ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: **«Розробка front end сайту на основі JavaScript»**

на здобуття освітнього ступеня бакалавра
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

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

_____ Олександр ШКРАБА
(підпис) *(Ім'я, ПРІЗВИЩЕ здобувача)*

Виконав: здобувач вищої освіти гр. КНД-41
Олександр ШКРАБА
(Ім'я, ПРІЗВИЩЕ)

Керівник: Віктор ВИШНІВСЬКИЙ
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Рецензент: _____
Науковий ступінь, вчене звання *(Ім'я, ПРІЗВИЩЕ)*

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти «Бакалавр»

Спеціальність 122 «Комп'ютерні науки»

Освітньо-професійна програма «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедрою Комп'ютерних наук

_____ Віктор ВИШНІВСЬКИЙ
«_____» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Шкраба Олександр Геннадійович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Створення інтернет-магазину сайту на основі NodeJS (Frontend)
керівник кваліфікаційної роботи Віктор ВИШНІВСЬКИЙ д.т.н, професор
(Ім'я, ПРІЗВИЩЕ науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «27» 02.2024 р. №36
2. Строк подання кваліфікаційної роботи «31» травня 2024 р.
3. Вихідні дані до кваліфікаційної роботи: Node.js documentation, MDN Web Docs, Bootstrap 5 Documentation, PHP Documentation, MySQL Documentation, Font Awesome Documentation, Visual Studio Code, Stack Overflow, GitHub
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Аналіз сучасних технологій веб-розробки
    2. Розробка дизайну та інтерфейсу користувача
    3. Реалізація фронтенд частини з використанням Bootstrap 5 та JavaScript
    4. Тестування та оптимізація веб-додатку
5. Перелік графічного матеріалу:
    1. Мета роботи, об'єкт та предмет дослідження

2. Налаштування серверу
3. Опис основного функціоналу
4. Середовище програмування
5. Написання скелету
6. Первинні налаштування
7. Використання Font Awesome
8. Інтегрування Bootstrap 5
9. Адаптивне маштабування
10. Тестування адмінської частини
11. Тестування клієнтської частини
12. Кінцеві правки перед релізом
13. Висновки

6. Дата видачі завдання «23» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Літературний огляд сучасних технологій веб-розробки	25.02-22.03.24	Виконано
2	Формулювання проблеми та мети дослідження	23.03-30.03.24	Виконано
3	Аналіз інструментів та фреймворків для створення динамічних веб-інтерфейсів	31.03-09.04.24	Виконано
4	Розробка дизайну та проектування інтерфейсу користувача	10.04-14.04.24	Виконано
5	Реалізація фронтенд частини з використанням Bootstrap 5 та JavaScript	15.04-29.04.24	Виконано
6	Інтеграція фронтенду з бекендом на Node.js	30.04-05.05.24	Виконано
7	Тестування користувацького інтерфейсу	06.05-11.05.24	Виконано
8	Оптимізація швидкості завантаження сторінок	12.05-15.05.24	Виконано
9	Написання тексту дипломної роботи	16.05-20.05.24	Виконано
10	Редагування та оформлення дипломної роботи	21.05-25.05.24	Виконано

Здобувач вищої освіти _____ Олександр ШКРАБА
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Віктор ВИШНІВСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня бакалавра: 39 сторінок, 8 рисунків, 20 джерел.

Мета роботи – розробка та реалізація інтернет-магазину сайту з використанням Node.js, HTML, CSS, PHP, SQL, Bootstrap 5, Express.js, Font Awesome, JQuery, MySQL.

Об’єкт дослідження – процес створення веб-додатків, зокрема інтернет-магазинів, з використанням сучасних веб-технологій.

Предмет дослідження – інструменти та фреймворки для розробки веб-додатків, методи створення адаптивного дизайну, інтеграція фронтенду з бекендом, оптимізація продуктивності веб-додатків.

Короткий зміст роботи: у даній роботі досліджується та реалізується процес створення інтернет-магазину з використанням технологій Node.js, HTML, CSS, PHP, SQL, Bootstrap 5, Express.js, Font Awesome, JQuery, MySQL. Робота включає аналіз сучасних технологій веб-розробки, проектування дизайну та інтерфейсу користувача, розробку адаптивного дизайну з використанням Bootstrap 5 та JavaScript, інтеграцію фронтенду з бекендом на основі Node.js, тестування та оптимізацію веб-додатку.

У ході роботи детально розглядаються методи створення динамічних елементів за допомогою JavaScript, налаштування серверної частини з використанням Node.js та Express.js, управління даними за допомогою MySQL.

КЛЮЧОВІ СЛОВА: NODE.JS, BOOTSTRAP 5, JAVASCRIPT, PHP, SQL, MYSQL, ФРОНТЕНД, БЕКЕНД, ВЕБ-РОЗРОБКА, ІНТЕРНЕТ-МАГАЗИН

ABSTRACT

Text part of the qualification work for the bachelor's degree: 39 pages, 8 pictures, 20 sources.

The aim of the work – to develop and implement an online store website using Node.js, HTML, CSS, PHP, SQL, Bootstrap 5, Express.js, Font Awesome, JQuery, MySQL.

The object of research – the process of creating web applications, specifically online stores, using modern web technologies.

The subject of research – tools and frameworks for web application development, methods for creating responsive design, frontend-backend integration, and optimization of web application performance.

Summary of the work: this paper explores and implements the process of creating an online store using technologies such as Node.js, HTML, CSS, PHP, SQL, Bootstrap 5, Express.js, Font Awesome, JQuery, and MySQL. The work includes an analysis of modern web development technologies, the design and user interface development, the creation of responsive design using Bootstrap 5 and JavaScript, the integration of the frontend with the backend based on Node.js, and the testing and optimization of the web application.

The methods for creating dynamic elements using JavaScript, setting up the server-side with Node.js and Express.js, managing data with MySQL, as well as methods for optimizing page load speed and improving user experience are examined in detail.

KEYWORDS: NODE.JS, BOOTSTRAP 5, JAVASCRIPT, PHP, SQL, MYSQL, FRONTEND, BACKEND, WEB DEVELOPMENT, ONLINE STORE.

ОГЛАВЛЕНИЕ

ВСТУП

Актуальність теми

Сучасний світ неможливо уявити без інтернет-магазинів, які стали важливою складовою електронної комерції. Розробка зручного та ефективного веб-додатку для інтернет-магазину вимагає ретельного підходу як до фронтенду, так і до бекенду. Актуальність теми полягає в необхідності створення адаптивного та інтерактивного користувацького інтерфейсу, який забезпечує зручний доступ до функціональності магазину на будь-яких пристроях, а також інтеграції з бекендом для забезпечення динамічності та інтерактивності веб-додатку.

Мета роботи

Метою даної роботи є розробка та реалізація інтернет-магазину з акцентом на фронтенд частину, використовуючи сучасні технології веб-розробки. Основною задачею є створення адаптивного дизайну, інтерактивних елементів та забезпечення ефективної взаємодії між фронтендом та бекендом.

Завдання роботи

Для досягнення поставленої мети необхідно вирішити наступні завдання:

Провести аналіз сучасних технологій для розробки фронтенд частини веб-додатків.

Вибрати оптимальні інструменти та фреймворки для створення динамічних веб-інтерфейсів.

Розробити дизайн та інтерфейс користувача з використанням принципів UX/UI дизайну.

Реалізувати адаптивний дизайн за допомогою Bootstrap 5.

Створити інтерактивні елементи за допомогою JavaScript.

Забезпечити інтеграцію фронтенду з бекендом на основі Node.js.

Провести тестування та оптимізацію веб-додатку для забезпечення його стабільної роботи та високої продуктивності.

Об’єкт дослідження

Об’єктом дослідження є процес розробки інтернет-магазину, зосереджуючись на фронтенд частині та частково на бекенді.

Предмет дослідження

Предметом дослідження є технології та методи, що використовуються для розробки веб-інтерфейсів, зокрема Bootstrap 5, JavaScript, а також інтеграція з бекендом на Node.js.

Методи дослідження

У роботі використовуються такі методи дослідження:

Аналіз літературних джерел та існуючих рішень у сфері веб-розробки.

Проектування користувацького інтерфейсу та розробка дизайну.

Практична розробка фронтенд та частково бекенд частин додатку.

Тестування та оптимізація веб-додатку для забезпечення його коректної роботи.

Структура роботи

Робота складається з вступу, чотирьох розділів, висновків, переліку посилань та демонстраційних матеріалів. У першому розділі представлено аналіз сучасних технологій веб-розробки та вибір оптимальних інструментів для реалізації проекту. Другий розділ присвячено розробці дизайну та інтерфейсу користувача. Третій розділ описує процес реалізації фронтенд частини з використанням Bootstrap 5 та JavaScript, а також інтеграцію з бекендом на Node.js. У четвертому розділі розглянуто тестування та оптимізацію веб-додатку. У висновках підведено підсумки роботи та окреслено перспективи подальших досліджень.

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ ВЕБ-РОЗРОБКИ

1.1 Вибір технологій для розробки фронтенд частини

У сучасній веб-розробці існує безліч технологій та інструментів, що дозволяють створювати високоякісні та зручні для користувача інтерфейси. Основними критеріями вибору технологій для розробки фронтенд частини інтернет-магазину були продуктивність, гнучкість, зручність використання та підтримка адаптивного дизайну.

HTML5 та CSS3 – базові технології для створення структури та стилізації веб-сторінок. HTML5 дозволяє використовувати нові семантичні теги та атрибути, що покращують доступність та SEO оптимізацію. CSS3 надає розширені можливості для стилізації, включаючи анімації та трансформації (рис. 1.1, 1.2).

```html
<!DOCTYPE html>
<html lang="uk">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Приклад HTML5 і CSS3</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <header>
        <h1>Ласкаво просимо на мій сайт</h1>
    </header>

    <main>
        <section>
            <h2>Про мене</h2>
            <p>Привіт! Я веб-розробник, який любить створювати красиві та функціональні
            <img src="https://via.placeholder.com/150" alt="Моє зображення">
        </section>

        <section>
            <h2>Контактна інформація</h2>
```

Рисунок 1.1 - Приклад HTML коду

```css
/* Скидання стилів за замовчуванням */
* {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
}

/* Основні стилі */
body {
    font-family: Arial, sans-serif;
    line-height: 1.6;
    background-color: #f4f4f4;
    color: #333;
    padding: 20px;
}

/* Стилі для заголовка */
header {
    background-color: #333;
    color: #fff;
    padding: 10px 0;
    text-align: center;
    margin-bottom: 20px;
}
```

Рисунок 1.2 - Приклад CSS3 коду

Bootstrap 5 – популярний фреймворк для створення адаптивного дизайну. Bootstrap 5 надає набір готових компонентів та утиліт, що дозволяють швидко та ефективно розробляти інтерфейси, які коректно відображаються на різних пристроях (рис. 1.3).

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Пример Bootstrap 5</title>
    <!-- Подключение CSS Bootstrap 5 -->
    <link href="https://stackpath.bootstrapcdn.com/bootstrap/5.1.3/css/bootstrap.min.css"
</head>
<body>
    <!-- Навигационная панель -->
    <nav class="navbar navbar-expand-lg navbar-light bg-light">
        <div class="container-fluid">
            <a class="navbar-brand" href="#">Мой сайт</a>
            <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs
                <span class="navbar-toggler-icon"></span>
            </button>
```

Рисунок 1.3 - Приклад підключення Bootstrap 5 і використання для створення навігаційної панелі

JavaScript – мова програмування, що використовується для створення інтерактивних та динамічних елементів на веб-сторінках. JavaScript дозволяє реалізувати обробку подій, динамічну зміну контенту та взаємодію з сервером без перезавантаження сторінки.

1.2 Інструменти та фреймворки для створення динамічних веб-інтерфейсів

Для створення динамічних веб-інтерфейсів використовуються різноманітні інструменти та фреймворки. Основними з них є:

React – JavaScript-бібліотека для створення користувацьких інтерфейсів, розроблена компанією Facebook. React дозволяє будувати компонентну архітектуру, що полегшує розробку та підтримку великих проектів. Він надає можливість

створення багаторазових компонентів, що спрощує процес розробки і підвищує продуктивність.

Vue.js – прогресивний фреймворк для створення користувацьких інтерфейсів. Vue.js має простий та інтуїтивно зрозумілий синтаксис, що робить його зручним для використання навіть новачками. Vue.js також забезпечує двосторонню прив'язку даних, що полегшує синхронізацію між моделлю та представленням.

Angular – потужний фреймворк для створення динамічних веб-додатків, розроблений компанією Google. Angular надає набір інструментів для створення масштабованих та продуктивних додатків, включаючи двосторонню прив'язку даних, систему залежностей, та інтеграцію з іншими сервісами. Angular також підтримує TypeScript, що дозволяє розробникам писати більш структурований та надійний код.

Після аналізу різних технологій було обрано комбінацію HTML5, CSS3, Bootstrap 5 та JavaScript для реалізації фронтенд частини інтернет-магазину. Це забезпечує необхідну гнучкість, продуктивність та зручність у використанні.

1.3 Переваги вибраних технологій

HTML5 та CSS3:

HTML5 дозволяє використовувати нові семантичні теги, що покращують доступність і SEO оптимізацію.

CSS3 надає розширені можливості для стилізації, включаючи анімації, трансформації та медіазапити для адаптивного дизайну.

Bootstrap 5:

Надає набір готових компонентів та утиліт, що дозволяють швидко розробляти адаптивні інтерфейси.

Підтримує сучасні стандарти веб-розробки і забезпечує коректне відображення на різних пристроях.

JavaScript:

Забезпечує можливість створення інтерактивних та динамічних елементів на веб-сторінках.

Дозволяє реалізувати обробку подій, динамічну зміну контенту та взаємодію з сервером без перезавантаження сторінки.

React, Vue.js, Angular:

Кожен з цих фреймворків має свої переваги та особливості, що дозволяють вибрати оптимальний інструмент для конкретного проекту.

Всі вони підтримують компонентну архітектуру, що полегшує розробку та підтримку великих проектів.

Вибрані технології забезпечують високу продуктивність, гнучкість та зручність у використанні, що є ключовими факторами для успішної реалізації інтернет-магазину (рис. 1.4).

```
// Знаходимо елементи на сторінці
const messageElement = document.getElementById('message');
const changeTextButton = document.getElementById('changeTextButton');

// Додаємо обробник події кліку на кнопку
changeTextButton.addEventListener('click', function() {
    // Змінюємо текст елемента
    messageElement.textContent = 'Текст змінено! Дякую, що натиснули кнопку.';

    // Відображаємо повідомлення
    alert('Текст успішно змінено!');
});
```

Рисунок 1.4 - Приклад JavaScript коду, за допомогою якого можна виводити всплиаючі повідомлення

2 РОЗРОБКА ДИЗАЙНУ ТА ІНТЕРФЕЙСУ КОРИСТУВАЧА

2.1 Проектування користувацького інтерфейсу

Проектування користувацького інтерфейсу (UI) є ключовим етапом у розробці веб-додатків. Основними завданнями на цьому етапі були створення wireframes та прототипів, а також врахування принципів UX/UI дизайну для забезпечення зручності та інтуїтивності використання.

Wireframes – схематичні зображення майбутнього інтерфейсу, що показують розташування основних елементів та їх взаємодію. Wireframes дозволяють швидко оцінити логіку та структуру інтерфейсу без відволікання на деталі дизайну. На цьому етапі визначаються основні компоненти сторінки, їх розміщення та взаємодія між ними.

Прототипи – більш детальні моделі інтерфейсу, що включають кольори, шрифти та інші стилістичні елементи. Прототипи допомагають побачити, як виглядатиме кінцевий продукт, та виявити можливі проблеми на ранніх етапах розробки. Вони дозволяють провести попереднє тестування з користувачами та отримати зворотний зв'язок.

Принципи UX/UI дизайну – підхід до проектування, що фокусується на зручності та задоволенні користувачів. Врахування принципів UX/UI дизайну дозволяє створити інтерфейс, який є інтуїтивно зрозумілим та легким у використанні. Це включає дослідження потреб користувачів, розробку користувацьких сценаріїв та забезпечення логічної навігації по сайту (рис. 2.1, 2.2).

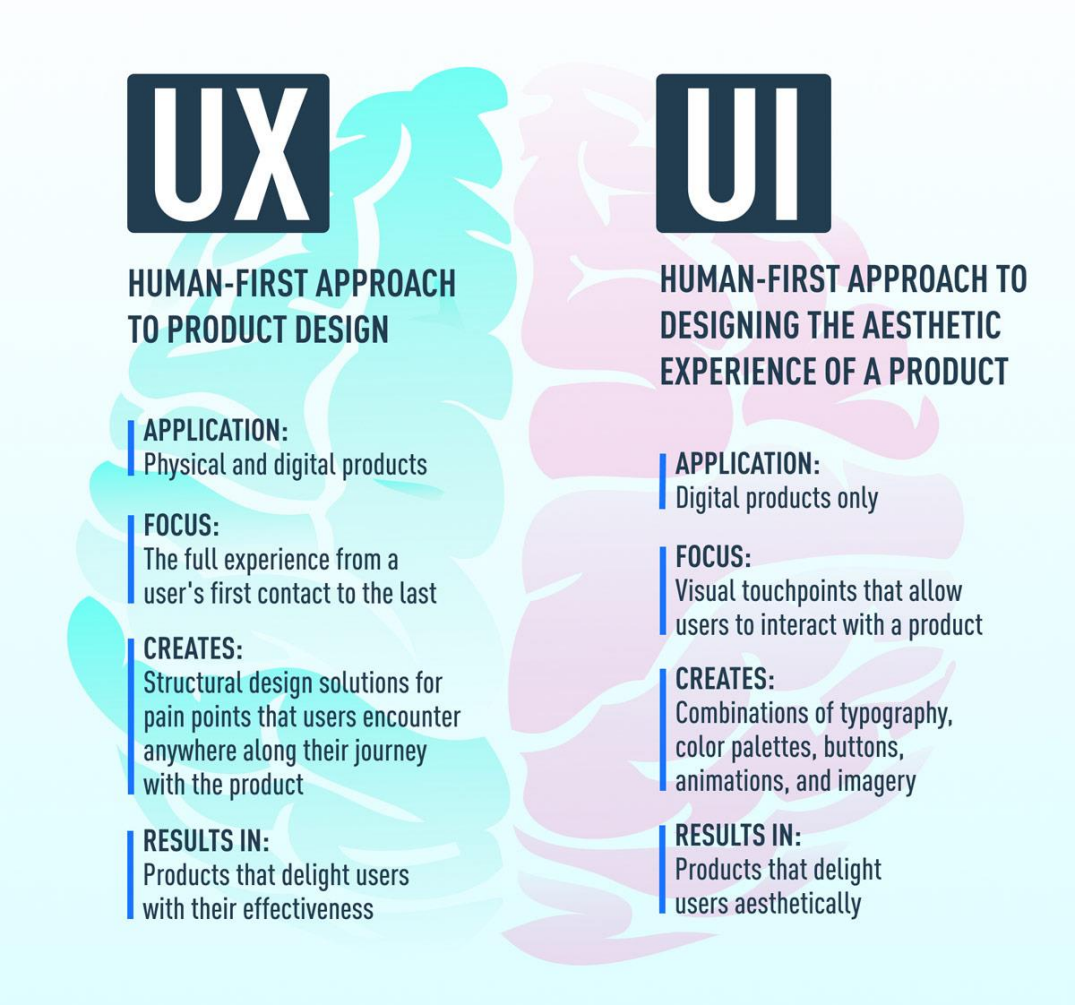


Рисунок 2.1 - Визначення UX/UI дизайну

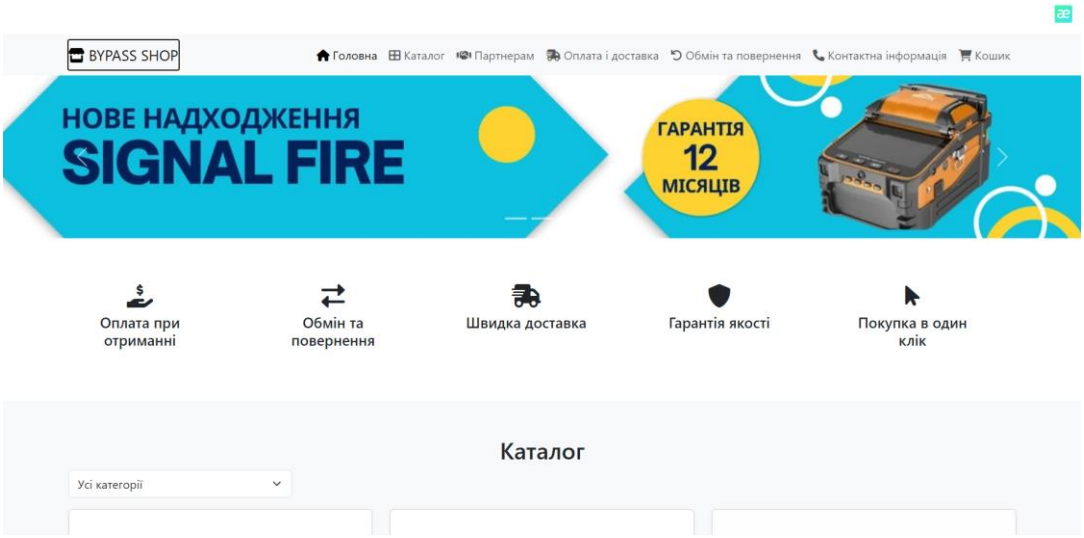


Рисунок 2.2 - Приклад дизайну сайту

2.2 Вибір кольорової гами та стилів

Вибір кольорової гами та стилів є важливим аспектом у розробці дизайну веб-додатка. Для інтернет-магазину було обрано сучасну та привабливу кольорову гаму, що забезпечує комфортне сприйняття контенту користувачами.

Основні кольори – визначення основних кольорів для інтерфейсу, що будуть використовуватися у всіх його елементах. Це можуть бути кольори бренду або нейтральні тони, що підкреслюють основний контент. Використання обмеженої кількості основних кольорів допомагає створити гармонійний та впізнаваний дизайн.

Типографія – вибір шрифтів та налаштування відступів для забезпечення зручного читання тексту. Використання різних типів шрифтів для заголовків, підзаголовків та основного тексту дозволяє покращити структуру та читабельність контенту. Важливо вибрати шрифти, що добре поєднуються між собою та підходять для читання на різних пристроях.

Стилі елементів – налаштування стилів для кнопок, форм, таблиць та інших елементів інтерфейсу. Використання єдиних стилів для всіх елементів дозволяє створити гармонійний та узгоджений дизайн. Кнопки повинні бути достатньо великими для натискання, а форми – зручними для заповнення.

2.3 Використання іконок та графічних елементів

Іконки та графічні елементи відіграють важливу роль у створенні привабливого та інтуїтивно зрозумілого інтерфейсу. Для інтернет-магазину було обрано набір іконок, що доповнюють дизайн та покращують взаємодію користувачів з додатком.

Вибір іконок – використання готових наборів іконок або створення власних для забезпечення унікальності дизайну. Іконки повинні бути зрозумілими та відповідати стилю додатка. Використання іконок у навігаційних панелях, кнопках та формах допомагає користувачам швидше орієнтуватися на сайті.

Графічні елементи – використання зображень, ілюстрацій та інших графічних елементів для покращення дизайну. Це можуть бути банери, фонові зображення або декоративні елементи, що підкреслюють основний контент. Важливо враховувати баланс між текстом та графікою, щоб дизайн не був перевантажений.

2.4 Адаптивний дизайн

Адаптивний дизайн є ключовим аспектом сучасного веб-дизайну, оскільки користувачі можуть заходити на сайт з різних пристроїв. Використання адаптивного дизайну дозволяє забезпечити коректне відображення контенту на всіх типах пристроїв – від мобільних телефонів до настільних комп'ютерів (рис. 2.3).

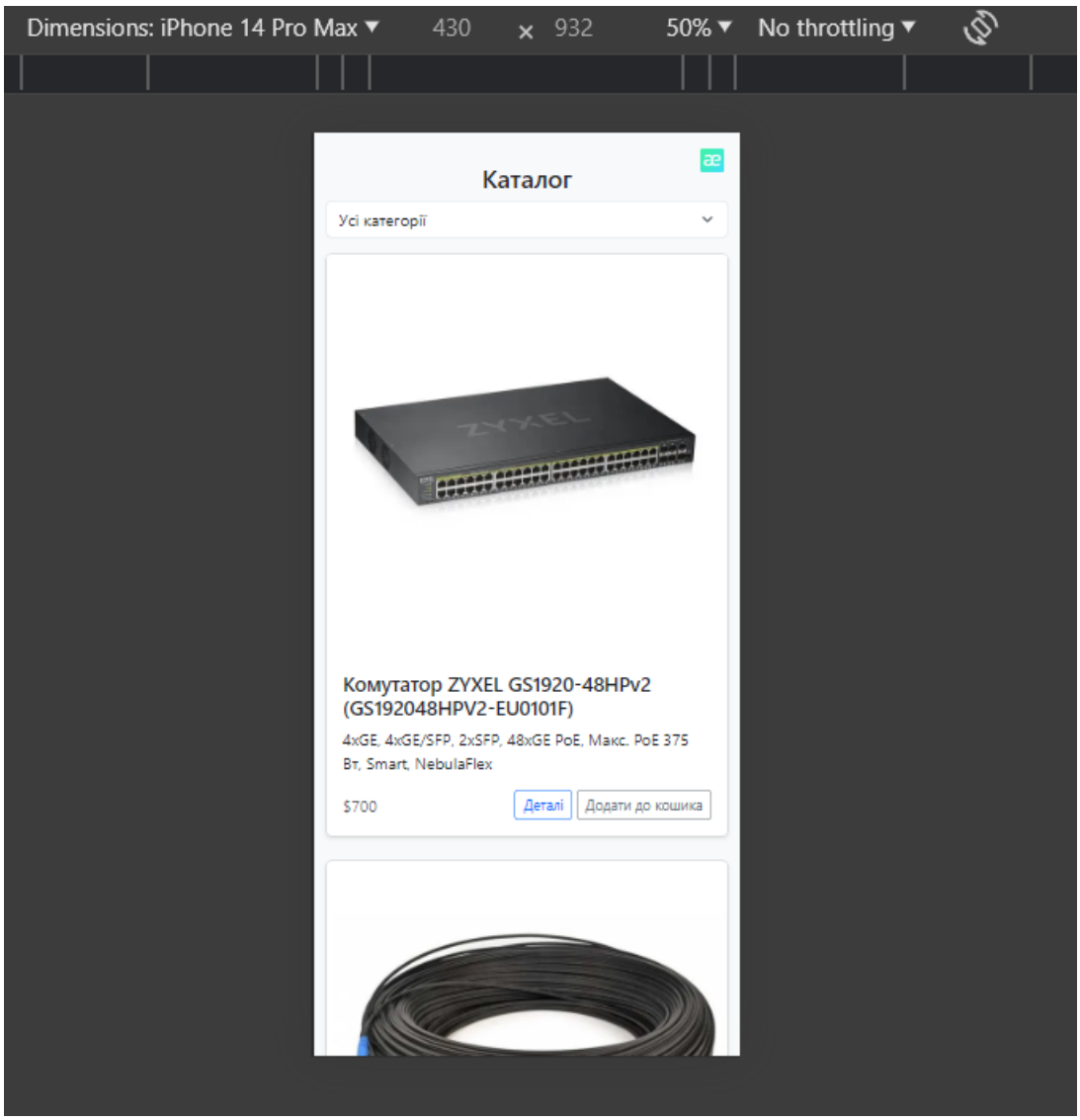


Рисунок 2.3 - Приклад адаптивності з мого сайту, за допомогою Bootstrap 5

Використання медіазапитів – налаштування стилів для різних розмірів екрану за допомогою CSS медіазапитів. Це дозволяє адаптувати дизайн під будь-який розмір екрану, змінюючи розташування та розмір елементів в залежності від пристрою.

Гнучкі сітки – використання гнучких сіток (flexbox, grid) для розташування елементів на сторінці. Гнучкі сітки дозволяють легко змінювати структуру сторінки в залежності від розміру екрану, забезпечуючи адаптивність та зручність користування.

Респонсивні зображення – використання респонсивних зображень, що автоматично підлаштовуються під розмір екрану. Це дозволяє зменшити час завантаження сторінки та покращити користувацький досвід.

2.5 Тестування дизайну

Тестування дизайну є важливим етапом у розробці веб-додатка. Проведення тестування дозволяє виявити та виправити помилки, забезпечити коректну роботу всіх елементів інтерфейсу та підвищити зручність використання додатка.

Юзабіліті тестування – проведення тестування з реальними користувачами для оцінки зручності використання інтерфейсу. Отримання зворотного зв'язку від користувачів дозволяє виявити можливі проблеми та покращити дизайн.

Тестування на різних пристроях – перевірка роботи додатка на різних типах пристроїв (мобільних телефонах, планшетах, настільних комп'ютерах). Це дозволяє забезпечити коректне відображення та функціонування інтерфейсу на всіх пристроях.

А/Б тестування – проведення А/Б тестування для порівняння різних варіантів дизайну та вибору найкращого з них. Це дозволяє покращити користувацький досвід та підвищити ефективність інтерфейсу.

3 РЕАЛІЗАЦІЯ ФРОНТЕНД ЧАСТИНИ З ВИКОРИСТАННЯМ BOOTSTRAP 5 ТА JAVASCRIPT

3.1 Створення адаптивного дизайну з використанням Bootstrap 5

Bootstrap 5 є одним з найпопулярніших фреймворків для створення адаптивних веб-додатків. Використання Bootstrap дозволяє швидко та ефективно розробляти інтерфейси, що коректно відображаються на різних пристроях.

Компоненти Bootstrap – використання готових компонентів для створення навігаційних панелей, форм, карток та інших елементів інтерфейсу. Це дозволяє зекономити час та забезпечити єдиний стиль для всіх елементів (рис. 3.1).

Наприклад, створення навігаційної панелі з використанням Bootstrap:
html

```
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <a class="navbar-brand" href="#">Магазин</a>
  <button class="navbar-toggler" type="button" data-toggle="collapse" data-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
    <span class="navbar-toggler-icon"></span>
  </button>
  <div class="collapse navbar-collapse" id="navbarNav">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" href="#">Головна</a>
      </li>
```

```
    <li class="nav-item">
      <a class="nav-link" href="#">Каталог</a>
    </li>
    <li class="nav-item">
      <a class="nav-link" href="#">Контакти</a>
    </li>
  </ul>
 </div>
</nav>
```

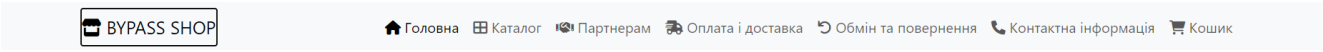


Рисунок 3.1 - Панель створена за допомогою Bootstrap 5

Grid-система – налаштування grid-системи для розташування елементів на сторінці. Використання grid-системи дозволяє створювати складні макети, що автоматично адаптуються до різних розмірів екрану.

Приклад використання grid-системи Bootstrap для створення макету сторінки:
html

```
<div class="container">
 <div class="row">
  <div class="col-md-4">Контент 1</div>
  <div class="col-md-4">Контент 2</div>
  <div class="col-md-4">Контент 3</div>
 </div>
</div>
```

Адаптивні зображення та відео – використання класів Bootstrap для забезпечення адаптивності зображень та відео.

Приклад використання класу img-fluid для зображень:

html

<img src="image.jpg" class="img-fluid" alt="Приклад зображення">

3.2 Реалізація динамічних елементів за допомогою JavaScript

JavaScript є потужним інструментом для створення інтерактивних та динамічних елементів на веб-сторінках. Використання JavaScript дозволяє додавати обробку подій, динамічну зміну контенту та взаємодію з сервером без перезавантаження сторінки.

Обробка подій – налаштування обробників подій для кнопок, форм та інших елементів інтерфейсу. Це дозволяє реалізувати інтерактивну взаємодію користувачів з додатком.

Приклад додавання обробника подій для кнопки:
javascript

```
document.getElementById('myButton').addEventListener('click', function() {
  alert('Кнопка натиснута!');
});
```

Динамічна зміна контенту – використання JavaScript для динамічної зміни контенту на сторінці. Це може бути додавання нових елементів, зміна стилів або завантаження даних з сервера.

Приклад динамічного додавання елементів до сторінки:
javascript

```
document.getElementById('addButton').addEventListener('click', function() {
```

```
  const newElement = document.createElement('div');
  newElement.textContent = 'Новий елемент';
  document.getElementById('content').appendChild(newElement);
});
```

Взаємодія з сервером за допомогою AJAX – використання AJAX для відправлення запитів та отримання даних з сервера без перезавантаження сторінки.

Приклад використання Fetch API для відправлення запиту:
javascript

```
fetch('/api/data')
  .then(response => response.json())
  .then(data => {
    document.getElementById('dataContainer').textContent = JSON.stringify(data);
  })
  .catch(error => console.error('Помилка:', error));
```

3.3 Інтеграція фронтенду з бекендом на Node.js

Інтеграція фронтенду з бекендом є важливим етапом у розробці веб-додатків. Використання Node.js для реалізації бекенд частини дозволяє забезпечити швидку обробку запитів та ефективну взаємодію з базою даних.

Налаштування взаємодії між фронтендом і бекендом – налаштування маршрутизації та обробки запитів для забезпечення коректної взаємодії між фронтендом і бекендом.

Приклад налаштування маршрутизації на Node.js:
javascript

```
const express = require('express');
const app = express();
app.use(express.json());

app.get('/api/products', (req, res) => {
  // Отримання даних з бази даних та відправка їх на фронтенд
  res.json(products);
});

app.post('/api/order', (req, res) => {
  // Обробка замовлення та збереження його в базі даних
  const order = req.body;
  // Збереження замовлення...
  res.json({ message: 'Замовлення прийнято!' });
});

app.listen(3000, () => {
  console.log('Сервер працює на порту 3000');
});
```

Обробка запитів та отримання даних з сервера – використання AJAX для відправлення запитів та отримання даних з сервера без перезавантаження сторінки.

Приклад використання Fetch API для відправлення замовлення:

javascript

```
document.getElementById('orderForm').addEventListener('submit', function(event) {
  event.preventDefault();
  const orderData = {
```

```
    name: document.getElementById('name').value,
    email: document.getElementById('email').value,
    phone: document.getElementById('phone').value,
    items: cartItems // Масив з товарами у кошику
  };
  fetch('/api/order', {
    method: 'POST',
    headers: {
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(orderData)
  })
  .then(response => response.json())
  .then(data => {
    alert(data.message);
  })
  .catch(error => console.error('Помилка:', error));
});
```

4 ТЕСТУВАННЯ ТА ОПТИМІЗАЦІЯ ВЕБ-ДОДАТКУ

4.1 Тестування користувацького інтерфейсу

Тестування користувацького інтерфейсу (UI) є важливим етапом у розробці веб-додатків. Проведення тестування дозволяє виявити та виправити помилки, забезпечити коректну роботу всіх елементів інтерфейсу та підвищити зручність використання додатка.

Юзабіліті тестування – проведення тестування з реальними користувачами для оцінки зручності використання інтерфейсу. Отримання зворотного зв'язку від користувачів дозволяє виявити можливі проблеми та покращити дизайн.

Методика юзабіліті тестування включає проведення інтерв'ю з користувачами, спостереження за їх взаємодією з додатком та аналіз результатів.
Тестування на різних пристроях – перевірка роботи додатка на різних типах пристроїв (мобільних телефонах, планшетах, настільних комп'ютерах). Це дозволяє забезпечити коректне відображення та функціонування інтерфейсу на всіх пристроях.

Використання емуляторів та реальних пристроїв для тестування адаптивності та респонсивності дизайну.
А/Б тестування – проведення А/Б тестування для порівняння різних варіантів дизайну та вибору найкращого з них. Це дозволяє покращити користувацький досвід та підвищити ефективність інтерфейсу.

Створення двох версій інтерфейсу (A і B) та аналіз результатів взаємодії користувачів з кожною версією (рис. 4.1).

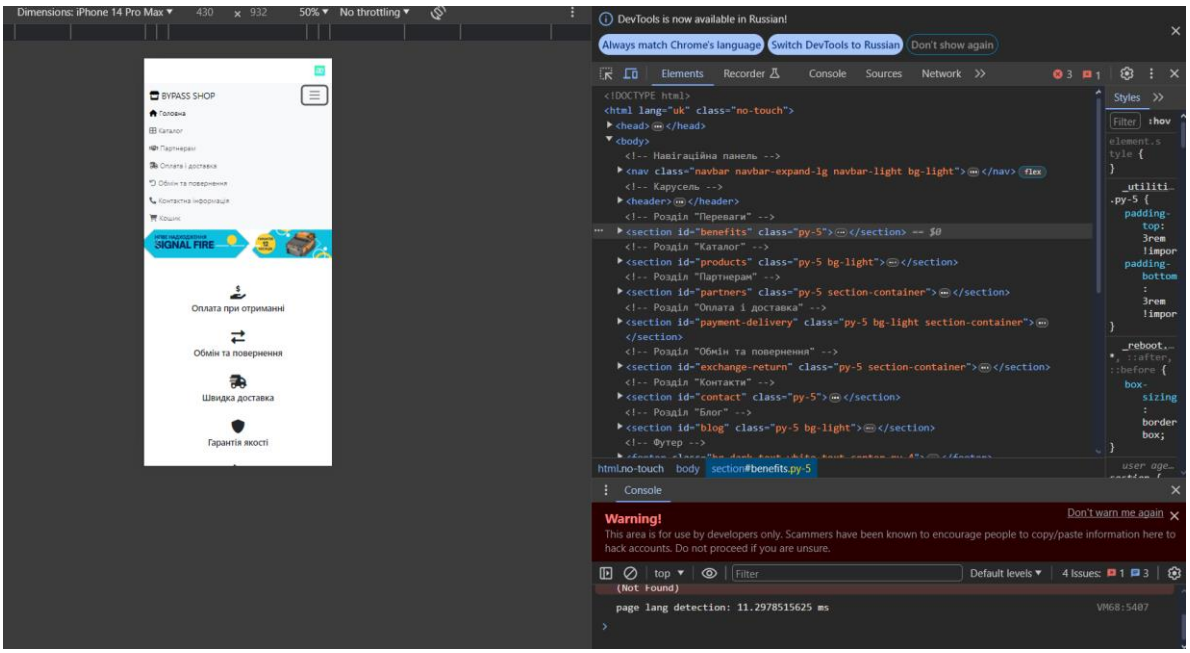


Рисунок 4.1 - Приклад як можна перевіряти адаптивність сайту

4.2 Оптимізація швидкості завантаження сторінок

Оптимізація швидкості завантаження сторінок є важливим аспектом у забезпеченні зручності використання веб-додатка. Швидке завантаження сторінок підвищує задоволеність користувачів та покращує SEO показники.

Оптимізація зображень – зменшення розміру зображень без втрати якості. Використання форматів зображень, що забезпечують кращу компресію та швидше завантаження.

Використання інструментів для стиснення зображень, таких як TinyPNG або ImageOptim.

Оптимізація ресурсів – зменшення розміру CSS, JavaScript та інших ресурсів за допомогою методів minification та compression. Використання кешування для зменшення часу завантаження сторінок.

Використання інструментів для мінімізації та компресії ресурсів, таких як UglifyJS для JavaScript та CSSNano для CSS.

Використання методів для прискорення завантаження – використання методів lazy loading для відкладеного завантаження зображень та інших ресурсів, що не відображаються одразу. Це дозволяє зменшити час завантаження початкової сторінки та покращити користувацький досвід.

Приклад використання lazy loading для зображень:

html

<img src="image.jpg" loading="lazy" alt="Приклад зображення">

Оптимізація кодів – уникання зайвих обчислень та викликів функцій, використання асинхронних операцій для покращення продуктивності.

Оптимізація JavaScript коду шляхом видалення зайвих обчислень та використання async/await для асинхронних операцій.

Зменшення кількості HTTP-запитів – об'єднання CSS та JavaScript файлів для зменшення кількості HTTP-запитів.

Використання інструментів для об'єднання файлів, таких як Webpack.

4.3 Інструменти для тестування та оптимізації

Для тестування та оптимізації веб-додатка використовуються різноманітні інструменти, що дозволяють автоматизувати процеси та отримати детальну інформацію про продуктивність додатка.

Google Lighthouse – інструмент для аналізу продуктивності, доступності та SEO веб-додатка. Lighthouse надає детальні рекомендації щодо покращення продуктивності та зручності використання додатка.

Використання Lighthouse для аналізу продуктивності сторінок та отримання рекомендацій щодо оптимізації.

WebPageTest – інструмент для тестування швидкості завантаження веб-сторінок. WebPageTest дозволяє отримати детальну інформацію про час завантаження кожного елементу сторінки та виявити можливі проблеми.

Використання WebPageTest для аналізу швидкості завантаження сторінок та отримання рекомендацій щодо оптимізації.

Browser Developer Tools – інструменти розробника в браузерах (Chrome DevTools, Firefox Developer Tools) для аналізу продуктивності, відладки та оптимізації веб-додатка.

Використання Developer Tools для аналізу продуктивності JavaScript коду, CSS та ресурсів сторінки.

ВИСНОВКИ

У ході виконання дипломної роботи на тему "Створення інтернет-магазину на основі Node.js з акцентом на фронтенд" було досягнуто низку важливих результатів, які підтверджують актуальність і значущість даного дослідження. Проведена робота охоплює всі етапи створення сучасного веб-додатка, починаючи від аналізу технологій і закінчуючи інтеграцією з бекендом та оптимізацією продуктивності.

Аналіз сучасних технологій веб-розробки

Було проведено детальний аналіз сучасних технологій та інструментів для створення веб-інтерфейсів. Вибір HTML5, CSS3, Bootstrap 5, JavaScript, а також фреймворків React, Vue.js та Angular був обґрунтований їх перевагами в продуктивності, гнучкості та зручності використання. HTML5 та CSS3 дозволяють створювати структуровані та привабливі веб-сторінки, які відповідають сучасним стандартам. Bootstrap 5 забезпечує швидку розробку адаптивного дизайну, що коректно відображається на різних пристроях. JavaScript, у свою чергу, додає інтерактивності та динаміки до веб-сторінок, що підвищує зручність використання додатка.

Розробка дизайну та інтерфейсу користувача

Створення адаптивного дизайну є ключовим аспектом розробки сучасних веб-додатків. Використання wireframes та прототипів дозволило розробити інтуїтивно зрозумілий та зручний інтерфейс, що відповідає сучасним стандартам UX/UI дизайну. Вибір кольорової гами та стилів, а також використання іконок та графічних елементів сприяли створенню привабливого та гармонійного дизайну.

Адаптивний дизайн забезпечує коректне відображення контенту на всіх типах пристроїв, що підвищує зручність використання веб-додатка для користувачів.

Реалізація фронтенд частини з використанням Bootstrap 5 та JavaScript

Було реалізовано адаптивний дизайн з використанням компонентів Bootstrap 5 та налаштування grid-системи. Використання готових компонентів Bootstrap дозволило швидко розробити інтерфейс, що коректно відображається на різних пристроях. JavaScript був використаний для створення інтерактивних та динамічних елементів, що покращують взаємодію користувачів з додатком. Додавання обробки подій, динамічної зміни контенту та взаємодії з сервером без перезавантаження сторінки підвищило ефективність та зручність використання додатка.

Інтеграція фронтенду з бекендом на Node.js

Інтеграція фронтенду з бекендом є важливим етапом у розробці веб-додатків. Використання Node.js для реалізації бекенд частини дозволило забезпечити швидку обробку запитів та ефективну взаємодію з базою даних. Було налаштовано маршрутизацію та обробку запитів, що забезпечує коректну взаємодію між фронтендом і бекендом. Використання AJAX дозволило забезпечити динамічне завантаження даних без перезавантаження сторінки, що покращило продуктивність та зручність використання додатка.

Тестування та оптимізація веб-додатка

Проведено детальне тестування всіх компонентів системи для виявлення та виправлення помилок. Було проведено юзабіліті тестування з реальними користувачами, тестування на різних пристроях та А/Б тестування. Це дозволило виявити можливі проблеми та покращити дизайн та функціональність додатка.

Оптимізація зображень, ресурсів та використання методів lazy loading дозволила покращити швидкість завантаження сторінок та підвищити продуктивність додатка. Використання інструментів для аналізу продуктивності, таких як Google Lighthouse, WebPageTest та Browser Developer Tools, дозволило отримати детальну інформацію про продуктивність додатка та впровадити необхідні покращення.

Практична цінність роботи

Розроблений інтернет-магазин може бути використаний як основа для створення комерційних веб-додатків. Отримані знання та досвід можуть бути застосовані для розробки інших проектів у сфері веб-розробки та електронної комерції. Використання сучасних технологій та методів дозволяє створювати ефективні та надійні системи, які відповідають вимогам ринку та забезпечують високий рівень обслуговування клієнтів. Створений інтернет-магазин має високу продуктивність, надійність та зручність у використанні, що робить його конкурентоспроможним на ринку електронної комерції.

Перспективи подальших досліджень

Подальші дослідження можуть бути спрямовані на впровадження додаткових функціональних можливостей, таких як система рекомендацій товарів на основі аналізу поведінки користувачів, інтеграція з іншими платіжними системами, покращення безпеки даних користувачів та оптимізація продуктивності системи. Розробка мобільної версії інтернет-магазину або додатка також може стати перспективним напрямом розвитку проекту. Дослідження нових технологій та інструментів для веб-розробки дозволить постійно покращувати функціональність

та продуктивність веб-додатків, забезпечуючи їх відповідність сучасним вимогам та очікуванням користувачів.

Загальні висновки

В результаті виконання роботи було створено повноцінний інтернет-магазин, який відповідає сучасним вимогам та стандартам веб-розробки. Впроваджені технології та методи дозволяють забезпечити високу продуктивність, надійність та зручність у використанні системи. Отримані результати можуть бути використані для подальшого вдосконалення та розвитку проекту, а також для реалізації подібних проектів у сфері електронної комерції. Виконана робота демонструє можливості сучасних технологій веб-розробки та їх застосування для створення інтернет-магазину. Досягнуті результати підтверджують доцільність вибору обраних технологій та методів, а також їх ефективність у реалізації поставлених завдань.

ПЕРЕЛІК ПОСИЛАНЬ

Node.js Documentation – Офіційна документація Node.js. Доступно за адресою: https://nodejs.org/en/docs/

MySQL Documentation – Офіційна документація MySQL. Доступно за адресою: https://dev.mysql.com/doc/

PHP Documentation – Офіційна документація PHP. Доступно за адресою: https://www.php.net/docs.php

Bootstrap Documentation – Офіційна документація Bootstrap 5. Доступно за адресою: https://getbootstrap.com/docs/5.0/getting-started/introduction/

Sequelize Documentation – Офіційна документація Sequelize. Доступно за адресою: https://sequelize.org/master/

Express.js Documentation – Офіційна документація Express.js. Доступно за адресою: https://expressjs.com/en/starter/installing.html

Telegram Bot API – Офіційна документація Telegram Bot API. Доступно за адресою: https://core.telegram.org/bots/api

Git Documentation – Офіційна документація Git. Доступно за адресою: https://git-scm.com/doc

GitHub Documentation – Офіційна документація GitHub. Доступно за адресою: https://docs.github.com/en

Visual Studio Code Documentation – Офіційна документація Visual Studio Code. Доступно за адресою: https://code.visualstudio.com/docs

Google Lighthouse – Інструмент для аналізу продуктивності, доступності та SEO веб-додатка. Доступно за адресою: https://developers.google.com/web/tools/lighthouse

WebPageTest – Інструмент для тестування швидкості завантаження веб-сторінок. Доступно за адресою: https://www.webpagetest.org/

Browser Developer Tools – Інструменти розробника в браузерах (Chrome DevTools, Firefox Developer Tools) для аналізу продуктивності, відладки та оптимізації веб-додатка. Інформація доступна в самих браузерах.

Selenium Documentation – Офіційна документація Selenium. Доступно за адресою: https://www.selenium.dev/documentation/en/

Cypress Documentation – Офіційна документація Cypress. Доступно за адресою: https://docs.cypress.io/

Webpack Documentation – Офіційна документація Webpack. Доступно за адресою: https://webpack.js.org/concepts/

TinyPNG – Інструмент для стиснення зображень. Доступно за адресою: https://tinypng.com/

ImageOptim – Інструмент для стиснення зображень. Доступно за адресою: https://imageoptim.com/mac

UglifyJS – Інструмент для мінімізації JavaScript. Доступно за адресою: https://github.com/mishoo/UglifyJS

CSSNano – Інструмент для мінімізації CSS. Доступно за адресою: https://cssnano.co/

Розробка Front End сайту на основі JavaScript

Виконав: Студент Шкраба О.Г.

Керівник: Вишнівський В.В.

Мета роботи, Об’єкт та Предмет дослідження

- **Мета роботи:** Розробка та реалізація фронтенд частини сучасного веб-сайту з використанням JavaScript та суміжних технологій.
- **Об’єкт дослідження:** Процес створення веб-додатків з використанням сучасних веб-технологій.
- **Предмет дослідження:** JavaScript та пов'язані з ним технології для розробки фронтенд частини веб-додатків, методи створення адаптивного дизайну, оптимізація продуктивності веб-додатків.

Актуальність роботи

- **Застосування сучасних технологій веб-розробки** дозволяє створювати високопродуктивні, масштабовані та безпечні веб-сайти. JavaScript є основним інструментом для розробки інтерактивних та динамічних веб-сторінок, що робить його незамінним у сучасному веб-дизайні.

Вибір технологій

- - **JavaScript:** Основна мова для створення інтерактивних веб-сторінок.
- - **HTML5 та CSS3:** Основи для структури та стилізації веб-сторінок.
- - **Bootstrap 5:** Фреймворк для створення адаптивного дизайну.
- - **JQuery:** Бібліотека для спрощення роботи з JavaScript.

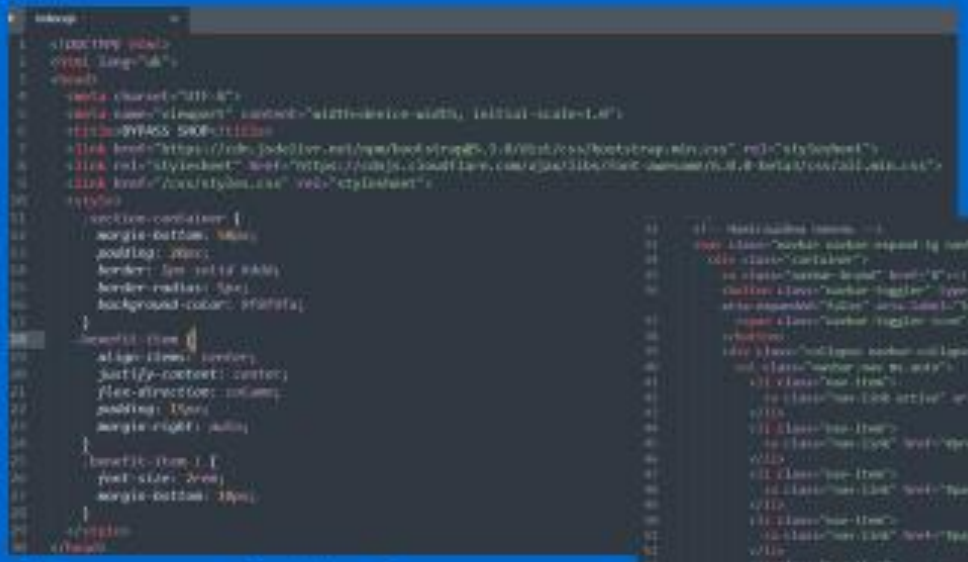
Розробка дизайну та інтерфейсу

- **1. Проектування користувацького інтерфейсу:**
- - Створення макетів та прототипів.
- **2. Вибір кольорової гами та стилів:**
- - Використання стандартів UI/UX.
- **3. Використання іконок та графічних елементів:**
- - Font Awesome для іконок.
- **4. Адаптивний дизайн:**
- - Bootstrap 5 для адаптивності.

Реалізація фронтенд частини

- **1. Створення адаптивного дизайну:**
- - Використання компонентів Bootstrap 5.

Реалізація фронтенд частини



Підключення Bootstrap5

Навігаційне меню

Реалізація фронтенд частини

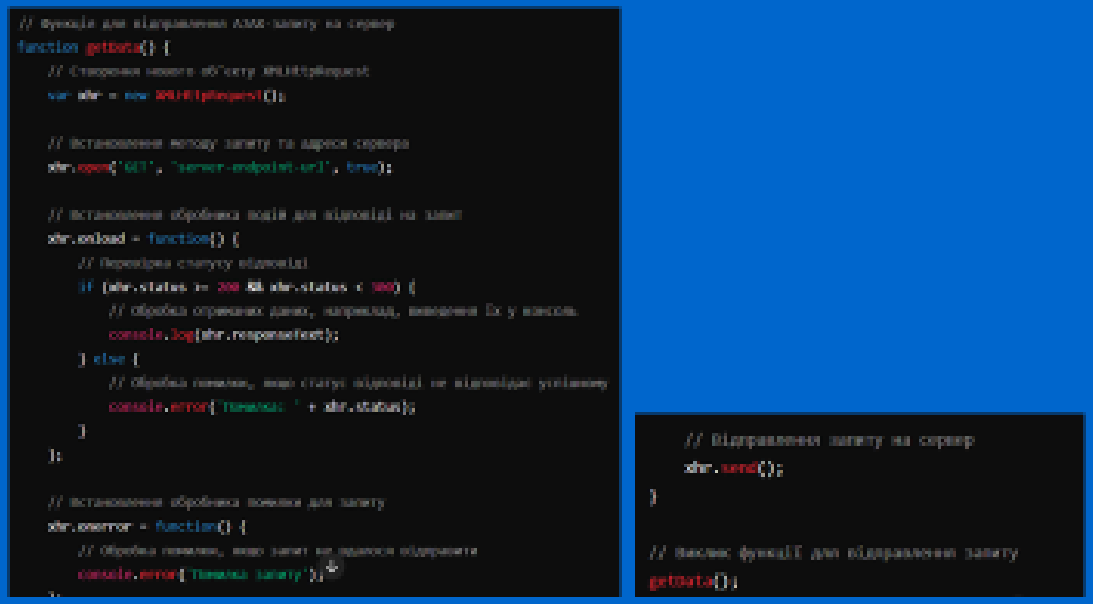
- **2. Реалізація динамічних елементів:**
- - JavaScript та JQuery для інтерактивності.

```
function addToCart(id, name) {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  const existingItem = cart.find(item => item.id === id);

  if (existingItem) {
    existingItem.quantity++;
  } else {
    cart.push({ id, name, quantity: 1 });
  }

  localStorage.setItem('cart', JSON.stringify(cart));
}

function showCart() {
  const cart = JSON.parse(localStorage.getItem('cart')) || [];
  const cartItemsElement = document.getElementById('cart-items');

  cartItemsElement.innerHTML = '';

  cart.forEach(item => {
    const li = document.createElement('li');
    li.className = 'list-group-item d-flex justify-content-between align-items-center';
    li.textContent = `${item.name} (Кількість: ${item.quantity})`;
    const removeButton = document.createElement('button');
    removeButton.className = 'btn btn-danger btn-sm';
    removeButton.textContent = 'Видалити';
    removeButton.addEventListener('click', () => {
      removeFromCart(item.id);
    });
    li.appendChild(removeButton);
    cartItemsElement.appendChild(li);
  });
```

Додавання кошика на сайт

Реалізація фронтенд частини

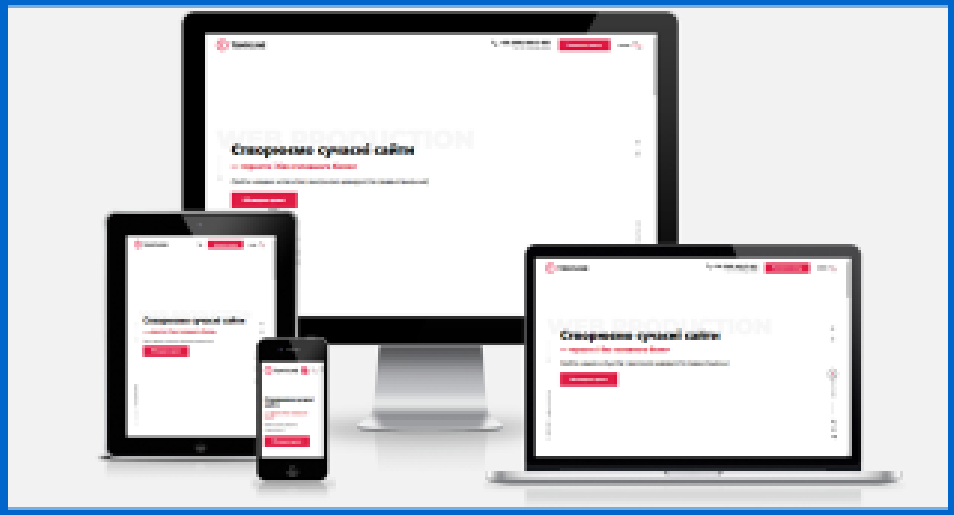
- **3. Інтеграція фронтенду з бекендом:**
- - Використання методів AJAX для взаємодії з сервером.



```
// Функція для відправлення AJAX-запиту на сервер
function getData() {
    // Створення нового об'єкту XMLHttpRequest
    var xhr = new XMLHttpRequest();

    // Встановлення методу запиту та адреси сервера
    xhr.open('GET', 'server-endpoint-url', true);

    // Встановлення обробника подій для відповіді на запит
    xhr.onload = function() {
        // Перевірка статусу відповіді
        if (xhr.status >= 200 && xhr.status < 300) {
            // Обробка отриманих даних, наприклад, виведення їх у консоль
            console.log(xhr.responseText);
        } else {
            // Обробка помилки, якщо статус відповіді не відповідає успішному
            console.error('Помилка: ' + xhr.status);
        }
    };

    // Встановлення обробника помилки для запиту
    xhr.onerror = function() {
        // Обробка помилки, якщо запит не вдалося відправити
        console.error('Помилка запиту');
    };

    // Відправлення запиту на сервер
    xhr.send();
}

// Виклик функції для відправлення запиту
getData();
```

Тестування та оптимізація

- **1. Тестування користувацького інтерфейсу:**
- - Виявлення та усунення багів.



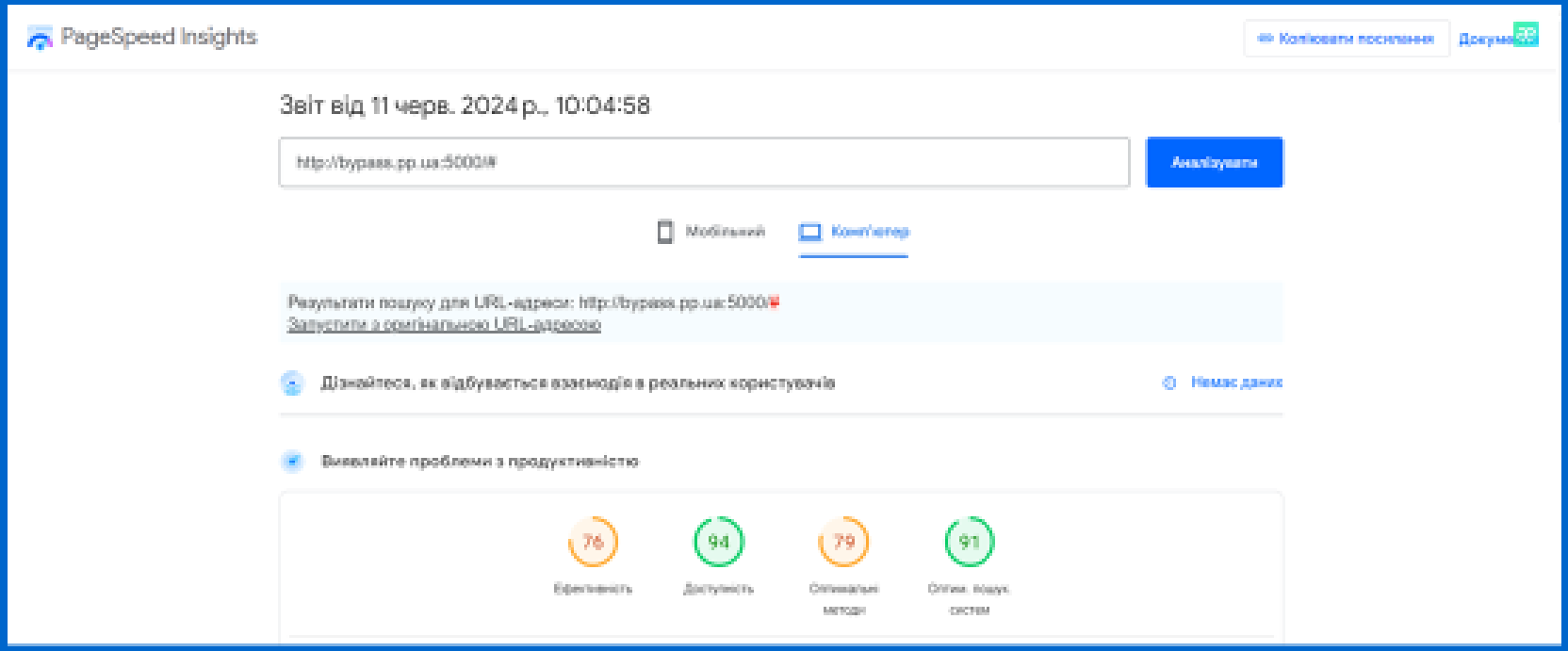
Тестування та оптимізація

- **2. Оптимізація швидкості завантаження сторінок:**
- - Оптимізація зображень, використання кешування.

Тестування та оптимізація

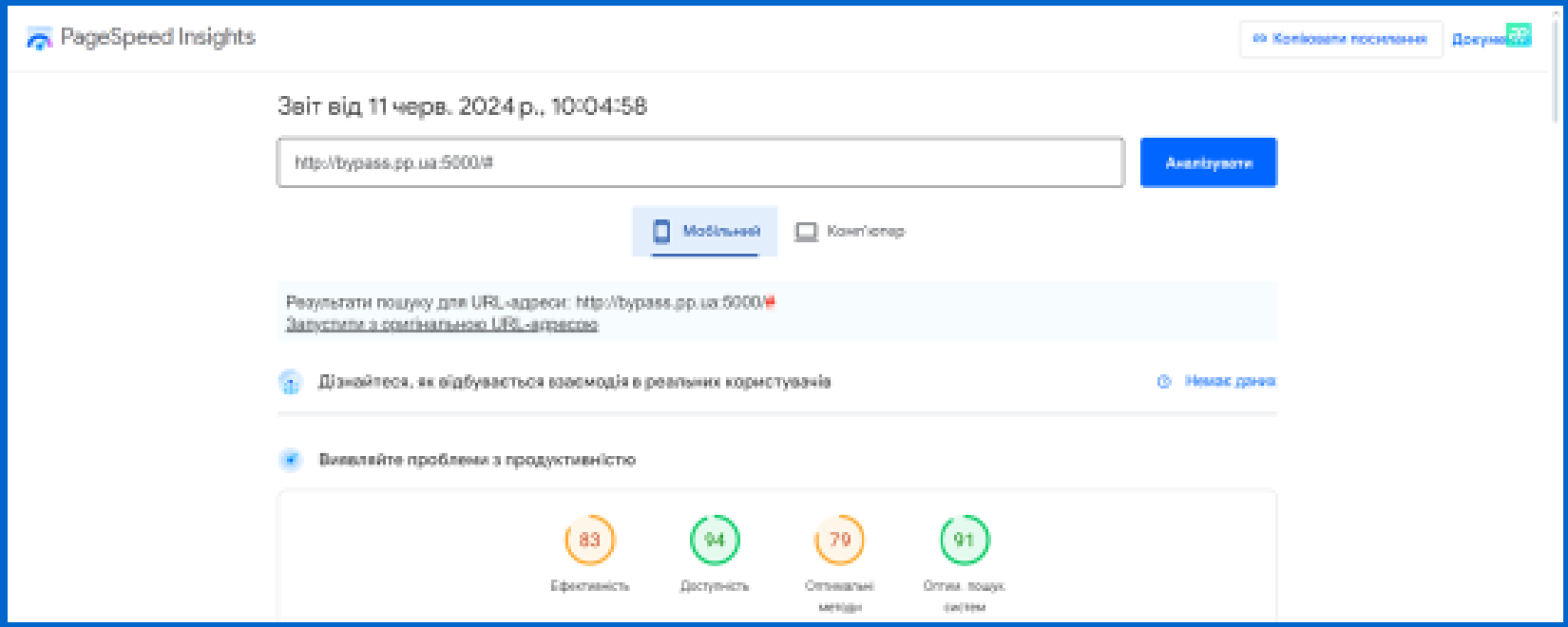
- **3. Інструменти для тестування та оптимізації:**
- - Використання Google Lighthouse, WebPageTest.

Тестування та оптимізація



ПК версія

Тестування та оптимізація



Мобільна версія

Висновки

- **1. Досягнуті результати:**
- - Реалізована фронтенд частина сучасного веб-сайту на основі JavaScript та суміжних технологій.
- **2. Переваги розробленого рішення:**
- - Висока продуктивність, масштабованість, безпека.
- **3. Можливості для подальших досліджень:**
- - Додавання нових функціональностей, покращення безпеки.