Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Кафедра програмної інженерії

КУРСОВА РОБОТА
ПОЯСНЮВАЛЬНА ЗАПИСКА
з дисципліни «Об’єктно-орієнтоване програмування»
ІНТЕРНЕТ-МАГАЗИН КНИГ З ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Керівник , доц. Побіженко І.О
Студент гр. ПЗПІ-21-7 Ткаченко М.А.

Комісія:

Ст. викл. _____ Черепанова Ю.Ю.
Доц. _____ Побіженко І.О.
Доц. _____ Кравець Н.С.

Харків 2022

ХАРКІВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ РАДІОЕЛЕКТРОНІКИ

Кафедра *програмної інженерії*
Рівень вищої освіти *перший (бакалаврський)*
Дисципліна *Об'єктно-орієнтоване програмування*
Спеціальність *121 Інженерія програмного забезпечення*
Освітня програма: *Програмна інженерія*
Курс 1 . Група *ПЗПІ-21-7* . Семестр *2* .

ЗАВДАННЯ
на курсовий проект студента
Ткаченко Максим Андрійович

1 Тема проекту: Інтернет-магазин книг з інформаційних технологій

2 Термін здачі студентом закінченого проекту: ***"1" - липня - 2022 р.***
3 Вихідні дані до проекту:
Специфікація програми, методичні вказівки до виконання курсової роботи

4 Зміст розрахунково-пояснювальної записки:
Вступ, специфікація програми, проектна специфікація, інструкція користувача, висновки

5 Перелік графічного матеріалу:
Діаграма класів, фрагменти алгоритмів, каркасна схема інтерфейсу

КАЛЕНДАРНИЙ ПЛАН

№	*Назва етапу*	*Термін виконання*
1	Видача теми, узгодження і затвердження теми	21.03.2022 р.
2	Формулювання вимог до програми	21.03.2022 – 21.04.2022 р.
3	Проектування архітектури. Створення діаграми класів, каркасної схеми інтерфейсу	21.03.2022 – 12.05.2022 р.
4	Розробка функціоналу та інтерфейсу	21.03.2022 – 12.05.2022 р.
5	Тестування і доопрацювання розробленої програми	10.06.2022 – 12.06.2022 р.
6	Оформлення пояснювальної записки, додатків, графічного матеріалу	12.05.2022 – 18.06.2022 р.
7	Захист	06.06.2022 – 24.06.2022 р.

Студент _____

Керівник _____ Побіженко Ірина Олександрівна

«25» червня_____2022 р.

РЕФЕРАТ

Пояснювальна записка до курсової роботи: 34 с., 17 рис., 1 табл., 3 джерела.

ІНТЕРНЕТ-МАГАЗИН, КЛАСИ, КОМПОНЕНТИ, МОВА ПРОГРАМУВАННЯ JAVASCRIPT, ОБ'ЄКТНО-ОРІЄНТОВАНЕ ПРОГРАМУВАННЯ, САЙТ, CSS, HTML.

Метою роботи є розробка програми-сайту «Інтернет-магазин книг з інформаційних технологій» на засадах об'єктно-орієнтованої парадигми.

Методи розробки базуються на використанні редактору вихідного коду Microsoft Visual Studio Code 2020, хостінгах Netlify**,** jsonbin.io, imgbb, мови програмування JavaScript, мови розмітки HTML і стилів CSS. Також, використовувалась при розробці система контролю версій Git Bash та хостинг вихідного коду GitHub.

В результаті отримана програма під назвою "Інтернет-магазин книг з інформаційних технологій", яка дозволяє замовляти спеціалізовану літературу по інформаційним технологіям, переглянути каталог книг, додати до корзини товари і оформити замовлення за допомогою оброблюваної форми.

ЗМІСТ

ВСТУП

Мета курсового проектування – розробити програму-сайт в об’єктно-орієнтованій парадигмі, написаний на чистому JavaScript. Створити масштабований, читабельний, багаторазово використовуваний код за допомогою класів “фабрик об’єктів”. Спроектувати архітектуру програми з графічним інтерфейсом користувача. Використовувати переваги мови програмування JavaScript в веб-розробці динамічного інтерактивного сайту. Розмістити сайт у відкритому доступі в мережі з використанням хостингів. Створити програму з використанням інтерфейсів прикладного програмування. Створити програму за допомогою маніпуляцій з деревом об'єктної моделі документа. Розібратися з механізмом браузера під капотом. Створити компоненту класову структуру на мові програмування JavaScript. Використовувати конвенції. Використовувати класи і функції, щоб приховувати внутрішню роботу програми для керованості програми. Ретельно підготуватися до розробки: визначити проблему, виробити вимоги, спроектувати архітектуру. Створити програму пройшовши послідовно етапи розробки програми.

Галузь застосування – електронна торгівля книгами з інформаційних технологій.

1 СПЕЦИФІКАЦІЯ ПРОГРАМИ

1.1 Функціонал програми

Програма надає користувачу наступний функціонал:

- отримання даних із серверів-хостингів асинхронно з підтримкою обміну ресурсами з багатьох джерел;
- рендер каталогу товарів;
- додавання/видалення товару з корзини;
- обробка помилки отримання даних із серверів;
- обробка форми;
- локальне зберігання товарів на стороні клієнта на комп'ютері користувача, щоб не перезавантажувати ресурси із серверів та зберегти вибір товару, доданого до корзини;
- спіннер завантаження;
- внутрішній пошук по сайту.

Програма надає адміністратору наступний функціонал:

- обробка форми, отримання даних форми на пошту;
- захист від спаму ботів за допомогою спам-фільтру Akismet і поля приманки;
- валідація даних на стороні клієнту, що вводяться у форму.

1.2 Сценарії функцій користувача

1.2.1 Сценарій “Внутрішній пошук по сайту”

1. Користувач клацає поле пошуку в шапці сайту (див. рис. 1.1).
2. Користувач вводить запит назви книги або автора та натискає кнопку "Пошук".
3. Введені дані перевіряються на збіги і спливає вікно сповіщення про збіги.
4. Якщо збігів немає, з'являється текстове повідомлення “Немає збігів пошуку”. Якщо збіги є, то з'являється текстове повідомлення назв і авторів книг в наявності.

Рисунок 1.1 – Каркасна схема класу “Header”

1.2.2 Сценарій “Додавання / видалення товару з корзини”

1. У вікні інтерфейсу користувача (див. рис. 1.4) користувач клацає кнопку “Додати до корзини” на потрібній картці товару.
2. Якщо користувач бажає видалити товар, натискається кнопка "Видалити з корзини" на потрібній картці товару.

1.2.3 Сценарій “Корзина”

1. У шапці сайту (див. рис. 1.1) користувач клацає напис “КОРЗИНА”.
2. З'явиться вікно корзини (див. рис. 1.2) з накладеними товарами та підсумковою сумою.

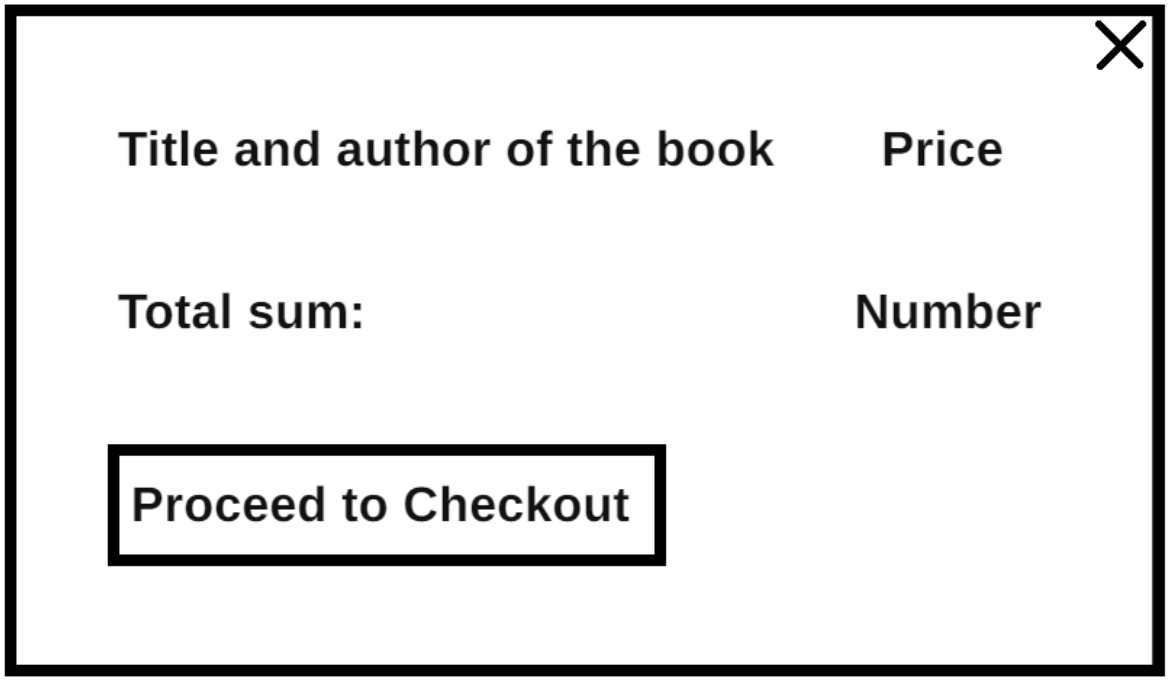

Рисунок 1.2 – Каркасна схема “Корзина”

1.2.4 Сценарій “Форма”

1. Користувач клацає кнопку “Оформити замовлення” у вікні корзини (див. рис. 1.2), яке відкрито за допомогою сценарію “Корзина”.
2. Відкривається сторінка “Форми” (див. рис. 1.3).
3. Користувач заповнює поля форми.
4. Користувач клацає кнопку “Відправити”.

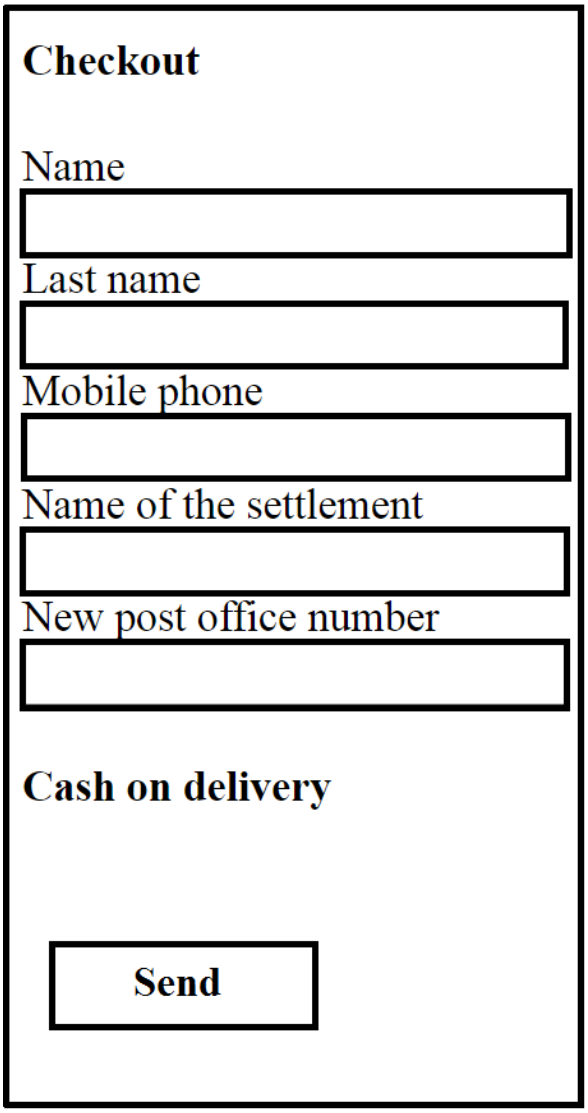

Рисунок 1.3 – Каркасна схема “Форма”

1.3 Сценарії функцій адміністратора

1.3.1 Сценарій “Обрабка форми. Отримання даних форми на пошту за допомогою хостінгу Netlify”

1. Відкрити сайт Gmail за адресою URL mail.google.com
2. Відкрити останнє повідомлення у вхідних від Netlify по темі “Надсилання форми з контактної форми”.

1.4 Детальний опис інтерфейсу користувача

1.4.1 Карткова логічна структура і компонентна файлова структура

Вибір карткової логічної структури для зручного перегляду каталогу книжної продукції. Розглянемо каркасну схему інтерфейсу користувача на рисунку 1.4.

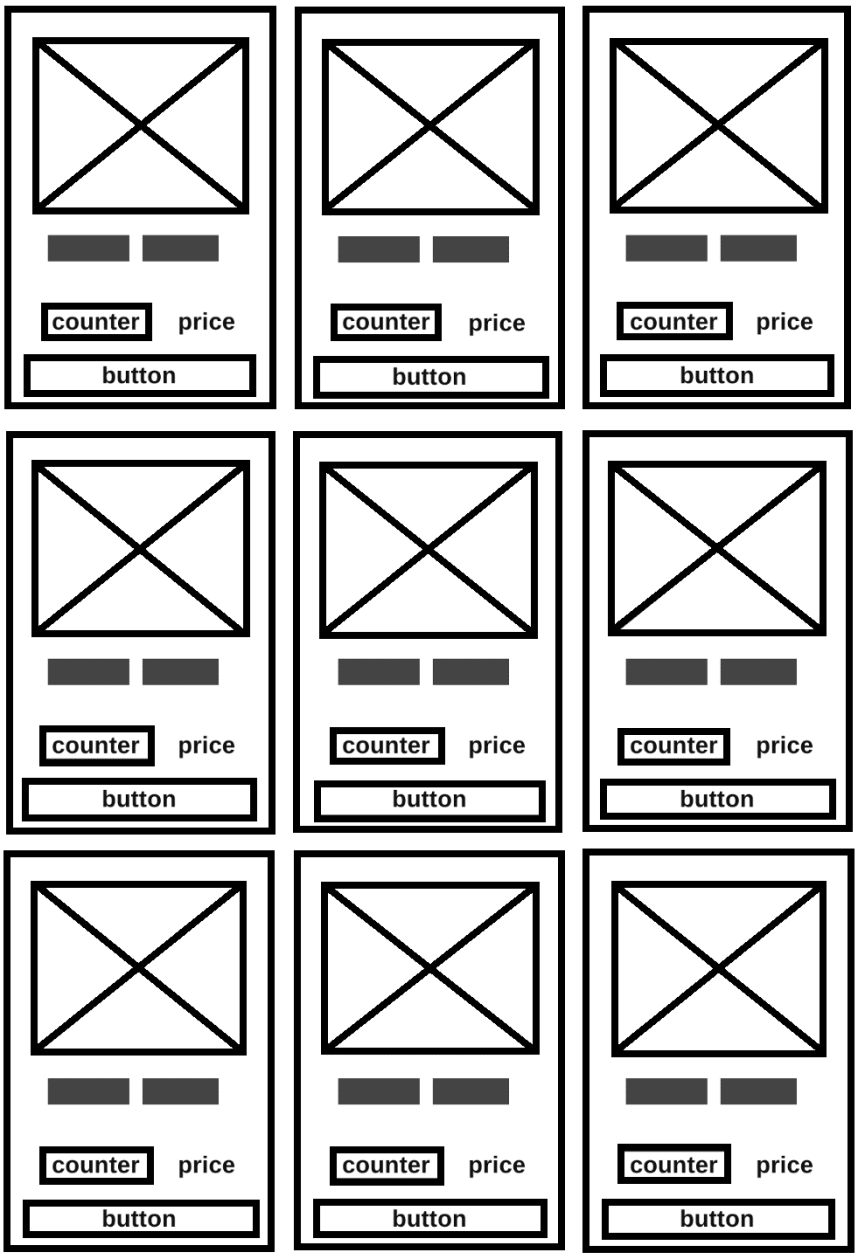

Рисунок 1.4 — Каркасна схема інтерфейсу користувача

На цьому місці закінчується опис функціоналу і інтерфейсу.

2 ПРОЕКТНА СПЕЦИФІКАЦІЯ

2.1 Детальний опис даних

Архітектура: веб графічний інтерфейс.

Основний формат зберігання зовнішніх даних: json.

Зберігання бази даних json на хостінгу jsonbin.io, зберігання бази даних зображень на хостінгу imgbb, зберігання бази даних файлів JavaScript, HTML, CSS, фавікону, іконки хрестика, svg спінеру на хостингу Netlify.

Дані товарів, що зберігаються в json: унікальний ідентифікатор, найменування, зображення, ціна.

Дані форми замовлення о покупці: ім'я, прізвище, мобільний телефон, назва населеного пункту, номер відділення нової пошти.

Розглянемо діаграму класів на рисунку 2.1.

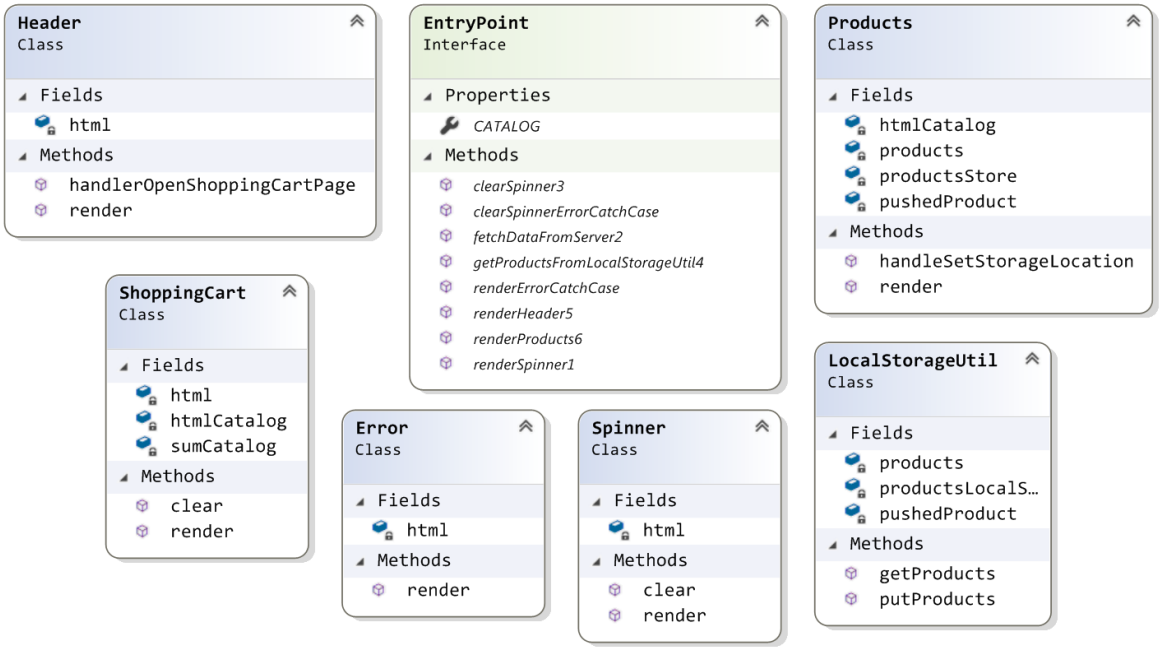

Рисунок 2.1 — Діаграма класів загальної структури програми

Розглянемо класову файлову структуру на рисунку 2.2.

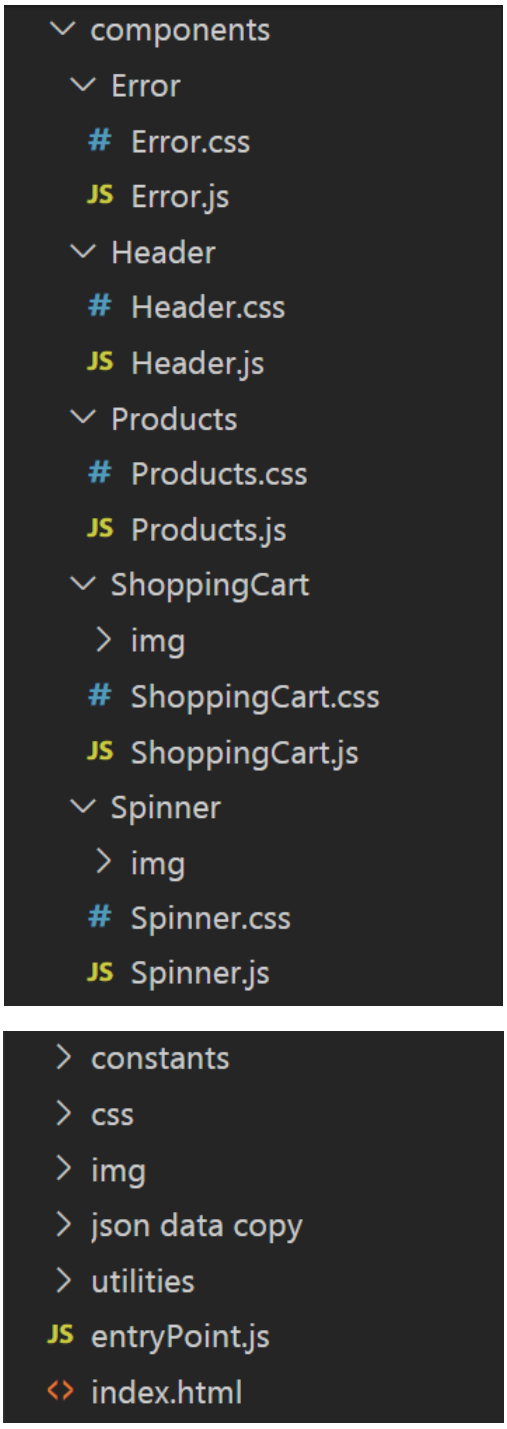

Рисунок 2.2 — Класова файлова структура

2.2 Детальний опис об’єктної структури

Розглянемо компонентну класову структуру.

2.2.1 Інтефейс “entryPoint”. Одна точка входу в програму

Загальний інтерфейс програми, в якому міститься функція отримання даних з сервера-хостінгу jsonbin.io за домогою Fetch API у вигляді then - catch з обробкою помилки отримання даних. Отриманий json проміс перетворюється в масив. Дані зберігаються в локальне сховище за допомогою Storage API. При завантаженні даних відображається спіннер. Потім, викликаються функції рендерингу (візуалізації) шапки і товарів. Розглянемо точку входу всієї програми на рисунку 2.3.

```
function render() {
    const productsStore =
localStorageUtil.getProducts();


    headerPage.render(productsStore.length);
    productsPage.render();
}


spinnerPage.render();


let CATALOG = [];


fetch('https://api.jsonbin.io/b/6262a7bebc312b30ebeb5dd
a'
).then(response => response.json())
    .then(data => {
```

```
    CATALOG = data;
    spinnerPage.handleClear();
    render();
})
.catch(() => {
    spinnerPage.handleClear();
    errorPage.render();
});
```

Рисунок 2.3 — Фрагмент коду інтерфейсу “entryPoint”

2.2.2 Детальний опис інтерфейсу “entryPoint” з усім функціоналом

Інтерфейс “entryPoint” складається з публічної глобальної змінної CATALOG, асинхронних функцій fetch(), then(), catch(), і викликів функцій у визначеному порядку: spinnerPage.render(), fetch('string') , spinnerPage.handleClear(), localStorageUtil.getProducts(), headerPage.render(), productsPage.render(), і у випадку помилки отримання даних із серверу-хостінгу викликати функції spinnerPage.handleClear(), errorPage.render().

2.2.3 Клас “Header”

Клас “Header” складається з приватної змінної html, функцій handlerOpenShoppingCartPage(), render(). Після цього класу йде функція внутрішнього пошуку сайту. Розглянемо клас “Header” на рисунку 2.4.

```
class Header {
    handlerOpenShoppingCartPage() {
        shoppingCartPage.render();
    }
    render(count) {
        const html = `
        <div class="header-container">
        <div id="searchBox">
            <input type="text" id="input"
placeholder="Поиск книг">
            <input type="button" id="searchBtn"
value="Поиск" onclick="search();">
        </div>
            <div class="header-counter"
onclick="headerPage.handlerOpenShoppingCartPage();">
              КОРЗИНА ${count}
            </div>
        </div>
        `;

        ROOT_HEADER.innerHTML = html;
    }
}
const headerPage = new Header();
```

```
const array = ["Грокаем алгоритмы. Адитья Бхаргава",
"Код. Чарльз Петцольд", "Совершенный код. Стив
Макконнелл",
    "Чистая архитектура. Роберт Мартин", "Чистый код.
Роберт Мартин", "Тестирование DOT COM. Роман Савин.",
    "CLR via C#. Джеффри Рихтер", "Паттерны
проектирования. Эрик Фримен", "Рефакторинг. Мартин
Фаулер",
    "Замыкания и Объекты. Кайл Симпсон"];


window.addEventListener("keydown", event => {
    if (event.keyCode === 13) { search(); }
});


function search() {
    let input = document.getElementById("input");


    const lowerCaseArray = array.map(element => {
        return element.toLowerCase();
    });


    let substring = input.value.toLowerCase();
    const matches = lowerCaseArray.filter(element => {
        if (element.indexOf(substring) !== -1 &&
substring.length !== 0) {
            return true;
        }
    });
    console.log(matches);
    if (matches.length === 0) {
        alert("Нет совпадений поиска");
```

```
    } else {
        alert(`Есть совпадение поиска: ${matches}`);
    }
}
```

Рисунок 2.4 — Фрагмент коду класу “Header” і функції внутрішнього пошуку сайту

2.2.4 Клас “ShoppingCart”

Клас “ShoppingCart” складається з приватної змінної html, htmlCatalog, sumCatalog, функцій handleClear(), render(). Розглянемо клас “ShoppingCart” на рисунку 2.5.

```
class ShoppingCart {
    handleClear() {
        ROOT_SHOPPING_CART.innerHTML = '';
    }

    render() {

        const productsStore =
localStorageUtil.getProducts();
        let htmlCatalog = '';
        let sumCatalog = 0;

        CATALOG.forEach(({ id, name, price }) => {
            if (productsStore.indexOf(id) !== -1) {
                htmlCatalog += `
                    <tr>
```

```
                        <td class="shopping-cart-
element__name">⚡ ${name}</td>
                        <td class="shopping-cart-
element__price">${price.toString()} грн</td>
                    </tr>
                `;
                sumCatalog += price;
            }
        });

        const html = `
            <div class="shopping-cart-container">
                <div class="shopping-cart__close"
onclick="shoppingCartPage.handleClear()"></div>
                <table>
                    ${htmlCatalog}
                    <tr>
                        <td class="shopping-cart-
element__name">✹ Сумма:</td>
                        <td class="shopping-cart-
element__price">${sumCatalog.toString()} грн</td>
                    </tr>
                </table>
                <button class="btn1"
onclick="document.location='https://ordering-it-
books.netlify.app'">Оформить заказ</button>
            </div>
        `;
        ROOT_SHOPPING_CART.innerHTML = html;
    }
}
```

```
const shoppingCartPage = new ShoppingCart();
```

Рисунок 2.5 — Фрагмент коду класу “ShoppingCart”

2.2.5 Клас “Error”

Клас “Error” складається з приватної змінної html, функцій render(). Розглянемо клас “Error” на рисунку 2.6.

```
class Error {
    render() {
        const html = `
        <div class="error-container">
            <div class="error-message">
                <h3>Ошибка. Не удалось получить
данные.</h3>
            </div>
        </div>
        `;

        ROOT_ERROR.innerHTML = html;
    }
}

const errorPage = new Error();
```

Рисунок 2.6 — Фрагмент коду класу “Error”

2.2.6 Клас “Spinner”

Клас “Spinner” складається з приватної змінної html, функцій handleClear(), render(). Розглянемо клас “Spinner” на рисунку 2.7.

```
class Spinner {
    handleClear() {
        ROOT_SPINNER.innerHTML = '';
    }

    render() {
        const html = `
            <div class="spinner-container">
                <img class="spinner__img"
src="components/Spinner/img/spinner.svg"/>
            </div>
        `;

        ROOT_SPINNER.innerHTML = html;
    }
}

const spinnerPage = new Spinner();
```

Рисунок 2.7 — Фрагмент коду класу “Spinner”

2.2.7 Клас “LocalStorageUtil”

Клас “LocalStorageUtil” складається з приватних змінних товари, productsLocalStorage, pushProduct, функцій getProducts(), putProducts(). Розглянемо клас “LocalStorageUtil” на рисунку 2.8.

```
class LocalStorageUtil {
    constructor() {
        this.keyName = 'products';
    }
    getProducts() {
        const productsLocalStorage =
localStorage.getItem(this.keyName);
        if (productsLocalStorage !== null) {
            return JSON.parse(productsLocalStorage);
        }
        else return [];
    }
    putProducts(id) {
        let products = this.getProducts();
        let pushProduct = false;
        const index = products.indexOf(id);
        if (index === -1) {
            products.push(id);
            pushProduct = true;
        } else {
            products.splice(index, 1);
        }
        localStorage.setItem(this.keyName,
JSON.stringify(products));
```

```
        return { pushProduct, products }
    }
}
const localStorageUtil = new LocalStorageUtil();
```

Рисунок 2.8 — Фрагмент коду класу “LocalStorageUtil”

2.2.8 Клас “Products”

Клас “Products” складається з приватних змінних html, products, productsStore, pushProduct, функцій handleSetStorageLocation(), render(). Розглянемо клас “Products” на рисунку 2.9.

```
class Products {
    constructor() {
        this.classNameActive = 'products-
element__btn_active';
        this.labelAdd = 'Добавить в корзину';
        this.labelRemove = 'Удалить из корзины';
    }

    handleSetStorageLocation(element, id) {
        const { pushProduct, products } =
localStorageUtil.putProducts(id);

        if (pushProduct) {

element.classList.add(this.classNameActive);
            element.innerHTML = this.labelRemove;
        } else {

element.classList.remove(this.classNameActive);
```

```
            element.innerHTML = this.labelAdd;
        }

        headerPage.render(products.length);
    }

    render() {
        const productsStore =
localStorageUtil.getProducts();
        let htmlCatalog = '';

        CATALOG.forEach(({ id, name, price, img }) => {
            let activeClass = '';
            let activeText = '';

            if (productsStore.indexOf(id) === -1) {
                activeText = this.labelAdd;
            } else {
                activeClass = ' ' +
this.classNameActive;
                activeText = this.labelRemove;
            }

            htmlCatalog += `
                <li class="products-element">
                    <span class="products-
element__name">${name}</span>
                    <img class="products-element__img"
src="${img}" />
                    <span class="products-
element__price">
```

```
                        ⚡ ${price.toString()} грн
                    </span>
                    <button class="products-
element__btn${activeClass}"

onclick="productsPage.handleSetStorageLocation(this,
'${id}');">
                        ${activeText}
                    </button>
                </li>
            `;
        });


        const html = `
            <ul class="products-container">
                ${htmlCatalog}
            </ul>
        `;


        ROOT_PRODUCTS.innerHTML = html;
    }
}
const productsPage = new Products();
```

Рисунок 2.9 — Фрагмент коду класу “Products”

2.3 Детальний опис функціоналу програми для користувача

2.3.1 Отримання даних із сервера асинхронно

Fetch API надає інтерфейс JavaScript для доступу та маніпулювання запитами та відповідями HTTP-конвеєра. Fetch API надає інтерфейс для асинхронного отримання ресурсів через мережу з обміном ресурсами з безлічі джерел. Потім відповідь перетворюється на json формат[0].

2.3.2 Рендер каталогу товарів

Для рендеру сайту використовується об'єктна модель документа і функції render().

2.3.3 Додавання/видалення товару з корзини

Додавання товару в корзину і видалення товару з корзини за допомогою властивості classList.

2.3.4 Обробка помилки

Використовується функція catch() для обробки невдалого запиту і виведення повідомлення про помилку.

2.3.5 Обробка форми

Використовуваний хостінг Netlify поставляється із вбудованою обробкою форм. Боти-розбирачі роблять це шляхом розбирання HTML-файлів безпосередньо під час розгортання. Всі відправлені форми фільтруються на предмет спаму за допомогою Akismet. А також доданий додатковий захист від спаму. Автоматично відхиляються заявки, що не пройшли перевірку поля приманки. Поля приманки - це приховані поля форми, які заманюють ботів заповнити поле, яке не можуть виявити користувачі-люди. Форма, надіслана із заповненим полем приманки, може

бути безпечно відхилена, тому що тільки бот може виявити та заповнити це поле[1].

2.3.6 Локальне сховище

Локальне зберігання товарів в постійній пам'яті на стороні клієнта. Було використано Storage API.

2.3.7 Спіннер завантаження

Використана масштабована векторна графіка, функція render() і об'єктна модель документа.

2.3.8 Внутрішній пошук по сайту

Головна функція цього алгоритму це filter(), яка відфільтровує збіги підрядків з масиву рядків.

2.4 Детальний опис функціоналу програми для адміністратору

2.4.1 Обробка форми. Отриманням даних форми на пошту

Форма оброблюється за допомогою хостінгу Netlify і дані форми відправляються на пошту.

2.4.2 Захист від спаму ботів

Спам-фільтр Akismet і поля приманки поставляються із хостингом Netlify.

2.4.3 Валідація даних форми

Дані форми перевіряються на стороні клієнта за допомогою атрибутів HTML та регулярного виразу.

3 ІНСТРУКЦІЯ КОРИСТУВАЧА

3.1 Як користуватися програмою

Перший крок – перевірити інтернет-з’єднання. Другий крок – запустити браузер. Третій крок - набрати в пошукову систему запит з адресою сайту https://buy-it-books.netlify.app/.

Четвертий крок – додати потрібний товар до корзини, клацнувши кнопку “Додати до корзини”. Розглянемо четвертий крок на рисунку 3.1.

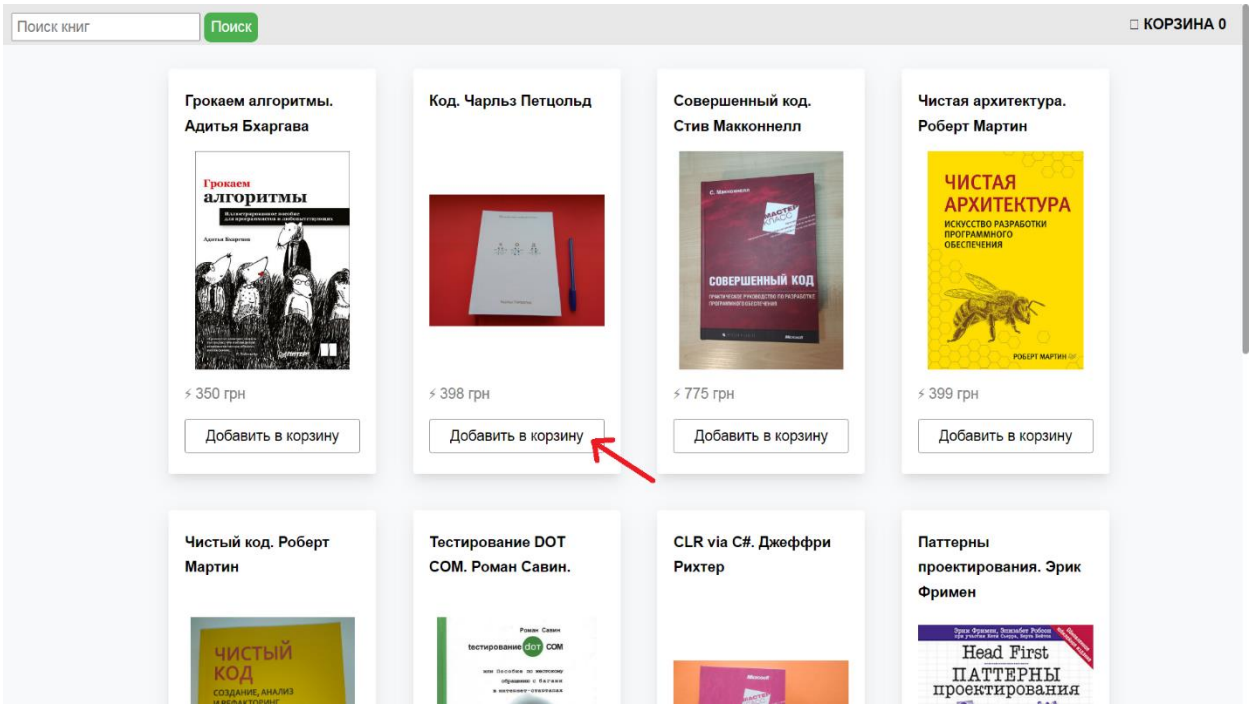

Рисунок 3.1 — Четвертий крок “Додавання товару до корзини”

П’ятий крок – натиснути напис КОРЗИНА. Розглянемо п’ятий крок на рисунку 3.2.

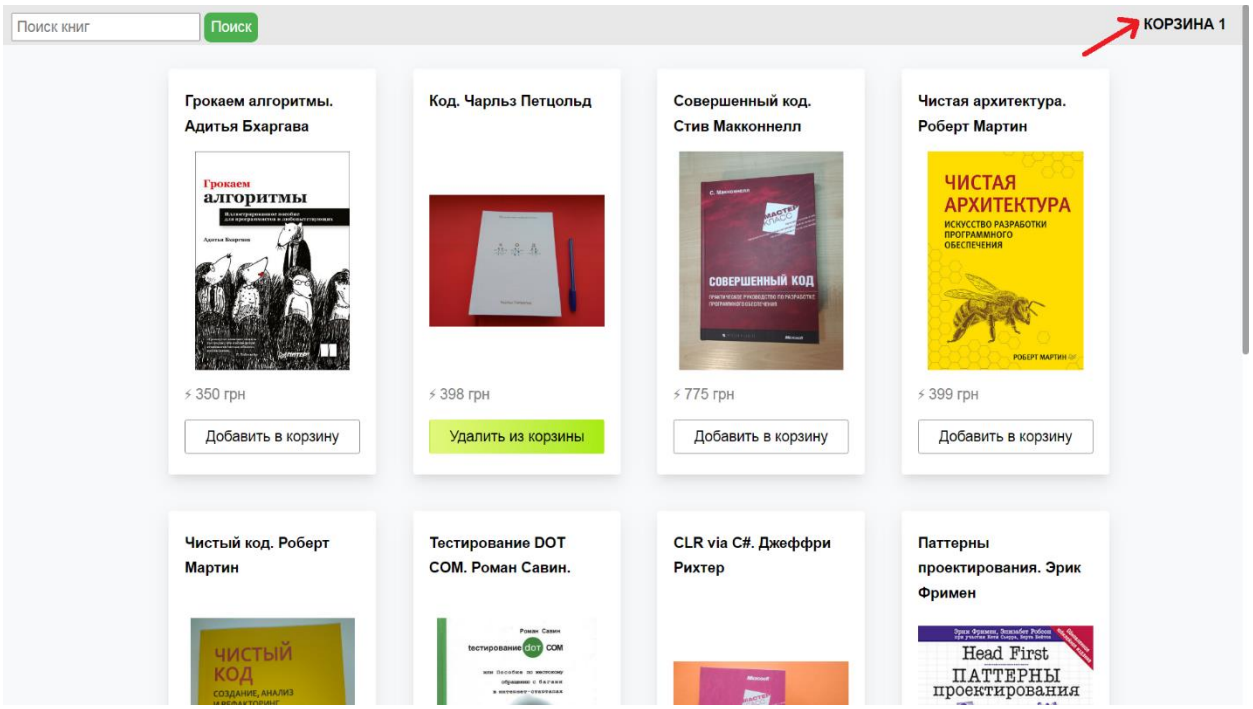

Рисунок 3.2 — П'ятий крок “Натиснення напису КОРЗИНА”

Шостий крок – натиснути кнопку “Оформити замовлення”. Розглянемо шостий крок на рисунку 3.3.

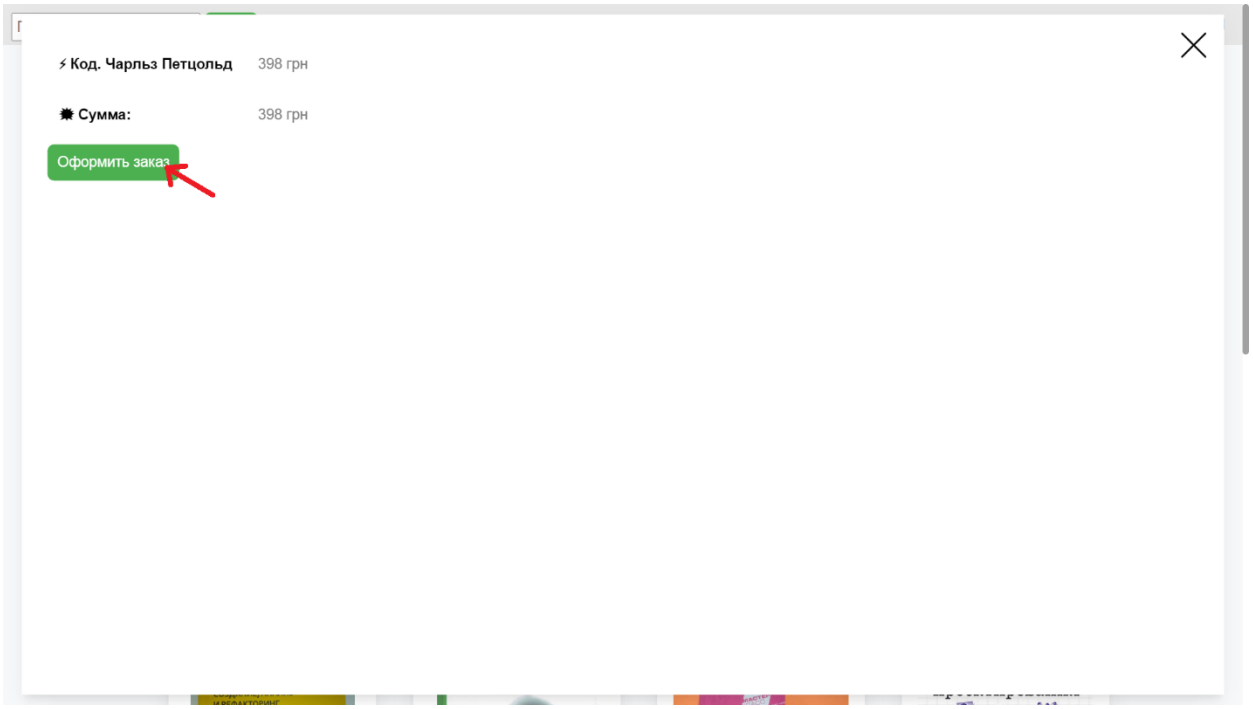

Рисунок 3.3 — Шостий крок “Натиснення кнопки Оформити замовлення”

Сьомий крок – заповнити поля форми та натиснути кнопку “Відправити”. Розглянемо сьомий крок на рисунку 3.4.

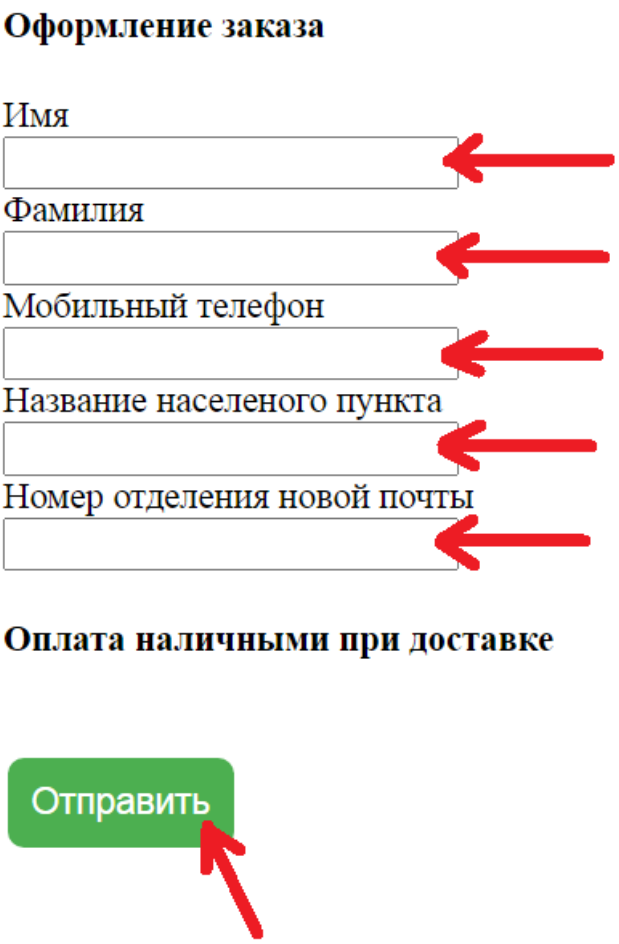

Рисунок 3.4 — Сьомий крок “Заповнення полів форми та натискання кнопки Відправити”

На цьому місці закінчується інструкція, як користуватися програмою.

ВИСНОВКИ

Розроблено програму-сайт з теми “Інтернет-магазин книг інформаційних технологій” в об’єктно-орієнтованій парадигмі з використанням інтерфейсів прикладного програмування, і який розміщено у відкритому доступі в мережі з використанням хостингів. Виконано ретельну підготовку, тобто виконані наступні етапи розробки сайту: визначення проблеми, вироблення вимог, проектування архітектури. Спроектовано компоненту класову архітектуру програми з графічним інтерфейсом користувача. Розробка програми-сайту в об’єктно орієнтованій парадигмі має такі переваги як економія часу на розробку та вища продуктивність, краща якість програмного забезпечення та менші витрати на обслуговування, масштабованість та можливість повторного використання коду. Використовувалися конвенції іменування змінних та функцій. Використовувалися конвенції оголошування та надання значень локальним змінним на початку функції в одному місці. Використовувалися принципи приховування внутрішньої роботи програми за допомогою абстракції та інкапсуляції для керованості програми. Додано графічний матеріал для кращого розуміння: схема класів, фрагменти алгоритмів, каркасна схема інтерфейсу. Наприклад, створено діаграму класів для розробки, редагування та рефакторингу класів, а також візуалізації різних частин коду. Можливий шлях розвитку програми – це масштабування сайту, додавання більше книг і різних категорій. Початкова галузь застосування сайту — це електронна торгівля книгами з інформаційних технологій.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Using the Fetch API - Web APIs | *MDN Web Docs*. URL: https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API/Using_Fetch (дата звернення: 12.05.2022).
2. Forms setup. *Netlify Docs*. URL: https://docs.netlify.com/forms/setup/?_ga=2.80674199.1296566291.1652341374-1516064.1650632455 (дата звернення: 12.05.2022).
3. JavaScript Tutorial. *W3Schools Online Web Tutorials*. URL: https://www.w3schools.com/js/default.asp (date of access: 21.06.2022).