу на операційній системі "Windows" виявили важливі фактори та визначили ключові напрямки розробки. Ретельний аналіз вибору технічного стеку вказує на використання HTML, CSS, та JavaScript для реалізації фронтенду, а також вибір мови програмування та бази даних для бекенду. Це забезпечить ефективність та розширюваність системи.Підкреслено важливість адаптивності та привабливого користувацького інтерфейсу для забезпечення зручного та приємного досвіду покупців. Використання сучасних методик дизайну є ключовим аспектом.Аналіз ефективності та безпеки вказує на необхідність ретельної роботи з системними ресурсами, впровадження ефективних методів аутентифікації та авторизації, а також застосування найсучасніших практик безпеки.

В цілому, аналіз технології вказав на важливість комплексного та систематичного підходу до розробки вебзастосунку, що базується на операційній системі "Windows". Врахування всіх аспектів, починаючи від

технічного стеку та закінчуючи ефективністю та безпекою, грає ключову роль у створенні успішного продукту для кінцевого користувача.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ВЕБЗАСТОСУНКУ

2.1. Опис вимог до вебзастосунку

Функціональні Вимоги:

- Реєстрація та Авторизація:

- Можливість реєстрації нового користувача.

- Система авторизації для зареєстрованих користувачів.

- Головна Сторінка:

- Відображення актуальних оновлень товарів та знижок.

- Кнопки навігації для швидкого доступу до інших розділів.

- Каталог Товарів:

- Сортування та фільтрація товарів за різними критеріями.

- Можливість додавання товару в кошик.

- Кошик:

- Відображення вмісту кошика з вказанням кількості та ціни товарів.

- Можливість зміни кількості товару та видалення товарів з кошика.

- Оформлення Замовлення:

- Форма для введення адреси та інформації про доставку.

- Виведення підсумкової інформації перед підтвердженням замовлення.

- Сторінка Відгуків:

- Можливість користувачів залишати відгуки та оцінки до придбаних товарів.

- Відображення загальної рейтингу та коментарів.

Дизайн та Інтерфейс:

-Користувацький Інтерфейс:

- Мінімалістичний та привабливий дизайн.

- Зручна навігація та легкий доступ до основних функцій.

- Адаптивність:

- Адаптивний дизайн для оптимального відображення на різних пристроях (планшети, мобільні телефони, настільні комп'ютери).

Безпека та Аутентифікація:

- Захист Даних:

- Використання шифрування для передачі конфіденційної інформації.

- Захист від SQL-ін'єкцій та інших атак на безпеку.

- Аутентифікація та Авторизація:

- Безпечний процес входу та збереження даних профілю користувача.

- Авторизація та доступ до особистого кабінету.

Адміністративний Функціонал:

- Управління Товарами:

- Додавання, редагування та видалення товарів з каталогу.

- Налаштування акцій та знижок.

- Моніторинг Замовлень:

- Перегляд та обробка замовлень з адміністративного панелі.

- Сповіщення адміністратора про нові замовлення.

- Керування Відгуками:

- Можливість видалення неприпустимих відгуків.

- Модерація коментарів від користувачів.

Технічні Вимоги:

- Сумісність з Windows:

- Оптимізація для операційної системи Windows.

- Використання API Windows для покращення функціональності.

- Веб-Сервер та Хостинг:

- Сумісність з вибраним веб-сервером та інструментами для розгортання.

- Відповідність вимогам хостингу.

Цей опис вимог визначає основні функції, які має виконувати вебзастосунок для продажу домашнього одягу на операційній системі Windows, забезпечуючи зручний та безпечний процес для користувачів і адміністраторів.

2.2. Вибір та використання середовища розробки

Visual Studio Code:

Переваги:

1. Безкоштовний та відкритий код: VSCode є безкоштовним і розповсюджується з відкритим вихідним кодом.

2. Широкі можливості розширення: Має велику кількість розширень, що дозволяє розширювати його функціональність для роботи з різними мовами та технологіями.

3. Інтеграція з Git: Вбудована підтримка системи контролю версій Git дозволяє легко взаємодіяти з репозиторіями.

4. Легка та швидка: Займає мало ресурсів системи та швидко запускається.

Недоліки:

1. Може бути менш продуктивним для великих проектів: Для дуже великих проектів з великою кількістю файлів може виникнути менша продуктивність.

2. Не такий вбудований функціонал, як у повноцінних IDE: У порівнянні з деякими повноцінними IDE може відсутній деякий функціонал, але це компенсується розширеннями.

Visual Studio Code є популярним та універсальним редактором коду, який підходить як для початківців, так і для досвідчених розробників, завдяки своїй простоті та розширюваності.

Sublime Text:

Переваги:

1. Швидкість та легкість: Sublime Text дуже швидкий та легкий, завдяки чому відмінно підходить для роботи навіть на менш потужних пристроях.

2. Багатофункціональність: Має ряд корисних функцій, таких як швидкий пошук та заміна, вибірку курсорів, розширені можливості мов та інші.

3. Широкі можливості налаштувань: Високий рівень налаштувань та можливість встановлення плагінів для додаткового функціоналу.

4. Багатомовність: Підтримка багатьох мов програмування.

Недоліки:

1.Не безкоштовний: Sublime Text платний, хоча його можна використовувати в режимі "демо" без обмежень.

2. Менше вбудованих функцій для роботи з Git: У порівнянні з іншими редакторами чи IDE, підтримка Git менш розширена.

3. Обмежена функціональність без розширень: Деякі корисні функції можуть вимагати встановлення додаткових плагінів.

Sublime Text є швидким та надзвичайно гнучким редактором коду, ідеальним для тих, хто цінує продуктивність та можливість налаштувань.

Eclipse:

Переваги:

1. Відкритий код та безкоштовність: Eclipse є відкритим програмним забезпеченням та повністю безкоштовним для використання.

2. Широкі можливості розширення: Має велику кількість плагінів та розширень для роботи з різними мовами програмування та технологіями.

3. Багатомовність: Підтримка багатьох мов програмування та інструментів розробки.

4. Середовище розробки для різних типів проектів: Ідеально підходить для розробки різних типів програм, від мобільних додатків до великих корпоративних систем.

Недоліки:

1. Важкий та ресурсомісткий: Eclipse може бути важким для ресурсозбереження, що може впливати на продуктивність на менших пристроях.

2. Менше інтеграції з Git порівняно з іншими інструментами: У порівнянні з сучасними IDE, інтеграція з системами контролю версій, зокрема Git, може бути менш розширеною.

3. Специфічний інтерфейс користувача: Інтерфейс Eclipse може виглядати застарілим або складним для новачків порівняно з іншими редакторами.

Eclipse є потужним інтегрованим середовищем розробки, яке використовується для розробки різноманітних програм. Його відкритий код та розширюваність роблять його популярним серед розробників.

Atom:

Переваги:

1. Відкритий код та безкоштовність: Atom є відкритим програмним забезпеченням та повністю безкоштовним для використання.

2. Легкість та швидкість: Atom має легкий та ефективний інтерфейс, що робить його швидким та легким у використанні.

3. Спільнота та розширення: Має активну спільноту та широкий вибір розширень, які можна встановити для розширення функціоналу.

4. Інтеграція з GitHub: Атом має вбудовану підтримку для GitHub, що полегшує роботу з репозиторіями.

Недоліки:

1. Витрати пам'яті: Atom може використовувати багато пам'яті, особливо при роботі з великими проектами, що може впливати на продуктивність.

2. Іноді повільний для великих проектів: Завдяки використанню веб-технологій для реалізації інтерфейсу, Atom може бути повільним для великих проектів порівняно з іншими редакторами.

3. Менше можливостей для роботи з мовами програмування: У порівнянні з деякими іншими IDE, Atom може мати менше можливостей для роботи з певними мовами програмування або технологіями.

Atom є легким та приємним у використанні редактором, спрямованим на розробників. Відкритий код і широка підтримка розширень роблять його популярним серед користувачів, які шукають простий та розширюваний текстовий редактор.

Вибір Visual Studio Code (VS Code) є відмінним варіантом для розробки вебзастосунку для продажу домашнього одягу на операційній системі Windows. Visual Studio Code — це легкий, швидкий та розширюваний текстовий редактор, розроблений компанією Microsoft.

Рис. 2.1. Іконка застосунку VScode

Переваги використання VS Code:

Багатофункціональність:

- В VS Code ви можете працювати з різними мовами програмування та технологіями, включаючи HTML, CSS, JavaScript, а також мови для бекенду, такі як Node.js, Python, та інші.

Легкість та Простота Використання:

- VS Code відомий своєю легкістю та швидкістю роботи. Він має інтуїтивний інтерфейс, що полегшує роботу з ним, навіть для новачків.

Розширюваність:

- Широкий вибір розширень, які ви можете легко встановлювати з магазину розширень, робить VS Code вельми розширюваним та можливо найкращим інструментом для вашого проекту.

Інтеграція з Git:

- VS Code має вбудовану інтеграцію з Git, що дозволяє легко керувати версіями вашого коду, використовуючи потужні інструменти для роботи з гілками та злиттями.

Локальний Розробник:

- Завдяки своєму характеру VS Code чудово підходить для локальної розробки та праці в середовищах без доступу до великих ресурсів.

Робота з Розширеннями Frontend:

- Велика кількість розширень для роботи з frontend-технологіями, такими як автодоповнення HTML/CSS/JavaScript, підтримка фреймворків, та інше.

Комунікація та Спільна Робота:

- Інтеграція з різними сервісами командної роботи, такими як Slack, забезпечує зручну комунікацію та спільну роботу в команді.

Співпраця із Visual Studio:

- VS Code може використовуватися в парі з Visual Studio для спільної роботи над проектами, що полегшує співпрацю між фронтенд та бекенд розробниками.

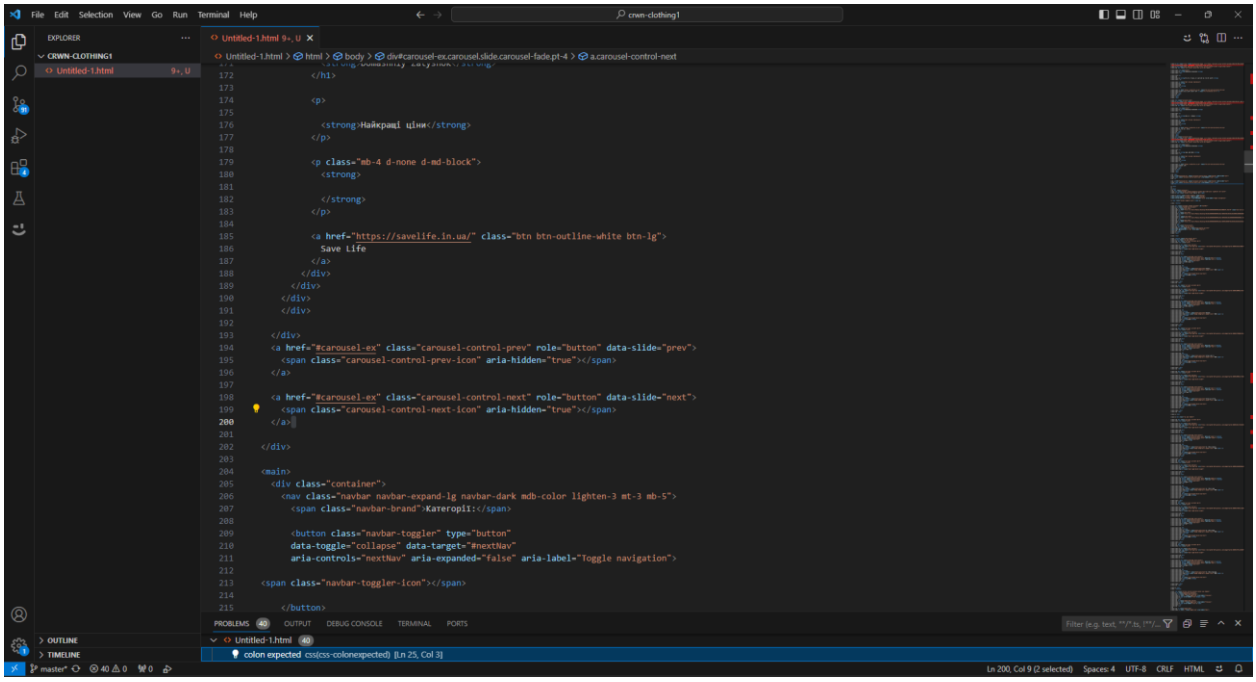

Рис. 2.2. Вигляд VScode на прикладі коду мого вебзастосунку

Загалом, Visual Studio Code — це потужний та зручний інструмент для розробки веб-додатків, який може забезпечити необхідні можливості для створення ефективного та високоякісного продукту.

2.3. Головні аспекти функціоналу вебзастосунку

Функціонал **"Перегляд та Пошук Товарів"** є ключовим для будь-якого інтернет-магазину або вебзастосунку для продажу товарів. Ось кілька важливих причин, для яких це функціонал є важливим:

Зручність користувача:

- Користувачам легко знаходити потрібний товар за допомогою полів для пошуку та категорій.

- Відображення товарів у вигляді списку або сітки полегшує перегляд та вибір товарів.

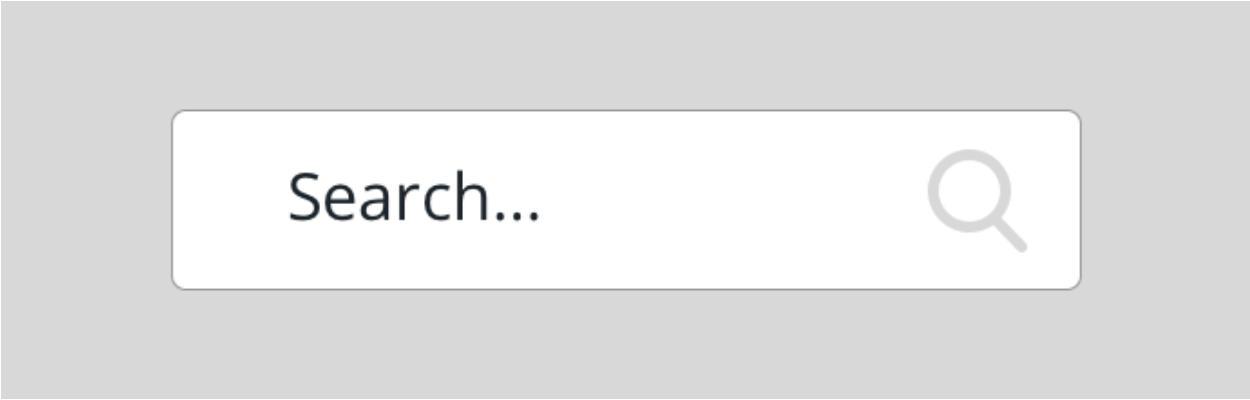

Рис. 2.3. Приклад вигляду поля для пошуку

Швидкий доступ до інформації:

- Користувачам не потрібно прокручувати велику кількість сторінок, щоб знайти потрібний товар.

- Зручний пошук дозволяє швидко фільтрувати товари за ключовими словами чи параметрами.

Підтримка маркетингових стратегій:

- Відображення акцій, розпродаж та нових товарів на головній сторінці або в секції "Актуальні товари" привертає увагу покупців.

- Пошук дозволяє знайти товари, що відповідають конкретним потребам або зацікавленням.

Покращення продажів:

- Зручний та ефективний пошук допомагає залучати користувачів та підвищує ймовірність їхнього конвертування в покупці.

- Привабливий та легкий у використанні інтерфейс збільшує імовірність успішних транзакцій.

Аналіз та статистика:

- Збір інформації про пошукові запити та перегляди товарів дозволяє аналізувати популярність товарів та уподобання покупців.

- Ці дані можуть використовуватися для удосконалення асортименту та розробки маркетингових стратегій.

Підтримка SEO: - Зручний та ефективний пошук сприяє оптимізації для пошукових систем, підвищуючи видимість вебзастосунку та товарів у пошукових результатах.

Керування асортиментом:

- Функціонал дозволяє легко додавати, видаляти та оновлювати товари в асортименті, забезпечуючи актуальність та варіативність.

Усі ці аспекти роблять функціонал "Перегляд та Пошук Товарів" необхідним для створення успішного та конкурентоспроможного вебзастосунку для продажу товарів.

Функціонал "**Детальна Інформація про Товар**" є важливим елементом для забезпечення повної та зрозумілої інформації користувачам про конкретний товар. Ось кілька причин, чому це важливо:

Рішення про покупку:

- Детальна інформація про товар надає користувачам достатньо даних для прийняття обізнаного рішення про покупку.

- Опис, характеристики, розміри, матеріали та інші важливі дані допомагають уникнути непорозумінь та невпевненості.

Створення довіри:

- Забезпечення інформації про якість та особливості товару допомагає встановити довіру між покупцем і продавцем.

- Фотографії товару з різних кутів та високоякісні зображення допомагають користувачам краще розглядати товар перед покупкою.

Уникнення повернень та непорозумінь:

- Чим більше інформації про товар користувач отримає перед покупкою, тим менше ймовірність, що товар не відповідає очікуванням, і його доведеться повертати.

- Якісний опис та фотографії роблять процес покупки більш прозорим.

Український розмір	46	48	50	52	54	56	58 - 62	64 - 66	68
Міжнародний розмір	XS	S	M	L	XL	2XL	3XL	4XL	5XL
Обхват грудей	92	96	100	104	108	112	116 - 124	128 - 132	136
Обхват талії	76	82	88	94	100	106	112 - 120	124 - 128	132
Обхват стегон	96	100	104	108	112	116	120 - 128	132 -134	136

Рис. 2.4. Приклад інформаційної сітки

Підвищення конверсії:

- Користувачі, які мають достатньо інформації, більш схильні здійснити покупку.

- Детальна інформація може бути рішучим чинником для того, щоб користувач зробив покупку, особливо коли у нього є всі необхідні дані.

Підтримка SEO:

- Розміщення детальної інформації на сторінці товару дозволяє покращити його оптимізацію для пошукових систем.

- Якщо користувачі шукають конкретні характеристики чи опис, це сприяє збільшенню шансів того, що ваш товар буде відображений у результатах пошуку.

Узагалі, детальна інформація про товар є ключовою для успішного інтернет-магазину, оскільки вона впливає на рішення покупців та забезпечує їхнє задоволення від покупки.

Функціонал "**Додавання Товарів у Кошик**" в інтернет-магазині або вебзастосунку для продажу товарів є важливим елементом для зручності та ефективності процесу покупки.

Зручність користувача: - Дозволяє користувачам додавати товари у віртуальний кошик за кілька клацань, спростовуючи процес вибору та придбання товарів.

- Кошик служить місцем для зберігання вибраних товарів перед оформленням замовлення.

Оцінка вартості замовлення:

- Користувачі можуть швидко переглянути вміст кошика та оцінити загальну вартість замовлення, включаючи вартість товарів і можливі додаткові витрати.

Формування замовлення:

- Зберігання товарів у кошику дозволяє користувачам переглядати та редагувати свій вибір перед оформленням замовлення.

- Процес додавання товарів до кошика призначений для формування замовлення та зручного його завершення.

Можливість зберігання на деякий час:

- Інформація про товари у кошику може бути збережена, навіть якщо користувач закриє вкладку чи перезавантажить сторінку.

- Це дозволяє покупцям повернутися до свого кошика в будь-який момент та продовжити процес покупки.

Управління запасами:

- Додавання товарів до кошика може використовуватися для резервування товарів на певний час, щоб інші покупці не могли купити їх протягом цього періоду.

Можливість застосування знижок та промокодів:

- Деякі знижки або промокоди можуть бути застосовані тільки під час оформлення замовлення у кошику, що стимулює покупців завершити покупку.

Використання аналітики:

- Інформація про товари у кошику може бути використана для аналізу покупкового поведінки та покращення стратегій продажу.

Загалом, функціонал "Додавання Товарів у Кошик" покликаний зробити процес покупки простішим, зручнішим та забезпечити задоволення від онлайн-шопінгу.

Функціонал **"Реєстрація" та "Авторизація"** є важливими для забезпечення безпеки та зручності взаємодії користувачів з вебзастосунком для продажу товарів або інтернет-магазином. Давайте розглянемо, чому ці функції є необхідними:

Забезпечення Контролю та Безпеки:

- Реєстрація: Дозволяє ідентифікувати користувача та встановлювати особистий обліковий запис.

- Авторизація: Забезпечує доступ до особистого облікового запису тільки авторизованим користувачам, зберігаючи таким чином конфіденційні дані.

Рис. 2.5 Порядок авторизації

Персоналізація та Зручність:

- Реєстрація: Дозволяє користувачам створити особистий профіль з основними даними та налаштуваннями.

- Авторизація: Надає доступ до персоналізованих функцій та можливостей для авторизованих користувачів.

Управління Замовленнями та Історією:

- Реєстрація: Створення облікового запису дозволяє зберігати історію замовлень та відстежувати їх стан.

- Авторизація: Надає можливість авторизованим користувачам переглядати та керувати своїми замовленнями.

Взаємодія з Клієнтами:

- Реєстрація: Дозволяє здійснювати ефективну комунікацію з користувачами через електронну пошту чи особистий кабінет.

- Авторизація: Забезпечує можливість надсилання сповіщень та оновлень, які стосуються особистого облікового запису.

Захист Інформації:

- Реєстрація: Дозволяє зберігати та захищати особисті дані користувачів від несанкціонованого доступу.

- Авторизація: Забезпечує захист особистих даних та дозволяє контролювати рівень доступу до інформації.

Можливість Спільного Використання та Рефералів:

- Реєстрація: Дозволяє користувачам об'єднувати свої облікові записи для спільного використання та реферальних програм.

- Авторизація: Надає можливість отримувати реферальні бонуси та використовувати спільні функції облікового запису.

Загалом, функції "Реєстрація" та "Авторизація" є важливими для створення персоналізованого та безпечного досвіду користувачів у вебзастосунку для продажу товарів.

Функціонал "**Особистий Кабінет Користувача**" є важливим елементом для забезпечення комфортного та індивідуалізованого досвіду користувачів у вебзастосунку для продажу товарів чи інтернет-магазині. Ось декілька ключових причин, чому цей функціонал є важливим:

Персоналізація та Зручність:

- Користувачі можуть зберігати свої особисті дані, адреси доставки, вибори та налаштування в одному місці, що робить їхній досвід більш зручним та персоналізованим.

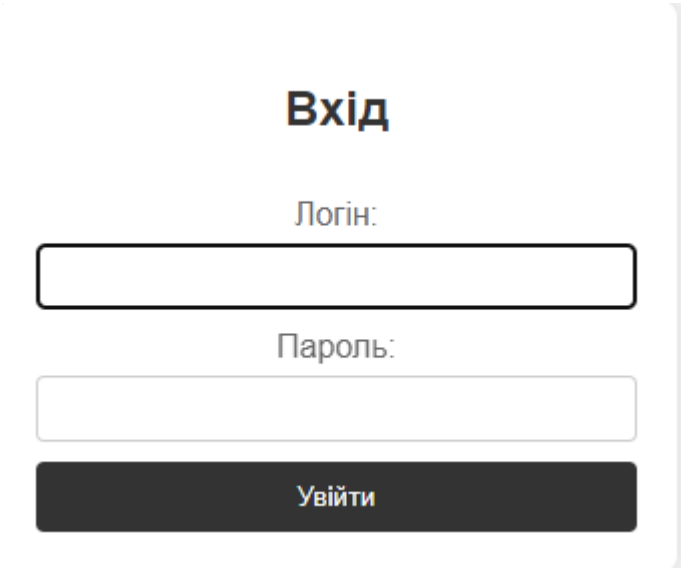

Рис. 2.6. Вигляд полів для входу

Управління Замовленнями:

- В особистому кабінеті користувач може переглядати історію своїх замовлень, статус доставки, докладну інформацію про кожен товар і здійснювати повернення чи обмін.

Керування Обліковим Записом:

- Можливість змінювати пароль, електронну адресу та інші дані дозволяє користувачам контролювати свій обліковий запис та забезпечує безпеку особистих даних.

Збереження Уподобань та Закладок:

- Користувачі можуть зберігати улюблені товари, створювати список бажань та використовувати це для зручного вибору та покупок.

Контроль Інформації:

- Користувачі можуть самостійно керувати тим, яку інформацію вони бажають отримувати від вебзастосунку, налаштовуючи підписки та повідомлення.

Взаємодія з Обслуговуванням Клієнтів:

- Зручний спосіб звертатися до служби підтримки, ставити питання та отримувати індивідуальний сервіс через особистий кабінет.

Участь в Програмах Лояльності:

- Користувачі можуть слідкувати за своїм статусом у програмах лояльності, отримувати бонуси та персоналізовані пропозиції через свій обліковий запис.

Аналітика та Звітність:

- Власники бізнесу можуть використовувати дані з особистих кабінетів для аналізу поведінки користувачів та покращення стратегій маркетингу.

Отже, особистий кабінет користувача є інструментом для покращення взаємодії та створення особистого та зручного досвіду для кожного користувача вебзастосунку.

Адміністративний розділ вебзастосунку для продажу товарів на базі операційної системи "Windows" є важливим елементом для здійснення управління та контролю за різними аспектами бізнесу. Ось декілька ключових причин, чому адміністративний розділ може бути важливим:

Управління Товарами та Категоріями:

- Додавання, видалення та редагування товарів, управління їх категоріями, характеристиками та іншими параметрами.

Аналітика та Звітність:

- Перегляд статистики продаж, аналіз популярності товарів, звіти про фінансові транзакції та інші дані, які допомагають в прийнятті стратегічних рішень.

Управління Замовленнями:

- Перегляд, підтвердження та відміна замовлень, взаємодія з клієнтами та службою підтримки.

Управління Користувачами:

- Додавання, блокування або видалення облікових записів користувачів, вирішення питань з безпекою та конфіденційністю.

Адміністрування Системи:

- Налаштування параметрів вебзастосунку, управління правами доступу для адміністраторів, моніторинг системи та її безпека.

Підтримка та Сервіс:

- Служба підтримки може використовувати адміністративний розділ для вирішення проблем клієнтів та надання їм інформації.

Акції та Промоції:

- Встановлення знижок, акцій та промо-кодів для привертання нових клієнтів та залучення популярності до конкретних товарів.

Інтеграція з Іншими Системами:

- Забезпечення можливості інтеграції з іншими сервісами або платформами для покращення функціональності та розширення можливостей додатку.

Адміністративний розділ дозволяє власникам бізнесу та адміністраторам ефективно управляти всією інфраструктурою вебзастосунку та надає можливість реагувати на зміни в бізнес-процесах.

Функція "**Відгуки**" вебзастосунку для продажу товарів на базі операційної системи "Windows" є важливим елементом для створення довіри серед користувачів та поліпшення їхнього досвіду. Ось кілька ключових причин, чому функція відгуків є важливою:

Довіра та Відкритість:

- Відгуки дозволяють клієнтам дізнатися думку інших людей про товар чи послугу, що сприяє відкритості та створенню довіри.

Покращення Покупок:

- Інші користувачі можуть скористатися досвідом інших у виборі товару, розумінні його переваг та недоліків.

Рис. 2.7. Відгук користувача

Стимулювання Продажів:

- Позитивні відгуки можуть слугувати додатковим стимулом для інших користувачів здійснити покупку, особливо якщо вони бачать, що інші користувачі задоволені товаром чи послугою.

Залучення Уваги:

- Зацікавленість користувачів часто привертається до відгуків, особливо якщо вони надають докладну інформацію та реальний досвід використання товару.

Збільшення Довгострокової Репутації:

- Позитивні відгуки сприяють формуванню доброї репутації бренду чи магазину, що може вплинути на його успіх у майбутньому.

Взаємодія з Клієнтами:

- Відгуки створюють можливість для взаємодії між користувачами та адміністраторами, де клієнти можуть ставити питання чи отримувати додаткові консультації.

Покращення Товарів та Послуг:

- Компанії можуть використовувати конструктивні відгуки для вдосконалення своїх товарів чи послуг, реагуючи на запитання та пропозиції користувачів.

Реалізація Контент-Маркетингу:

- Позитивні відгуки можуть стати частиною контент-маркетингу, де вони використовуються для просування товарів та виробників.

Загалом, наявність функції "Відгуки" дозволяє створити позитивне сприйняття серед клієнтів.

Функції "**Повідомлення та Сповіщення**" вебзастосунку для продажу товарів можуть бути корисними для поліпшення взаємодії з користувачами та забезпечення їм кращого досвіду використання.

Інформування про Акції та Знижки:

- Сповіщення можуть слугувати для повідомлення користувачів про акції, знижки та спеціальні пропозиції, що сприяє стимулюванню продажів.

Статус Замовлення:

- Клієнти можуть отримувати сповіщення про стан свого замовлення, такі як підтвердження, відправлення, доставка та інші оновлення.

Оповіщення про Новини та Оновлення:

- Сповіщення дозволяють інформувати користувачів про нові поступлення товарів, оновлення каталогу чи зміни в бізнес-процесах.

Важливі Повідомлення від Адміністрації:

- Адміністратори можуть використовувати повідомлення для надсилання важливої інформації користувачам, наприклад, щодо умов користування або змін у політиці конфіденційності.

Сповіщення про Відгуки та Рейтинги:

- Користувачі можуть отримувати сповіщення про нові відгуки чи рейтинги щодо товарів, які їх цікавлять.

Оповіщення про Помилки або Проблеми:

- Клієнти можуть бути сповіщені про можливі помилки чи проблеми з їхнім замовленням та отримати інструкції щодо їх вирішення.

Персоналізовані Сповіщення:

- Ви можете створювати персоналізовані повідомлення та сповіщення на основі попередніх покупок, інтересів користувача чи його поведінки на сайті.

Взаємодія та Ретенція Клієнтів:

- Сповіщення допомагають підтримувати активність користувачів, залучати їх до повторних покупок та утримувати їх інтерес до бренду.

Загалом, функції "Повідомлення" та "Сповіщення" можуть покращити комунікацію з користувачами та забезпечити їм більший контроль та зручність при використанні вебзастосунку.

Інтеграція з платіжними системами вебзастосунку для продажу товарів на базі операційної системи "Windows" є ключовою для забезпечення зручності платіжних операцій та створення надійної і безпечної платіжної інфраструктури.

Зручність для Користувачів:

- Забезпечення різноманітних методів оплати дозволяє користувачам обирати той, який є для них найзручнішим, що може включати кредитні картки, електронні гроші, банківські перекази та інші.

Збільшення Конверсії:

- Широкий вибір платіжних методів може сприяти збільшенню конверсії, оскільки користувачам необхідно менше зусиль для здійснення оплати.

Надійність та Безпека:

- Використання визначених платіжних систем гарантує надійність та безпеку фінансових транзакцій, оскільки вони дотримуються високих стандартів безпеки.

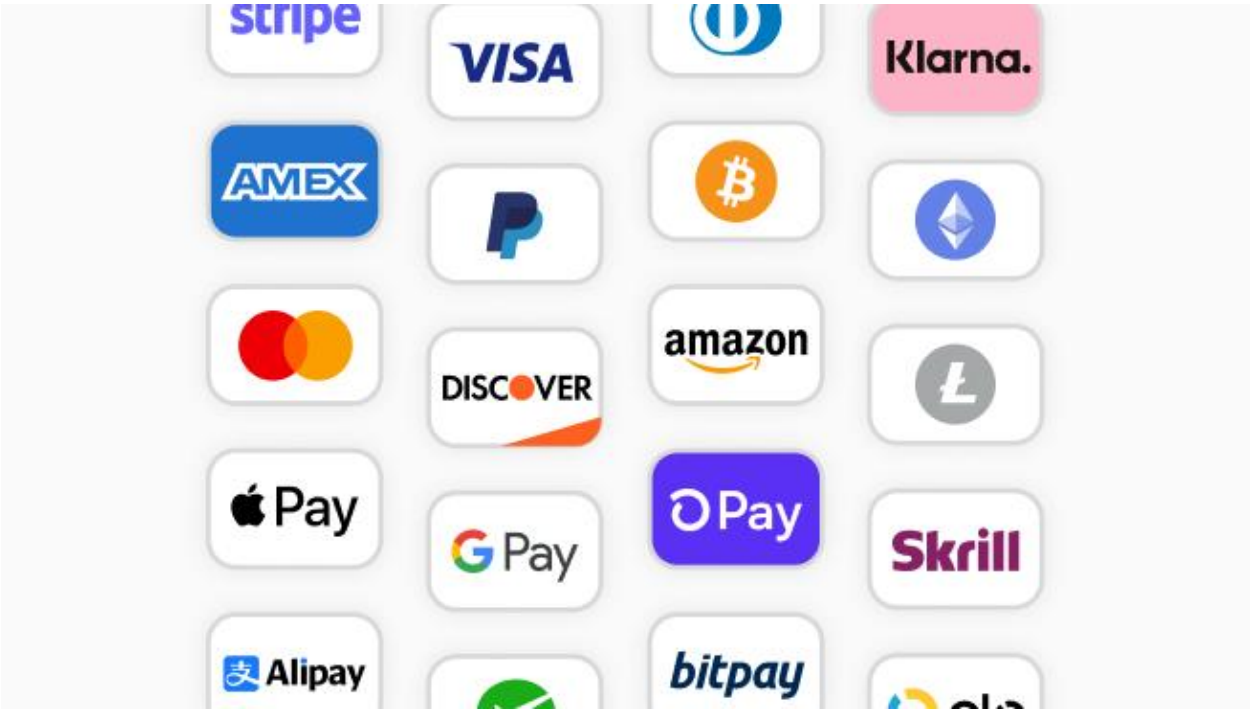

Рис. 2.8. Вибірка платіжних систем

Автоматизація Обробки Замовлень:

- Інтеграція з платіжними системами дозволяє автоматизувати процес обробки замовлень, сприяючи швидкому та ефективному виконанню оплат.

Відстеження та Звітність:

- Забезпечення доступу до відстеження стану платежів та генерації звітів, що дозволяє аналізувати фінансові транзакції та контролювати фінансовий стан бізнесу.

Міжнародні Платежі:

- Деякі платіжні системи підтримують міжнародні платежі, що розширює географію здійснення покупок та привертає клієнтів з різних країн.

Дотримання Законодавства:

варів на базі операційної системи "Windows" є важливим аспектом, оскільки він забезпечує оптимальний та зручний користувацький досвід на різних пристроях та розмірах екранів.

Широкий Діапазон Пристроїв:

- Користувачі використовують різні пристрої, такі як смартфони, планшети, ноутбуки та настільні комп'ютери. Адаптивний дизайн дозволяє оптимізувати вигляд та функціоналість для будь-якого пристрою.

Зручність для Користувача:

- Адаптивний дизайн покращує зручність використання, оскільки сторінка автоматично адаптується до розмірів екрану, запобігаючи необхідність прокрутки та зумування.

Покращення SEO:

- Пошукові системи віддають перевагу адаптивному дизайну, що може позитивно впливати на позиції в результатах пошуку.

Заощадження Часу та Ресурсів:

- Розробка одного адаптивного дизайну дозволяє уникнути необхідності створювати окремий дизайн для кожного типу пристрою.

Зменшення Відскоку:

- Користувачі на мобільних пристроях часто покидають сайти з поганим адаптивним дизайном. Його вдосконалення допомагає зменшити відсоток відскоку та збільшити задоволеність користувачів.

Підтримка Різних ОС:

- Адаптивний дизайн не обмежується конкретною операційною системою, що робить його універсальним та доступним для користувачів різних пристроїв. Можливість перегляду вебзастосунку з будь-якого пристрою високо цінується серед користувачів інтернету та допомогає витрачати менше часу за для проведення онлайн покупок чи навіть звичайного перегляду будь-чого .

Кожен користувач інтернету в наш час має смартфон , але не всі мають можливість використовувати комп’ютери для пошуку інформації.

Рис. 2.9. Кросс-платформ

Технологічний Розвиток:

- З ростом та розвитком технологій нові пристрої та розміри екранів стають доступнішими. Адаптивний дизайн готовий до швидкого впровадження на нових пристроях без значних змін.

Конкурентні Переваги:

- Багато користувачів вважають адаптивний дизайн ознакою професіоналізму та передового підходу, що може підвищити конкурентоспроможність.

Загалом, адаптивний дизайн дозволяє забезпечити високий рівень комфорту та задоволення для користувачів незалежно від пристрою, яким вони користуються.

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі були описані всі вимоги та пояснення їх необхідності в функціоналі до застосунку.

Додаток має різноманітний функціонал, включаючи перегляд товарів, додавання їх до кошика, оформлення замовлення, реєстрацію та авторизацію користувачів. Важливим елементом є адаптивний дизайн для зручного використання на різних пристроях.

Було розглянуто з використанням сучасного стеку технологій, що використовується при створенні вебдодатків з допомогою операційної системи Windows з використанням сучасного середовища Visual Studio Code.

За допомогою розгорнутих у другому розділі прикладах функціоналу, адаптивний дизайн та інтеграція з платіжними системами створюють зручне та сучасне середовище для користувачів. Розглянуті аспекти інтеграції з операційною системою роблять додаток оптимізованим для використання на пристроях з ОС "Windows" і не тільки. Вивчення цих тем дозволяє забезпечити успішну реалізацію та популяризацію вебзастосунку на ринку електронної комерції. Також слід зазначити, що програма відповідає як описаним, так і загальноприйнятим вимогам до вебзастосунків.

РОЗДІЛ 3
РОЗРОБКА ТА ТЕСТУВАННЯ ВЕБЗАСТОСУНКУ

3.1. Головна сторінка

Розробка головної сторінки вебзастосунку для продажу домашнього одягу є ключовим етапом в процесі створення користувацького інтерфейсу та взаємодії з користувачем.

У цьому розділі розглянуто етапи розробки головної сторінки, включаючи визначення структури, вибір дизайну, додавання контенту та навігаційних елементів.

Визначення мети та структури:

Першим етапом було визначено основні цілі головної сторінки. Вона має приваблювати увагу користувачів, швидко передавати основну інформацію про продукцію та забезпечувати зручну навігацію.

Структура сторінки була розроблена так, щоб легко впоратися із збільшенням обсягу товарів та інших функцій.

Приклад написання початку сторінки:

```
<!DOCTYPE html>
<html lang="en">

<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Магазин Домашнього Одягу</title>
```

Вибір дизайну та стилів:

Дизайн сторінки був розроблений з урахуванням сучасних тенденцій та естетичних вимог. Використання світлих кольорів сприяє враженню про чистоту та комфорт, а анімаційні ефекти додають динамізму.

Додавання контенту:

```
 <div class="product">
        <img src="" alt="Товар 1">
        <h3>Футболка "Зручна Літнього Дня"</h3>
        <p>Колір: Синій</p>
        <p>Розмір: S, M, L</p>
        <p>Ціна: $20</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>

    <div class="product">
        <img src="" alt="Товар 2">
        <h3>Шорти "Легкі та Стильні"</h3>
        <p>Колір: Чорний</p>
        <p>Розмір: M, L, XL</p>
        <p>Ціна: $25</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>

    <div class="product">
        <img src="" alt="Товар 3">
        <h3>Спідниця "Елегантна Класика"</h3>
        <p>Колір: Сірий</p>
        <p>Розмір: S, M, L</p>
        <p>Ціна: $30</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>


<style>
    body {
```

```
    font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f6f6f6;
    color: #333;
}

header {
    background-color: #2629ce;
    color: #fff;
    padding: 1em;
    text-align: center;
}

nav {
    display: flex;
    justify-content: space-around;
    background-color: #333;
    padding: 0.5em;
}

nav a {
    color: #fff;
    text-decoration: none;
    padding: 0.5em 1em;
    border-radius: 5px;
    transition: background-color 0.3s;
}

nav a:hover {
    background-color: #555;
}

section {
    padding: 2em;
    display: flex;
    flex-wrap: wrap;
    justify-content: space-around;
}
```

Зображення продукції були додані з врахуванням візуальної привабливості та відображення різноманіття асортименту. Текстовий контент був створений із фокусом на зручне сприйняття та коротке, але інформативне представлення кожного товару.

Створення навігаційних елементів:

```
<body>
  <header>
```

```
    <h1>Магазин Домашнього Одягу</h1>
    <p>Одяг для комфорту та стилю</p>
  </header>

  <nav>
    <a href="#">Відгуки</a>
    <a href="#">Каталог</a>
    <a href="">Кошик</a>
    <a href="#">Для нього</a>
    <a href="#">Для неї</a>
  </nav>
```

Навігаційні елементи, такі як меню та посилання, були розміщені так, щоб забезпечити легкий доступ до основних розділів, таких як відгуки, каталог, кошик, для нього та для неї. Це робить взаємодію з додатком інтуїтивно зрозумілою для користувача.

Оптимізація для різних пристроїв:

```
@media only screen and (max-width: 768px) {
    /* Стилі для пристроїв з шириною екрану до 768px */
    body {
        font-size: 14px;
    }
}

@media only screen and (min-width: 769px) and (max-width: 1024px) {
    /* Стилі для планшетів з шириною екрану від 769px до 1024px */
    body {
        font-size: 16px;
```

Гнучкі контейнери:

Використання відсоткових значень для ширини та позиціонування елементів, щоб контент гнучко розтягувався та адаптувався до розмірів екрану.

```
.container {
    width: 100%;
    max-width: 1200px; /* Максимальна ширина контейнера */
    margin: 0 auto; /* Вирівнювання по центру */
}
```

Використання векторних зображень або іконок, які легко масштабуються без втрати якості. Для різних розмірів екрану можна завантажувати різні розміри зображень.

```
src "image.jpg" alt "Опис зображення" class "responsive-image"
```

Враховуючи різні розміри екранів, була використана техніка адаптивного дизайну. Це забезпечує оптимальний вигляд та функціонал на різних пристроях, включаючи комп'ютери, планшети та смартфони.

В результаті цього процесу була створена головна сторінка, яка поєднує в собі естетичний дизайн та високий рівень функціональності, забезпечуючи користувачам приємний та зручний досвід взаємодії з веб-додатком.

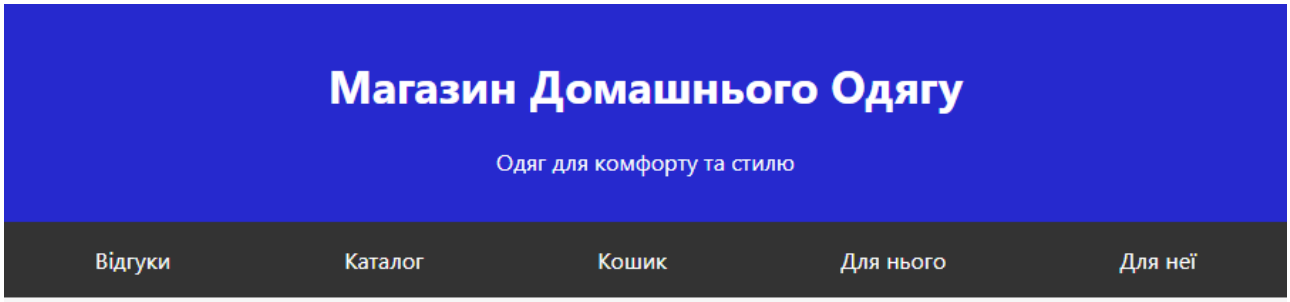

Рис. 3.1. Початковий вигляд головної сторінки

Карусель є важливим елементом на головній сторінці вебзастосунку, оскільки вона дозволяє ефективно відображати різноманітні зображення та привертати увагу користувачів. Її використання має кілька переваг:

Привабливість: Карусель дозволяє створити візуально привабливий ефект, використовуючи зміну зображень або слайдів.

Розробка стилю каруселі мовою CSS:

```
.carousel {
        max-width: 100%;
        max-height: 600px;
        overflow: hidden;
        margin: auto;
    }

    .carousel img {
        width: 100%;
        display: block;
```

```
        }
.carousel-item img {
  display: block;
  height: auto;
  max-width: 100%;
}

.carousel-control {
  position: absolute;
  top: 50%;
  transform: translateY(-50%);
  font-size: 2rem;
  color: #fff;
  opacity: 0.5;
  cursor: pointer;
}

.carousel-control:hover {
  opacity: 0.9;
}

.carousel-indicators {
  position: absolute;
  bottom: 2%;
  left: 0;
  right: 0;
  text-align: center;
}

.carousel-bullet {
  display: inline-block;
  width: 15px;
  height: 15px;
  border-radius: 50%;
  background-color: #aaa;
  margin: 0 5px;
  cursor: pointer;
  opacity: 0.5;
}

.carousel-bullet:hover {
```

```
  opacity: 0.9;
}

    </style>
</head>
```

Простота Навігації: Користувачам легко переглядати різні зображення, прокручуючи карусель, що полегшує їхню навігацію.

Розробка каруселі мовою HTML:

```
 <div class="carousel">
            <div class="carousel-inner">
              <input class="carousel-open" type="radio" id="carousel-
1" name="carousel" aria-hidden="true" hidden="" checked="checked">
              <div class="carousel-item">
                <img src="">
              </div>
              <input class="carousel-open" type="radio" id="carousel-
2" name="carousel" aria-hidden="true" hidden="">
              <div class="carousel-item">
                <img src="">
              </div>
              <input class="carousel-open" type="radio" id="carousel-
3" name="carousel" aria-hidden="true" hidden="">
              <div class="carousel-item">
                <img src="">
              </div>

              <label for="carousel-3" class="carousel-control prev
control-1">‹</label>
              <label for="carousel-2" class="carousel-control next
control-1">›</label>
              <label for="carousel-1" class="carousel-control prev
control-2">‹</label>
              <label for="carousel-3" class="carousel-control next
control-2">›</label>
              <label for="carousel-2" class="carousel-control prev
control-3">‹</label>
              <label for="carousel-1" class="carousel-control next
control-3">›</label>
            </div>

            <ol class="carousel-indicators">
```

```
            <li>
              <label for="carousel-1" class="carousel-
bullet">•</label>
            </li>
            <li>
              <label for="carousel-2" class="carousel-
bullet">•</label>
            </li>
            <li>
              <label for="carousel-3" class="carousel-
bullet">•</label>
            </li>
          </ol>
        </div>
```

Рис. 3.2. Карусель перший слайд

Для того , щоб зміна доданих фотографій виконувалась в автоматичному порядку потрібно написати код мовою JavaScript , ось приклад для мого вебзастосунку:

```
 const carousel = document.querySelector('.carousel');
const interval = 5000; // час показу слайду

let currentSlide = 0;

function changeSlide() {
  currentSlide++;

  if (currentSlide > carousel.querySelectorAll('.carousel-
item').length - 1) {
    currentSlide = 0;
  }

  carousel.querySelector('.carousel-open:checked')
    .removeAttribute('checked');
  carousel.querySelectorAll('.carousel-open')[currentSlide]
    .setAttribute('checked', 'checked');
}
setInterval(changeSlide, interval);
    </script>
</body>
```

Рис. 3.3. Карусель 2 слайд

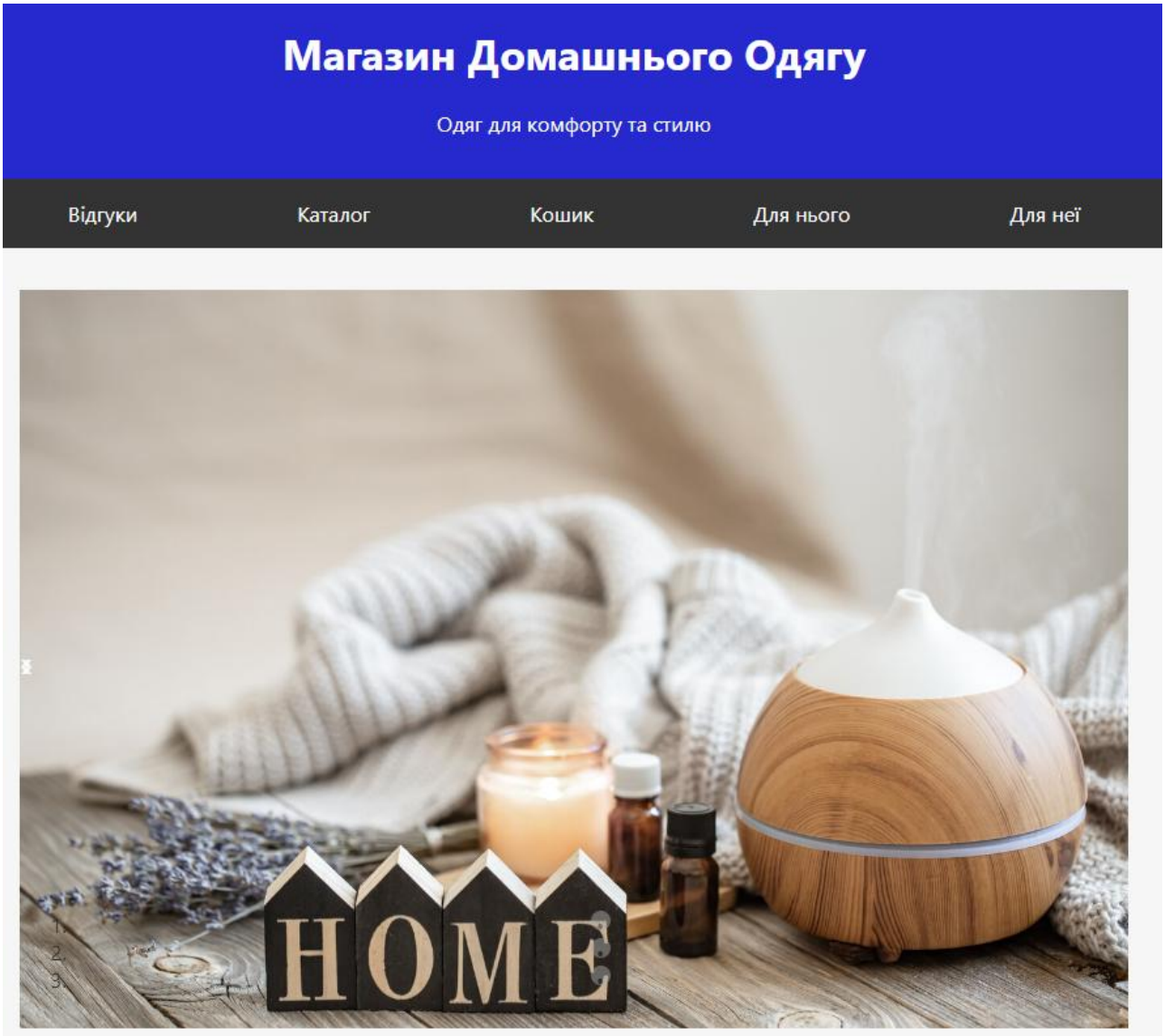

Рис. 3.4. Карусель 3 слайд

Секції на головній сторінці вебзастосунку виконують важливу роль у структуруванні та представленні контенту. Кожна секція призначена для конкретної групи інформації та виконує визначену функцію. Це полегшує навігацію користувача, надає зручний доступ до ключових функцій та інформації. Розглянемо кілька секцій та їхні функції:

Секція Каталогу:

- Мета: Представлення основного асортименту товарів.
- Функції: Відображення мініатюр товарів, їхніх назв, цін та основних характеристик.

```
<div class="product">
```

```
        <img src="" alt="Товар 1">
        <h3>Футболка "Зручна Літнього Дня"</h3>
        <p>Колір: Синій</p>
        <p>Розмір: S, M, L</p>
        <p>Ціна: $20</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>

    <div class="product">
        <img src="" alt="Товар 2">
        <h3>Шорти "Легкі та Стильні"</h3>
        <p>Колір: Рожевий</p>
        <p>Розмір: M, L, XL</p>
        <p>Ціна: $25</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>

    <div class="product">
        <img src="" alt="Товар 3">
        <h3>Спідниця "Елегантна Класика"</h3>
        <p>Колір: Сірий</p>
        <p>Розмір: S, M, L</p>
        <p>Ціна: $30</p>
        <button onclick="addToCart()">Додати в кошик</button>
        <button onclick="showDetails(this)">Деталі</button>
    </div>
```

Секція Відгуків:

- Мета: Збір та відображення відгуків користувачів про товари чи сервіс.

```
     id "reviews">
    <h2>Відгуки</h2>
    <p>Діліться своїми враженнями про наш одяг</p>
    <fom>
        <input type="text" placeholder="Ваше ім'я" required>
        <textarea placeholder="Ваш відгук" rows="4" required></textarea>
        <button type="submit">Додати відгук</button>
    </form>
</section>
```

- Функції: Можливість додавання відгуків, відображення рейтингів та коментарів.

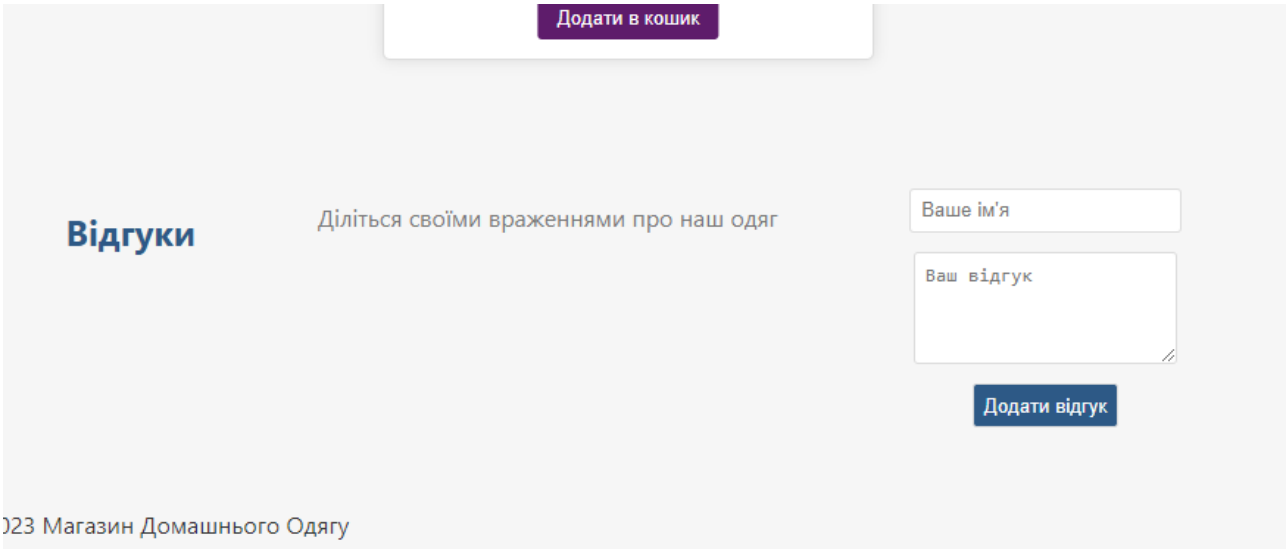

Рис. 3.5. Вигляд розділу з відгуками

Секція Кошика:

- Мета: Перегляд вибраних товарів та підготовка до оформлення замовлення.

```
#cart {
        padding: 2em;
        background-color: #fff;
        margin: 1em;
        border: 1px solid #ddd;
        border-radius: 5px;
        box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
        text-align: center;
    }

    #cart h2 {
        color: #2d5986;
    }

    #cart p {
        color: #777;
        margin: 1em 0;
    }
```

```
#cart ul {
    list-style: none;
    padding: 0;
}

#cart li {
    display: flex;
    justify-content: space-between;
    align-items: center;
    border-bottom: 1px solid #ddd;
    margin: 0.5em 0;
    padding-bottom: 0.5em;
}

#cart li img {
    max-width: 50px;
    max-height: 50px;
    margin-right: 1em;
}

#cart total {
    font-weight: bold;
    margin-top: 1em;
}
```

- Функції: Додавання та видалення товарів, відображення загальної суми.

```
<section id="cart">
    <h2>Кошик</h2>
    <ul>
        <!-- Cart items will be dynamically added here -->
    </ul>
    <p id="total">Загальна сума: $0</p>
</section>
```

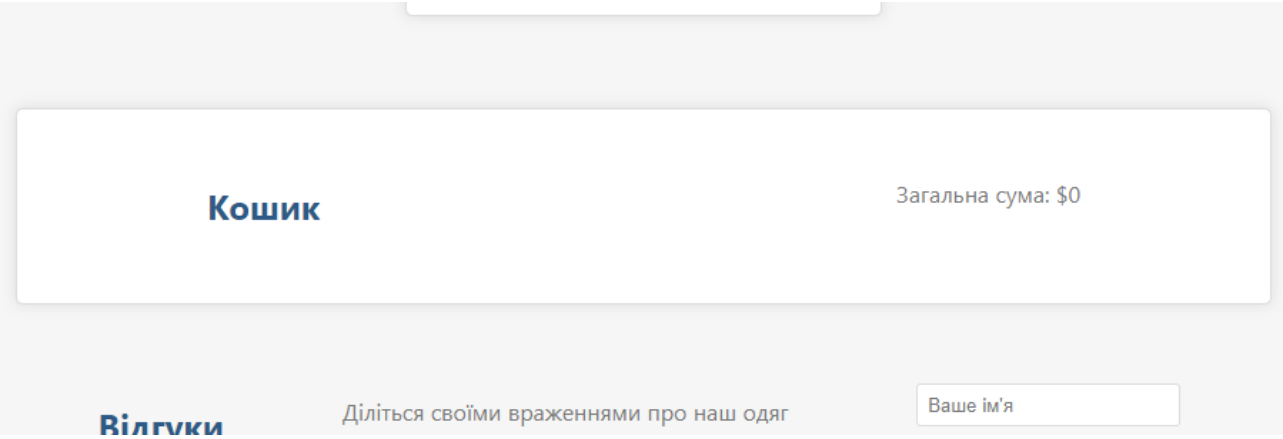

Рис. 3.6. Кошик

Секція товарів потрібна для одного з найголовнішого , вибіру товарів за для яких і був здійснений вхід до вебзастосунку , вибір потрібного товару , підбір розміру та перевірка ціни , можливість додати товар до кошику та побачити що товар додано , після чого зробити легку покупку та мати приємний досвід від користування вебзастосунку , код для секцій :

```
.product {
            margin: 1em;
            border: 1px solid #ddd;
            padding: 1em;
            text-align: center;
            background-color: #fff;
            border-radius: 5px;
            box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
            transition: transform 0.3s;
            width: 200px;
            position: relative;
            overflow: hidden;
         }
```

```
<div class="product">
      <img src="" alt="Товар 2">
      <h3>Шорти "Легкі та Стильні"</h3>
      <p>Колір: Рожевий</p>
      <p>Розмір: M, L, XL</p>
      <p>Ціна: $25</p>
```

```
            <button  class="add-to-cart-btn"  onclick="addToCart()">Додати  в
кошик</button>


        </div>

        <div class="product">
            <img src="">
            <h3>Спідниця "Елегантна Класика"</h3>
            <p>Колір: Сірий</p>
            <p>Розмір: S, M, L</p>
            <p>Ціна: $30</p>
            <button onclick="addToCart()">Додати в кошик</button>

        </div>

        <!-- Додайте ще товарів за аналогією -->

        <div class="details" id="details">
            <img src=".jpg" alt="Товар 1">
            <h3>Футболка "Літнього Дня"</h3>
            <p>Колір: Синій</p>
            <p>Розмір: S, M, L</p>
            <p>Ціна: $20</p>
            <button onclick="addToCart()">Додати в кошик</button>
            <button onclick="hideDetails()">Закрити</button>
        </div>
```

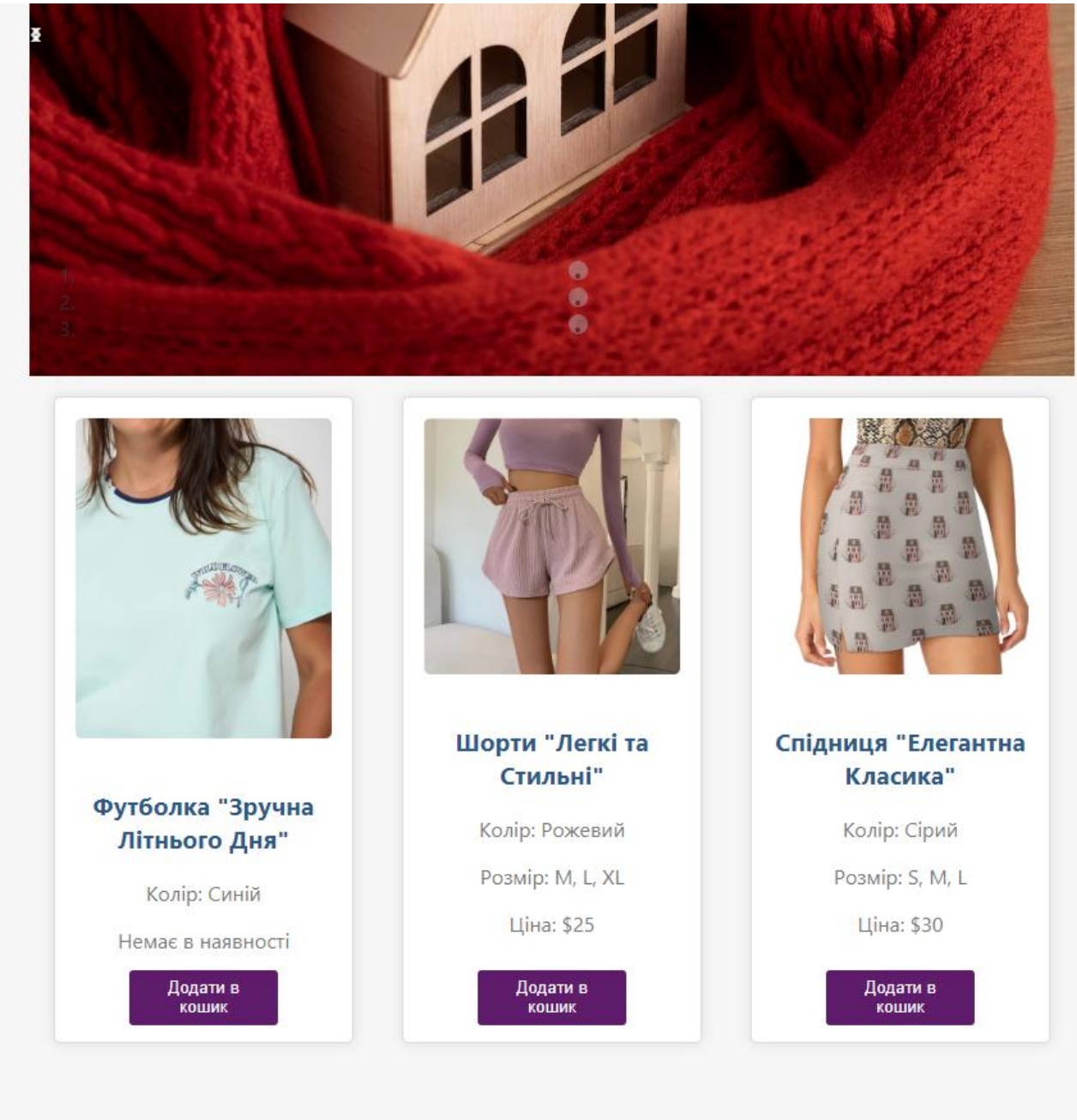

Рис. 3.7. Вигляд секції товарів

3.2. Аутентифікація , авторизація та реєстрація

Аутентифікація:

Значення та Потреба:

Аутентифікація – це процес перевірки ідентичності користувача для забезпечення доступу до системи. Важливою є безпека цього процесу, оскільки вона визначає, чи є особа, яка входить, таємною та чи вона має право на доступ.

Реалізація:

- Використання криптографічно безпечних хеш-функцій для зберігання паролів.

Для використання потрібно встановити біліотеку bcrypt :

npm install bcrypt

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

const saltRounds = 10; // Кількість ітерацій
const plaintextPassword = 'user_password';

bcrypt.genSalt(saltRounds, function(err, salt) {
    bcrypt.hash(plaintextPassword, salt, function(err, hash) {
        // 'hash' - захешований пароль, який можна зберегти в базі даних
        console.log('Hashed Password:', hash);
    });
});
```

- Встановлення механізмів блокування облікових записів після декількох невдалих спроб входу.

Встановлення бібліотек express-rate-limit:

npm install express express-session express-rate-limit

```
const express = require('express');
const session = require('express-session');
const RateLimit = require('express-rate-limit');

const app = express();

// Налаштування сесій
app.use(session({
    secret: 'your_secret_key',
    resave: false,
    saveUninitialized: true
}));

// Налаштування обмеження частоти запитів
const loginLimiter = new RateLimit({
    windowMs: 15 * 60 * 1000, // 15 хвилин
```

```
    max: 5, // Максимальна кількість спроб
    delayMs: 0, // Затримка перед новою спробою
    message: 'Спробуйте пізніше'
});

// Маршрут для обробки входу
app.post('/login', loginLimiter, (req, res) => {
    const username = req.body.username;
    const password = req.body.password;

    // Перевірка пароля та інша логіка входу

    // Якщо вхід успішний, скидаємо ліміт спроб
    req.session.loginAttempts = 0;

    // Редірект або відповідь користувачеві
});

// При кожній невдачі збільшуємо кількість спроб
app.post('/login', (req, res) => {
    req.session.loginAttempts = (req.session.loginAttempts || 0) + 1;

    // Додаткова логіка для блокування облікового запису
    if (req.session.loginAttempts >= 5) {
        // Заблокувати обліковий запис на певний час
        req.session.blockedUntil = Date.now() + 15 * 60 * 1000; // Блокувати
на 15 хвилин
    }

    // Відповідь користувачеві
});

// Додаткова логіка для перевірки, чи обліковий запис заблокований
app.use((req, res, next) => {
    if (req.session.blockedUntil && req.session.blockedUntil > Date.now()) {
        return res.status(403).send('Обліковий запис заблоковано, спробуйте
пізніше');
    }
    next();
```

```
});
```

- Використання двофакторної аутентифікації для підвищення рівня безпеки.

Важливість:

Аутентифікація гарантує, що лише правомірні користувачі мають доступ до ресурсів системи, а їхні дані залишаються конфіденційними.

Авторизація:

Значення та Потреба:

Авторизація – це процес надання користувачеві прав на використання конкретних ресурсів чи функціоналу після успішної аутентифікації.

Реалізація:

- Визначення рівнів доступу для різних груп користувачів (наприклад, адміністратор, звичайний користувач, гість).

- Контроль доступу на основі ролей та прав.

Важливість:

Авторизація гарантує, що користувач має доступ лише до тих функцій і ресурсів, які відповідають його ролі або правам.

Реєстрація:

Значення та Потреба:

Реєстрація – це процес створення нового облікового запису користувача в системі.

Реалізація:

- Збір основних інформаційних даних від користувача (логін, пароль, електронна пошта тощо).

- Перевірка унікальності ідентифікаторів (логінів, електронних адрес).

- Висилання підтверджувального листа для завершення реєстрації.

Встановлення бібліотек та створення сторінки реєстрації :

```
npm install express express-session body-parser
```

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

```
const session = require('express-session');
const bodyParser = require('body-parser');

const app = express();
const port = 3000;

app.use(session({ secret: 'your_secret_key', resave: false, saveUninitialized: true }));
app.use(bodyParser.urlencoded({ extended: true }));

app.get('/register', (req, res) => {
    res.sendFile(__dirname + '/register.html');
});

app.post('/register', (req, res) => {
    const username = req.body.username;
    const password = req.body.password;



    // Приклад збереження користувача
    const newUser = {
        username: username,
        password: password // Важливо: пароль слід шифрувати перед збереженням у базі даних
    };



    res.send('Реєстрація успішна!');
});

app.listen(port, () => {
    console.log(`Server is running on port ${port}`);
});
```

Важливість:

Реєстрація дозволяє користувачам створювати власні облікові записи та використовувати персоналізований функціонал системи.

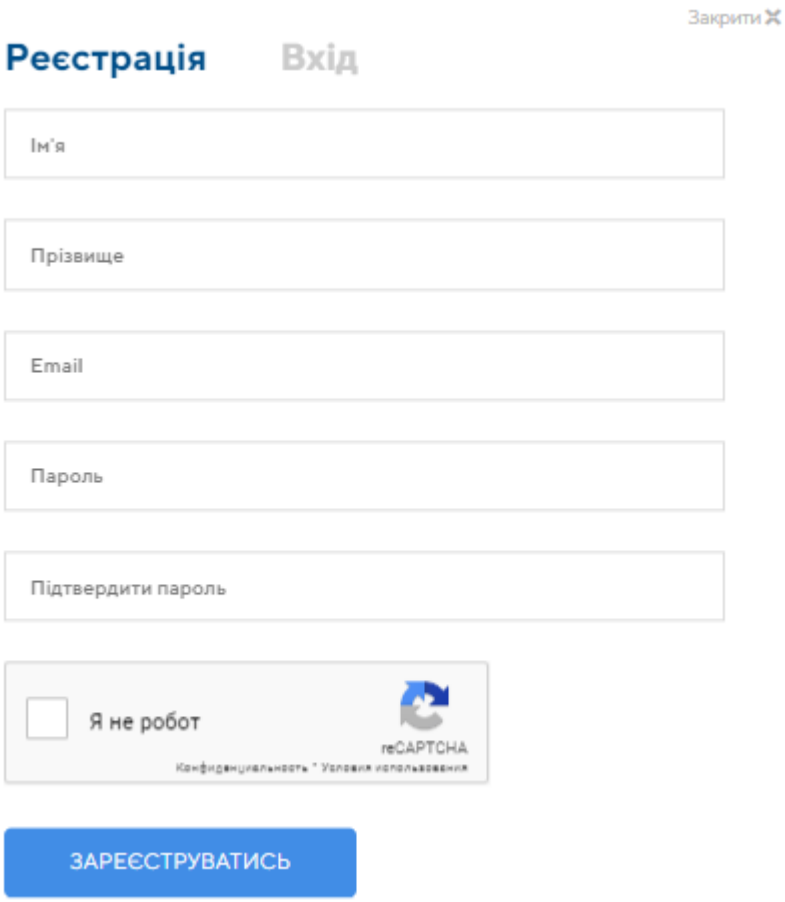

Рис. 3.8. Вхід та реєстрація

3.3. Інтеграція платіжних систем.

Інтеграція платіжних систем важлива для зручних та безпечних онлайн-платежів. Вона забезпечує:

Зручність користувачів: Легкість та швидкість оплати через вебзастосунок.

Безпеку інформації: Захист фінансових даних на безпечних серверах платіжних провайдерів.

Різноманітність платіжних методів: Приймання різних видів платежів.

Аналітику та відслідковування: Можливість аналізу фінансових транзакцій для стратегічного вдосконалення бізнесу.

Приклад коду для інтеграції платіжних систем, яка автоматично визначає, яка карта була введена :

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Інтеграція платіжної системи</title>
    <script src="https://js.stripe.com/v3/"></script>
    <style>
        body {
            font-family: 'Segoe UI', Tahoma, Geneva, Verdana, sans-serif;
            margin: 0;
            padding: 0;
            background-color: #f6f6f6;
            color: #333;
            display: flex;
            align-items: center;
            justify-content: center;
            height: 100vh;
        }

        #paymentForm {
            width: 400px;
            padding: 20px;
            background-color: #fff;
            border: 1px solid #ddd;
            border-radius: 5px;
        }

        #cardElement {
            margin-bottom: 20px;
        }

        #submitBtn {
            background-color: #4caf50;
            color: #fff;
            padding: 10px;
            border: none;
            border-radius: 3px;
            cursor: pointer;
        }

        #submitBtn:hover {
```

```
            background-color: #45a049;
        }
    </style>
</head>
<body>

<div id="paymentForm">
    <div id="cardElement"></div>
    <button id="submitBtn">Заплатити</button>
</div>

<script>
    // Ініціалізація Stripe
    const stripe = Stripe('публічний_ключ_stripe');
    const elements = stripe.elements();
    const cardElement = elements.create('card');

    // Додавання карти до сторінки
    cardElement.mount('#cardElement');

    // Обробник натискання на кнопку
    document.getElementById('submitBtn').addEventListener('click', async ()
=> {
        const { token, error } = await stripe.createToken(cardElement);

        if (error) {
            console.error(error);
        } else {
            // Відправка токену на сервер для обробки платежу

            console.log(token);
        }
    });
</script>

</body>
</html>
```

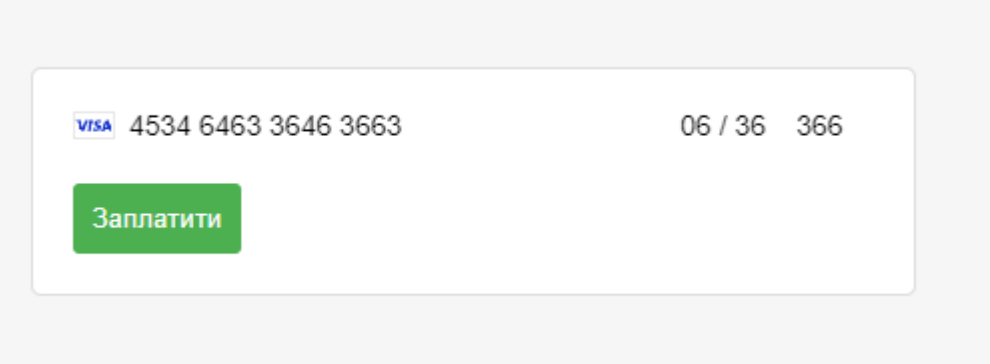

Рис. 3.9. Вигляд введення даних карти VISA.

На рис. 3.9. продемонстровано введення даних карти клієнта будь-якого банку з використанням платіжної системи VISA , також на рис. 3.10. можна побачити використання клієнтом платіжної системи MasterCard.

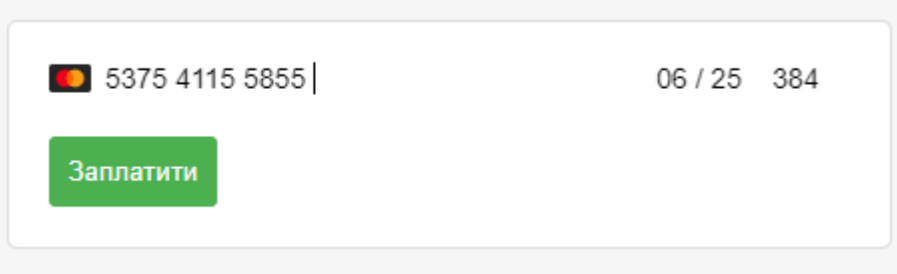

Рис. 3.10. Вигляд введення даних карти MasterCard

3.4. Створення бази даних SQL

Для правильного функціонування готового вебзастосунку потрібно створити базу даних для постійного оновлення асортименту та змін в користуванні в реальному часі , для створення бази даних використовується мова програмування SQL , приклад бази даних вебзастосунку:

```
CREATE DATABASE IF NOT EXISTS clothing_store;
-
USE clothing_store;

-
CREATE TABLE IF NOT EXISTS products (
    product_id INT AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR(255) NOT NULL,
    category VARCHAR(50) NOT NULL,
    color VARCHAR(50),
    size VARCHAR(10),
```

```
    price DECIMAL(10, 2) NOT NULL
);

-
INSERT INTO products (name, category, color, size, price)
VALUES
    ('Футболка "Зручна Літнього Дня"', 'Футболки', 'Синій', 'M', 20.00),
    ('Шорти "Легкі та Стильні"', 'Шорти', 'Чорний', 'L', 25.00),
    ('Спідниця "Елегантна Класика"', 'Спідниці', 'Сірий', 'S', 30.00);
```

ВИСНОВКИ ДО РОЗДІЛУ 3

В даному розділі було описано алгоритм роботи та розроблено веб-застосунок для продажу домашнього одягу на базі операційної системи "Windows". Застосунок написано на мовах програмування HTML, CSS, та JavaScript. Використано підходи та технології, які сприяють створенню зручного та інтуїтивно зрозумілого інтерфейсу для користувачів.

При розробці вебзастосунку було враховано принципи адаптивного дизайну для оптимального відображення на різних пристроях. Додатково встановлено функціонал, який включає в себе можливість перегляду деталей продуктів, додавання їх до кошика та роботу з кошиком покупок.

Технічний стек включає в себе HTML, CSS та JavaScript для фронтенду, а також розгортання на операційній системі "Windows". Серед інших важливих аспектів розглянуто інтеграцію з операційною системою, вибір веб-сервера та системні ресурси.

Оглянуто можливості інтеграції з операційною системою "Windows", використання служб та API, а також процеси інсталяції та оновлення вебзастосунку. Звернута увага на аспекти безпеки, зокрема аутентифікацію та авторизацію користувачів.

Завершено розділ із загальними висновками, де висвітлено ключові моменти розробки вебзастосунку та його аспекти, що впливають на користувацький досвід та ефективність роботи.

ВИСНОВКИ

У кваліфікаційній роботі був розроблений вебзастосунок для магазину домашнього одягу було враховано численні аспекти, спрямовані на створення ефективного та привабливого інтернет-простору для продажу одягу.

Було проведено глибокий аналіз та обрання оптимального технологічного стеку, що включав у себе не лише базові елементи веб-розробки, але й передові рішення для підвищення продуктивності та швидкості завантаження сторінок

Важливим етапом є вибір та використання потрібних технологій для розробки вебзастосунку. Використання таких інструментів, як HTML, CSS, JavaScript, та серверних мов програмування, може значно полегшити створення ефективного та відзначеного дизайну веб-сайту.

Ефективний та привабливий дизайн грає ключову роль у залученні та утриманні клієнтів. Важливо ретельно продумати інтерфейс користувача, забезпечити легкий доступ до основних функцій та створити приємний користувацький досвід.

Для оптимізації роботи магазину була використана інтеграція з платіжними системами, що дозволяє здійснювати зручні та безпечні транзакції. Також, використання соціальних мереж та інших сервісів може підвищити взаємодію з клієнтами.

Забезпечення повноцінної роботи вебзастосунку на різних мобільних пристроях, що дозволяє користувачам зручно переглядати та здійснювати покупки на різних платформах.

Було забезпечено легке та зручне управління каталогом товарів, можливість швидкого пошуку та фільтрації товарів сприяє покращенню користувацького досвіду. Управління замовленнями має бути ефективним та легким для використання як для клієнтів, так і для адміністраторів.

Забезпечення безпеки та конфіденційності особистих даних клієнтів є критичним завданням. Використання захисних механізмів, таких як шифрування

та механізми аутентифікації, є важливою складовою для збереження довіри користувачів.

Було проведено інтеграцію надійних платіжних систем та оптимізація процесів доставки для забезпечення зручності та безпеки транзакцій для клієнтів.

Впровадження стратегій пошукової оптимізації для підвищення видимості магазину в пошукових системах та залучення нових клієнтів.

З огляду на зростання використання мобільних пристроїв, важливо забезпечити адаптованість вебзастосунку до різних розмірів екранів. Мобільна дружність може підвищити досягнення цільової аудиторії.

В ході виконання кваліфікаційної роботи було створено вебзастосунок, який відповідає вимогам сучасного інтернет-простору, пропонуючи зручність та ефективність у взаємодії для якнайбільшого кола користувачів.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ніксон Р. Створюємо динамічні веб-додатки за допомогою PHP, MySQL, JavaScript, CSS і HTML5, 2020. - 510

2. Деріл Бартлетт. WordPress для початківців, 2019. – 320с.

3. Б. Вільямс, Д. Демстра, Х. Стерн. WordPress для професіоналів. Розробка і дизайн сайтів, 2019 – 412с.

4. Hrytsiuk, Yu I., та T. P. Biletskyi. "Веб-додаток для маркетингового аналізу соціальної мережі Instagram". Scientific Bulletin of UNFU 29, № 6 (27 червня 2019): 106–18. http://dx.doi.org/10.15421/40290622.

5. Феній, Н. С., та Ю. І. Грицюк. "Автоматизація процесу класифікації текстових новин з інтернет-сайтів методами нейронної мережі". Scientific Bulletin of UNFU 30, № 4 (17 вересня 2020): 123–33. http://dx.doi.org/10.36930/40300421.

6. Demianenko, V., та N. Ichanska. "ВИКОРИСТАННЯ СУЧАСНИХ ВЕБ-ТЕХНОЛОГІЙ ДЛЯ СИСТЕМИ КОНТРОЛЮ ТА МОНІТОРИНГУ ЗНАНЬ СТУДЕНТІВ". Системи управління, навігації та зв’язку. Збірник наукових праць 2, № 54 (11 квітня 2019): 83–86. http://dx.doi.org/10.26906/sunz.2019.2.083.

7. Tkachuk, V. "PWA, як перспективний напрямок об'єднання веб та мобільних технологій." COMPUTER-INTEGRATED TECHNOLOGIES: EDUCATION, SCIENCE, PRODUCTION, № 46 (14 квітня 2022): 83–87. http://dx.doi.org/10.36910/6775-2524-0560-2022-46-12.

8. СУБД MySQL і доступ до БД в PHP [Електронний ресурс]: http://www.znannya.org/?view=mysqli-intro

9. Leshchynskyi Volodymyr. Principles of explanation in e-commerce system based on sales dynamics. COMPUTER AND INFORMATION SYSTEMS AND TECHNOLOGIES KHARKIV, APRIL 2020, p.76-77.

10. Bielievtsov, I. Ruban, K. Smelyakov, D. Sumtsov Network technology for transmission of visual information // Selected Papers of the XVIII International Scientific and Practical Conference on Information Technologies and Security (ITS

2019). – CEUR Workshop Processing. – Kyiv, Ukraine, November 27, 2019. – Pp. 160-175.

11. Веб застосунок [Електронний ресурс]. – Точка доступу: URL: http://uk.wikipedia.org/wiki/Веб-застосунок– Веб застосунок.

12. Knack – easy online database apps [Електронний ресурс]. – Точка доступу: URL: https://www.knackhq.com/ – how does knack work?

13. Програмне забезпечення персональних комп'ютерів [Електронний ресурс]. – Точка доступу: URL: http://www.victoria.lviv.ua/html/oit/html/lesson8.htm – Програмне забезпечення

14. Google Cloud Computing [Електронний ресурс]. – Точка доступу: URL: https://cloud.google.com – How Google Cloud Platform works.

15. Автоматизація [Електронний ресурс]. – Точка доступу: URL: http://uk.wikipedia.org/wiki/Автоматизація – Автоматизація

16. HTML Підручник. Початок. Уроки для початківців. W3Schools українською [Електронний ресурс] – Режим доступу до ресурсу: https://w3schoolsua.github.io/html/index.html

17. HTML і CSS довідник українською [Електронний ресурс] – Режим доступу до ресурсу: https://html-css.co.ua/

18. HTML Tutorial: Learn HTML For Free | Codecademy [Електронний ресурс] – Режим доступу до ресурсу: https://www.codecademy.com/learn/learn-html

19. HTML For Beginners The Easy Way: Start Learning HTML & CSS Today [Електронний ресурс] – Режим доступу до ресурсу: https://html.com/

20. Український веб-довідник [Електронний ресурс] – Режим доступу до ресурсу: https://css.in.ua/

ДОДАТОК А

«API ДОКУМЕНТАЦІЯ»

https://api.magazin-odyagu.com

```
[
  {
    "id": 1,
    "name": "Футболка 'Зручна Літнього Дня'",
    "category": "Футболки",
    "price": 20.0
  },
  // інші товари
]
```

```
{
  "items": [
    {"product_id": 1, "quantity": 2},
    {"product_id": 3, "quantity": 1}
  ],
  "total": 60.0,
  "user_id": 123
}
```

```
{
  "id": 101,
  "total": 60.0,
  "status": "В обробці"
}
```

```
[
  {
    "id": 101,
    "total": 60.0,
    "status": "В обробці"
  },
  // інші замовлення
]
```

```
{
  "error": "Опис помилки"
}
```