

Міністерство освіти і науки України  
Державний заклад  
«Луганський національний університет імені Тараса Шевченка»

Навчально-науковий інститут фізики, математики та інформаційних  
технологій

Кафедра інформаційних технологій та систем

Терещенко Дмитро Олексійович

## **СТВОРЕННЯ ФРОНТЕНДА ДЛЯ ОНЛАЙН МАГАЗИНУ НА ОСНОВІ REACT-REDUX**

**кваліфікаційна робота**

**здобувача вищої освіти першого (бакалаврського) рівня  
освітньої програми «Інженерія програмного забезпечення»  
за спеціальністю 121 Інженерія програмного забезпечення**

Особистий підпис \_\_\_\_\_ Дмитро ТЕРЕЩЕНКО

Науковий керівник \_\_\_\_\_ Володимир МАТІЄВСЬКИЙ,  
асистент  
кафедри інформаційних технологій  
та систем

Завідувач кафедри \_\_\_\_\_ Микола СЕМЕНОВ,  
кандидат педагогічних наук, доцент  
кафедри інформаційних технологій  
та систем

Полтава – 2023

**Міністерство освіти і науки України**  
**Державний заклад**  
**„Луганський національний університет імені Тараса Шевченка”**

Факультет (інститут)	Навчально-науковий інститут фізики, математики та інформаційних технологій
Кафедра, циклова комісія	Інформаційних технологій та систем
Освітній ступень	Бакалавр
Напрямок підготовки (спеціальність)	121 Інженерія програмного забезпечення (код, назва)
Галузь знань	12, Інформаційні технології (код, назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри ІТС

\_\_\_\_\_  
(підпис)

М.А. Семенов  
(ініціали, прізвище)

“ \_ ” \_\_\_\_\_ 2022 р.

**ЗАВДАННЯ**  
**НА БАКАЛАВРСЬКУ РОБОТУ**  
**Терещенку Дмитру Олексійовичу**

(прізвище, ім'я, по батькові )

**1.Тема проекту (роботи) Створення фронтенда для онлайн магазину на основі React-Redux**

Керівник кваліфікаційної роботи Матієвський В.В.

(прізвище, ініціали, науковий ступінь, вчене звання)

затверджена наказом по університету від \_\_\_\_\_

**2.Строк подання студентом проекту (роботи)** \_\_\_\_\_

**3.Вихідні дані до роботи(проекту) у результаті виконання роботи**

(визначаються кількісні або (та) якісні показники, яким повинен відповідати об'єкт розробки)

повинно бути розроблено фронтенд для магазину із використанням React-Redux

**4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)**

ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ФРОНТЕНДА САЙТІВ

ОГЛЯД МОЖЛИВОСТЕЙ REACT & REDUX

ПРАКТИЧНА РЕАЛІЗАЦІЯ ФРОНТЕНД САЙТУ НА ОСНОВІ REACT-REDUX

(визначаються назви розділів або (та) перелік питань, які повинні увійти до тексту ПЗ)

**5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)**

## 6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання „\_\_\_\_\_” 2022 р.

### КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1.	Вибір теми роботи, вивчення наукової літератури, затвердження теми та керівника.	До 15 жовтня	
2.	Аналіз літературних джерел за темою роботи. Розробка та апробація методики дослідно-експериментальної роботи. Подання структури теоретичної частини роботи та плану експериментальних досліджень.	До 1 лютого	
3.	Робота над теоретичною частиною. Подання теоретичної частини роботи для першого читання науковим керівником.	До 15 лютого	
4.	Усунення зауважень, урахування рекомендацій наукового керівника. Подання теоретичної частини роботи на друге читання.	До 5 квітня	
5.	Проведення експериментальної роботи. Поетапний аналіз та обговорення її результатів. Перевірка стану виконання роботи.	Перший тиждень квітня	
6.	Урахування рекомендацій наукового керівника, усунення недоліків, підготовка варіанта роботи до передзахисту. Розробка презентації.	До 28 квітня	
7.	Попередній захист роботи на кафедрі	травень	
8.	Доопрацювання роботи з урахуванням рекомендацій після передзахисту. Подання роботи науковому керівникові та рецензентові на підготовку відгуку та рецензії	За 10 днів до державної атестації	
9.	Подання на кафедру остаточного варіанта роботи, переплетеного та підписаного автором, науковим керівником і рецензентом.	За 5 днів до державної атестації	

Студент

\_\_\_\_\_

підпис

Д. О. Терещенко

\_\_\_\_\_

(ініціали, прізвище)

Керівник роботи

\_\_\_\_\_

підпис

В. В. Матієвський

\_\_\_\_\_

(ініціали, прізвище)

## АНОТАЦІЯ

**Терещенко Д. О.**

**Тема:** Створення фронтенда для онлайн магазину на основі React-Redux

**Спеціальність:** 121 «Інженерія програмного забезпечення».

**Установа:** ЛНУ імені Тараса Шевченка, 2023 р.

**Кваліфікаційна робота:** 54 с., 13 рис., 10 джерел, 2 додатка.

**Мета роботи** – створення фронтед частини для інтернет магазину за допомогою React Redux.

**Об’єкт дослідження** – процес створення frontend частини сайту

**Предмет дослідження** – процес створення frontend частини сайту та розробка інтерфейсу за допомогою React Redux.

**Методи дослідження:** *теоретичні:* аналіз наукової літератури, узагальнення та систематизація теоретичних положень про створення веб-сайтів; аналіз технологій створення веб-сайтів; *експериментальні:* тестування розробленого сайту.

**Практичне значення розробки:** розроблений фронтед який може бути використаний як посібник для розробки інтерфейсу інтернет-магазину на основі React Redux.

**Висновки:** Було розглянуті основні інструменти розробника при роботі над фронтед частиною веб-додатків. Проведено огляд фреймворків React, Vue, Angular, вивчено основні можливості. Зроблено порівняльний аналіз цих фреймворків. Було детально розглянуто фреймворк React, та бібліотеку Redux; основні програмні прийоми, необхідні при роботі з цими бібліотеками. В ході роботи було розроблено фронтед частину для інтернет-магазину на основі React Redux.

**Ключові слова:** САЙТ, HTML, CSS, JS, REACT, REDUX

## ABSTRACT

**Dmytro Tereshchenko**

**Topic:** Creating a frontend for an online store based on React-Redux

**Specialty:** 121 "Software Engineering".

**Institution:** Luhansk Taras Shevchenko National University

**Qualification work:** 54 pages, 13 figures, 10 sources, 2 appendices.

**The purpose of the work:** creation of the front-end part for the online store using React Redux.

**Object of research:** the process of creating the frontend part of the site.

**The subject of research:** the process of creating the frontend part of the site and developing the interface using React Redux.

**Research methods:** theoretical: analysis of scientific literature, generalization and systematization of theoretical provisions on the creation of websites; analysis of website creation technologies; experimental: testing the developed site.

**Practical significance of the development:** developed a frontend that can be used as a guide to develop an online store interface based on React Redux.

**Conclusions:** The main developer tools for working on the front-end part of web applications were considered. An overview of the React, Vue, Angular frameworks was conducted, and the main capabilities were studied. A comparative analysis of these frameworks was made. The React framework and the Redux library were discussed in detail; basic programming techniques needed when working with these libraries. In the course of the work, the front-end part for the online store was developed based on React Redux.

**Keywords:** SITE, HTML, CSS, JS, REACT, REDUX

**ІТС ІПЗ4.0423-ВІ**  
**ВІДОМІСТЬ ПРОЄКТУ, СТВОРЕННЯ ФРОНТЕНДА ДЛЯ ОНЛАЙН**  
**МАГАЗИНУ НА ОСНОВІ REACT-REDUX**

Позначення	Найменування	Кількість прим/стор	Місцезна-ходження/Примітка
	Документація проєкту		
ІТС.ІПЗ4.0423-02-ТЗ	Створення фронтенда для	1/6	Формат А4
	Онлайн магазину на		
	основі React-Redux		
	Технічне завдання		
ІТС.ІПЗ4.0423-04-ІЗ	Створення фронтенда для	1/54	Формат А4
	Онлайн магазину на		
	основі React-Redux		
	Пояснювальна записка		
ІТС.ІПЗ4.0423-04-МТ	Створення фронтенда для	1/12	Формат А4
	Онлайн магазину на		
	основі React-Redux		
	Програма та методика		
	тестування		
	Створення фронтенда для	1/6	Формат А4
	Онлайн магазину на		
	основі React-Redux		
	Керівництво користувача		

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**  
Факультет (інститут) Навчально-науковий інститут фізики, математики  
та інформаційних технологій  
(повна назва)  
Кафедра Інформаційних технологій та систем  
(повна назва)

**ЗАТВЕРДЖУЮ**

Завідувач кафедри ІТС

М.А. Семенов

(підпис)

(ініціали, прізвище)

“ ” 2022 р.

**ТЕХНІЧНЕ ЗАВДАННЯ**  
на виконання програмної розробки (ПР):

**Створення фронтенда для онлайн магазину на основі React-Redux**

**ПОГОДЖЕНО**

Керівник кваліфікаційної роботи

В.В. Матієвський

“ ” 2022р

**ВИКОНАВЕЦЬ**

Студент групи 4ПЗ

Д.О. Терещенко

“ ” 2022р

Полтава – 2022

## ЗМІСТ

ВСТУП	3
1. Підстави для розробки програми	3
2. Призначення розробки	3
3. Вимоги до програми	3
4. Вимоги до програмної документації	5
5. Етапи виконання розробки	5
6. Порядок контролю і прийому	6
7. Порядок внесення змін до технічного завдання, що затверджено.	6



## **ВСТУП**

Фронтенд для інтернет-магазину на основі React Redux.

Область застосування: Додаток призначений для використання на персональних комп'ютерах, які мають доступ до інтернету.

### **1. Підстави для розробки програми**

1.1. Перелік документів, на підставі яких ведеться розробка:

- Технічне завдання
- Пояснювальна записка

1.2. Організація: ЛНУ імені Тараса Шевченка.

1.3. Терміни розробки:

Початок 30 жовтня 2022 р.

Закінчення 15 травня 2023р

1.4. Умовне позначення: Фронтенд для інтернет-магазину на основі React Redux.

### **2. Призначення розробки**

2.1. Функціональне призначення: Здатність підключитися до додатку, отримувати інформацію про товари, обирати товари к покупці.

2.2. Експлуатаційне призначення: Додаток повинен експлуатуватися на локальному сервері.

### **3. Вимоги до програми**

#### **3.1. Вимоги до функціональних характеристик:**

3.1.1. Користувач системи повинен мати змогу виконувати наступні функції:

- Можливість переглянути список доступних товарів у магазини.
- Можливість додавати товар до кошика.
- Можливість переглядати сумарну вартість доданих товарів у кошику.
- Можливість взаємодії з додатком за допомогою браузера;
- Можливість відстежувати процес за допомогою графічного UI.

### **3.2 Вимоги до надійності:**

3.2.1. Додаток повинен коректно відображати дані про товари.

3.2.2. Додаток повинен коректно обчислювати вартість доданих товарів.

3.2.2. Розробник гарантує роботу додатку без збоїв та переналаштувань.

### **3.3. Умови експлуатації:**

3.3.1. Кінцевий користувач використовує персональний комп'ютер з Internet, та локально запускає додаток.

### **3.4. Вимоги до складу та параметрів технічних засобів:**

3.4.1. Процесор з тактовою частотою 2 ГГц і більше.

3.4.2. Відеокарта із 64 Ми відео пам'яті і вище.

3.4.3. 2 Габ оперативної пам'яті і вище.

3.4.5. Дисплей із розширенням 1366x768 і вище.

3.4.6. Клавіатура, комп'ютерна миша/тапчан.

3.4.7. Мережа Internet

### **3.5. Вимоги до інформаційної і програмної сумісності:**

3.5.1. Додаток запускається локально, відображення в браузері.

### **3.6. Вимоги до маркування та упаковки:**

3.6.1. Програма може поширюватися за допомогою флеш-картки.

3.6.2. Вимоги до маркування не пред'являються.

### **3.7. Вимоги до поширення, інсталяції та зберігання:**

3.7.1. Поширення додатку можливе через мережу інтернет, на зовнішніх накопичувачах тощо – на розсуд замовника.

3.7.2. Інсталяція сайту відбувається на ПК замовника, на якому встановлено відповідне ПЗ(прт-менеджер тощо).

3.7.2. Особливості зберігання відповідні до особливостей зберігання накопичувачів, на яких розміщена програма.

### **3.8. Спеціальні вимоги:**

3.8.1. Мова програмування – Javascript

3.8.2. Версія фреймворку React – 18.2.0 та вище.

3.8.3. Версія фреймворку Redux – 4.0.5 та вище.

#### 4.Вимоги до програмної документації

Перелік документів, що йдуть у комплекті з програмою:

4.1. Технічне завдання

4.2. Пояснювальна записка

4.3. Програма та методика тестування

4.4. Керівництво користувача

#### 5. Етапи виконання розробки

Етапи виконання можуть бути уточнюваннями згідно календарного плану робіт по узгодженню між замовником та виконавцем.

Таблиця 1.

Етапи виконання робіт

№	Етапи виконання роботи	Термін виконання та обсяг робіт	звітні матеріали
1	Аналіз розробки сайту та розробка першої версії. Аналіз вимог. Розробка структури. Попереднє тестування	3 місяці	Демо-версія сайту на локальній машині, що виконує всі основні функції та звітна документація
2	Коректування структури. Розробка допоміжних функцій. Розробка остаточної версії додатку та його опрацювання. Тестування.	2 місяці	Готовий додаток та звітна документація
3	Доопрацювання окремих модулів. Розробка звітних матеріалів згідно п.4 цього ТЗ	1 місяць	звітні матеріали згідно пункту 4

## **6. Порядок контролю і прийому**

6.1. Представлення дипломної роботи до попереднього захисту

6.2. Представлення дипломної роботи до захисту

## **7. Порядок внесення змін до технічного завдання, що затверджено.**

Дане технічне завдання може уточнюватися в процесі розробки ПР при узгодженні сторін з оформленням доповнень до ТЗ.

**МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ**  
**ДЗ «ЛУГАНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ**  
**ТАРАСА ШЕВЧЕНКА»**

Навчально-науковий інститут фізики, математики та інформаційних  
технологій

---

(назва факультету, інституту)

Кафедра інформаційних технологій та систем

---

(назва кафедри)

Пояснювальна записка  
до дипломного проєкту (роботи)  
БАКАЛАВРА  
(освітньо-кваліфікаційний рівень)

На тему: **СТВОРЕННЯ ФРОНТЕНДА ДЛЯ ОНЛАЙН МАГАЗИНУ НА**  
**ОСНОВІ REACT-REDUX**

Виконав: студент 4 курсу  
Напряму підготовки (спеціальності)  
121 «Інженерія програмного забезпечення»  
(шифр і назва напряму підготовки, спеціальності)  
Терещенко Д.О.

Керівник \_\_\_\_\_ Матієвський В. В.  
Рецензент \_\_\_\_\_ Козуб Ю.Г.

Полтава – 2023

## ЗМІСТ

ВСТУП .....	3
РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ФРОНТЕНДА САЙТІВ ..	5
1.1 Аналіз процесу веб-розробки .....	5
1.2 Основні види веб-розробки .....	9
1.3 Основні фронтедн фреймворки. Angular, React, Vue.js .....	13
Висновки к Розділу 1 .....	22
ГЛАВА 2. ОГЛЯД МОЖЛИВОСТЕЙ REACT & REDUX .....	23
2.1 Огляд React .....	23
2.2 Основні хуки React .....	27
2.3 Основні можливості та особливості бібліотеки Redux .....	34
2.4 Висновки к розділу 2 .....	37
ГЛАВА 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ФРОНТЕНД САЙТУ НА ОСНОВІ REACT-REDUX .....	39
3.1 Опис інтерфейсу сайту .....	39
3.2 Реалізація головної сторінки сайту .....	40
3.3 Реалізація праці із корзиною .....	47
3.4 Тестування .....	49
3.5 Висновки до розділу .....	51
ВИСНОВОК .....	52
ВИКОРИСТАНІ ДЖЕРЕЛА .....	53
ДОДАТКИ .....	54
Додаток А Лістинг компоненту main .....	54
Додаток Б Лістинг mainSlice .....	58

## ВСТУП

У сучасному світі інтернет-магазини стали невід'ємною частиною життя багатьох людей. Вони надають можливість купувати товари та послуги у будь-який час доби, не виходячи з дому. Саме тому розвиток онлайн платформ для комерції має перспективи розвитку.

Однак, щоб забезпечити зручність та комфорт користувачів, необхідно створити зручний та функціональний інтерфейс інтернет-магазину. У цій дипломній роботі розглядається створення frontend частини інтернет-магазину на основі React Redux. У роботі описуються основи розробки фронтенда сайтів, основні принципи роботи з React Redux, а також процес розробки інтернет-магазину на засадах React Redux.

Актуальність даної роботи зумовлена необхідністю створення зручного та функціонального інтерфейсу інтернет-магазину, який забезпечить комфортну роботу користувача із сайтом. Крім того, розробка інтерфейсу на основі React Redux дозволяє спростити процес розробки та забезпечити високу продуктивність інтерфейсу.

Об'єктом дослідження є процес створення frontend частини сайту.

Предметом дослідження є створення frontend частини сайту та розробка інтерфейсу за допомогою React Redux.

Мета роботи - створення фронтед частини для інтернет магазину за допомогою React Redux.

Завдання роботи:

Провести аналіз процесу веб-розробки, розглянути основні інструменти та поняття.

Розглянути основні типи веб-розробки: fullstack, frontend, backend

Розглянути основні фреймворки для розробки фронтед частини: Vue, React, Angular; зробити їх порівняльний аналіз.

Провести огляд можливостей хуків React

Ознайомитись з можливостями Redux

Зробити прототип фронтенд частини інтернет магазину на засадах React Redux.

Методи дослідження - теоретичні: аналіз наукових статей, книг, підручників та інших джерел, пов'язаних із темою роботи; експериментальні: тестування інтерфейсу, визначення продуктивності; емпіричні: порівняльний аналіз різних підходів до розробки фронтенду, аналіз їх переваг та недоліків.

Результатом роботи є функціональний інтерфейс, який забезпечує роботу користувача з інтернет-магазином.

Таким чином, дана робота має практичну значущість і може бути використана як посібник для розробки інтерфейсу інтернет-магазину на основі React Redux.



# РОЗДІЛ 1. ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ ФРОНТЕНДА САЙТІВ

## 1.1 Аналіз процесу веб-розробки

WEB-розробка – процедура створення WEB-додатка або WEB-сайту. Основними етапами цього процесу є такі заходи, як WEB-дизайн, верстка сторінок сайту, WEB-програмування на стороні сервера та клієнта, а також роботи зі конфігурування WEB-сервера.

В даний час мають право жити кілька поширених етапів у розробці WEB-сайту, а саме:

- проектування WEB-додатка або самого сайту, тобто збирання та подальший аналіз усіх вимог, вироблення технічного завдання, складання проекту інтерфейсів;
- вироблення концепції сайту з урахуванням креативу;
- розробка дизайнерської концепції інтернет-ресурсу;
- розробка макетів сторінок сайту;
- створення та виконання FLASH-елементів та мультимедіа;
- верстання шаблонів та сторінок;
- роботи з програмного забезпечення, як-то створення функціональних інструментів, або інтеграція в вже існуючу систему управління вмістом CMS;
- розміщення на сайті та оптимізація його текстових матеріалів;
- тестування сайту та внесення, за необхідності, коригування;
- запуск створеного проекту на громадському майданчику в Інтернеті;
- роботи з обслуговування вже чинного порталу або його програмної частини.

Різниця між веб-дизайном та веб-розробкою:

Веб-дизайнери:

- Створюють все, що бачить користувач на веб-сайті чи програмному продукті, включаючи всі візуальні елементи, елементи кольору, типографіки та зручності використання
- Працюють безпосередньо з клієнтом для створення дизайну або працюють в команді для розробки бачення клієнта
- Може знадобитися знання мови програмування або деякі навички кодування, або принаймні певний рівень знайомства з широко використовуваними мовами, такими як HTML, CSS, PHP і JavaScript
- Створюють каркаси та прототипи для перевірки дизайнерських ідей, пропонують свій внесок щодо дизайну логотипів, брендингу, посібників із стилю компанії

Веб-розробники:

- Пишуть код, який забезпечує функціонування веб-сайту, незалежно від того, чи працює він на фронтенді чи на бекенді (на стороні сервера)
  - Створюють або впроваджують проекти, які вимагає клієнт або створені командою дизайнерів
  - Потрібні експертні знання з різних мов програмування, включаючи HTML (HyperText Markup Language), CSS, PHP і JavaScript, а також знання інших мов (Ruby, C/C++, Python), фреймворків і бібліотек
  - Рідко створюють макети, вибирають типографіку чи колірну палітру
- Технічне завдання (ТЗ)

Його розробку для WEB-фахівців виконує, як правило, менеджер всього інтернет-проекту. Ну, а робота із самим замовником починається із заповнення брифу, де він викладає свої бажання щодо структури сайту та його візуалізації, уточнює помилки та недоробки, у разі наявності, у минулій версії WEB-сайту, наводячи свої приклади, як у його конкурентів. На підставі брифа, менеджер створює ТЗ, враховуючи при цьому наявні можливості дизайнерських та програмних інструментів. Сам такий етап закінчується лише після затвердження ТЗ клієнтом. Однак, слід зауважити,

що всі етапи проекту WEB-сайту досить сильно залежні від безлічі різних факторів, як, наприклад, величина обсягу інтернет-порталу, його функціональність, а також завдання для яких призначений створюваний інтернет-ресурс і багато іншого. Проте є і кілька етапів, які обов'язково присутні при плануванні абсолютно будь-якого майбутнього проекту.

#### Дизайн сторінок WEB-сайту: основних та типових

Будь-яка робота на сайті починається зі створення його дизайну, зазвичай використовуючи для цього графічний редактор. WEB-дизайнер створює, як правило, кілька таких варіантів, але у суворій відповідності до ТЗ. При цьому окремо розробляється дизайн «Головної» сторінки сайту, і далі – дизайн інших типових сторінок, як-от, наприклад: новини, статті, про нас, каталог. Власне, сам «дизайн» являє собою графічний файл, як листовий малюнок, що включає більш дрібні картинки у вигляді шарів в загальній картинці. При цьому фахівець обов'язково враховує всі обмеження стандарту HTML, тобто не виробляє дизайн, який неможливо буде згодом реалізувати стандартними HTML-засобами. Винятком є тільки Flash-дизайн.

Кількість самих ескізів та порядок їх пред'явлення замовнику заздалегідь обумовлюються менеджерами всього проекту, який виконує контроль запланованих термінів. Ще, також у великих WEB-фірмах у процесі бере участь і Арт-директор, який контролює якість виконання графіки. Цей етап так само, як і попередній, закінчується його твердженням у замовника.

#### Верстка сторінок і шаблонів у HTML

Затверджений клієнтом дизайн далі передається фахівцю-верстальнику, який «нарізає» графічне зображення на окремі картинки, з яких пізніше буде складена HTML-сторінка. У ході такої роботи створюється програмний код, який можна вже дивитися за допомогою будь-якого браузера (інтернет-браузера). Ну, а такі ці типові сторінки, згодом, будуть застосовуватися, як HTML-шаблони.

## Програмування

Після проведених вище згаданих заходів готові файли у форматі HTML передаються в роботи WEB-програмісту. Розробка програмного забезпечення інтернет-сайту цілком може виконуватися як «з самого нуля», так і на основі системи CMS, часто так званого «CMS-движка».

У разі застосування системи управління сайтом слід зазначити, що вона сама, в якомусь сенсі слова, вже готовий сайт, що включає блоки, що замінюються. Ну, а самого програміста, в такому випадку, вірніше називатиме «CMS-фахівцем», який має замінити існуючий стандартний шаблон, на новий оригінальний, розроблений на базі початкового WEB-дизайну, з урахуванням індивідуальних побажань замовника.

Під час розробки програмного забезпечення інтернет-сайту спеціалісту з CMS також встановлюються контрольні терміни проведення робіт.

## Тестування як заключний етап WEB-розробки інтернет-сайту

Сам такий процес цілком може містити в собі різні види перевірок, як-то, наприклад: зовнішній вигляд сторінки сайту зі збільшеними шрифтами, при різних розмірах браузерного вікна, або через відсутність Flash-плеєра, і багато іншого. Також використовується і тестування користувача, так зване - юзабіліті.

Виявлені помилки в роботі сайту відправляються для їх виправлення до тих пір, поки виконавець їх не усуне. І тут терміни роботи контролює той самий проектний менеджер. Хоча на етапі тестування ще залучають до роботи і самого дизайнера, щоб він здійснював авторський нагляд.

## Розміщення нового порталу в Інтернет-мережі

Файли розробленого WEB-сайту розміщують на сервері, наприклад провайдера, де здійснюють необхідні налаштування. Слід зазначити, що на такому етапі інтернет-сайт ще поки що закритий для широкого кола користувачів.

## 1.2 Основні види веб-розробки

Можна виділити такі види веб-розробки (рис. 1)

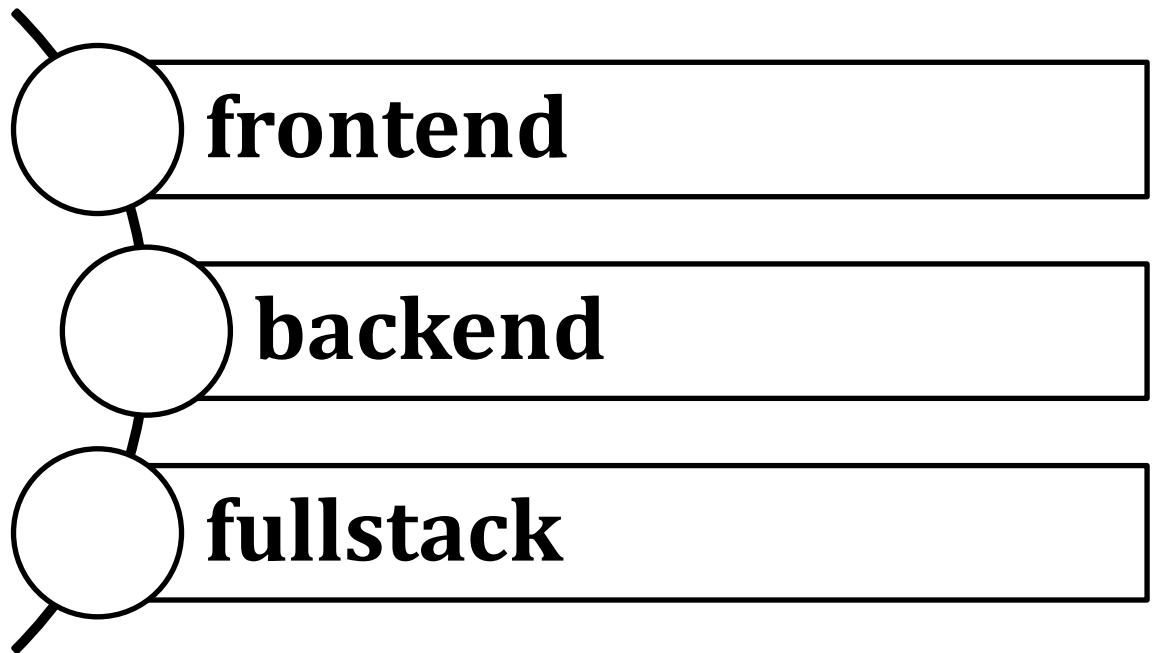


Рис. 1 Основні види веб-розробки

Фронтенд-фахівці займаються розробкою клієнтської частини, тобто відображенням даних. Програмісти, задіяні в цьому напрямі, взаємодіють з дизайнерами і відповідають за плавність анімації, правильність макета і всю фронтенд-частину, яку бачать користувачі. Саме frontend-сторона продукту взаємодіє з браузером. Фронтенд — це рівень представлення або частина веб-програми, яку бачить користувач. Наприклад, графічний інтерфейс користувача (GUI)[8].

Фронтенд закриває розрив між інтерфейсом та діями, які виконуються у фоновому режимі.

Це уможлиблює взаємодію користувача з серверною частиною, яка в іншому випадку була б утруднена або вимагала високого рівня спеціалізованих ноу-хау.

Фронтенд-розробник відповідає за реалізацію візуальних елементів, які бачать користувачі і з якими взаємодіють у веб-додатку. Зазвичай вони підтримуються внутрішніми веб-розробниками, які відповідають за логіку

серверних програм та інтеграцію роботи, що виконується зовнішніми розробниками

Фронтальна розробка передбачає «клієнтську» сторону веб-розробки. Тобто зазвичай інтерфейсна веб-розробка стосується тієї частини сайту, програми чи цифрового продукту, яку користувачі бачитимуть і з якою взаємодіятимуть. Таким чином, фронтенд-розробник відповідає за те, як цифровий продукт виглядає та «відчувається», тому їх часто також називають веб-дизайнерами.

Фронтальні веб-розробники зосереджуються на перекладі дизайну веб-сайту та візуальних ідей у код. Розробник зовнішнього програмного забезпечення використовує ідеї дизайну, створені іншими в командах веб-розробників, і програмує їх у реальність, діючи як міст між дизайном і технологією (рис. 2).

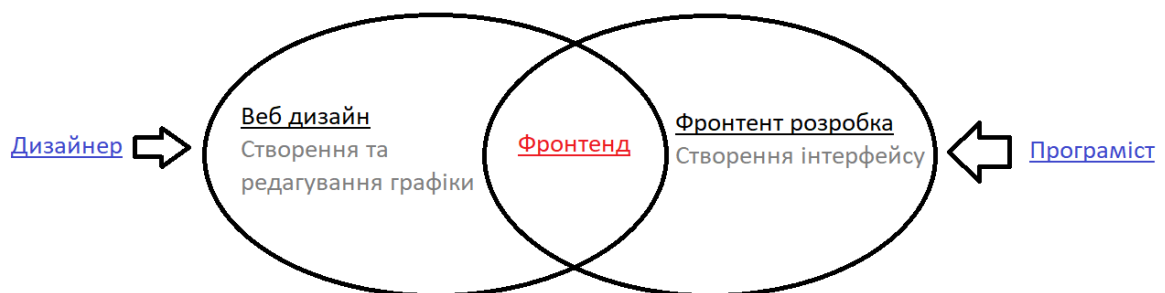


Рис. 2 - Структура фронтенду

Фронтальні розробники, як правило, повинні добре розуміти мови програмування, включаючи HTML, CSS і JavaScript, а також такі фреймворки, як React, Bootstrap, Backbone, AngularJS і EmberJS. Обов'язки фронтенд-розробника включатимуть створення адаптивних веб-сайтів (які добре виглядатимуть і функціонуватимуть на будь-якому пристрої), проведення тестування веб-сайтів і виправлення будь-яких помилок, виявлених у процесі веб-розробки, а також забезпечення того, щоб структура сайту відповідала найкращим практикам SEO.

На відміну від бекенд-розробки, існує ряд назв посад, які охоплюють різні набори навичок і рівні досвіду в інтересній розробці, зокрема:

1. Front-End розробник
2. Front-End інженер
3. Веб-розробник CSS/HTML
4. Інтерфейсний веб-дизайнер (зазвичай це означає роль, яка передбачає більше вимог до візуального дизайну та дизайну взаємодії)
5. Розробник інтерфейсу користувача (охоплює навички проектування взаємодії)
6. Мобільний веб-розробник
7. Front-End SEO Expert (зазвичай позначає розробника з досвідом впровадження стратегії SEO)
8. Експерт із доступності інтерфейсу
9. Front-End Dev Ops
10. Front-End QA (зокрема модульне тестування, функціональне тестування, тестування користувача та тестування A/B)

Розробник фронтенду веб-додатків може використовувати деякі програмні інструменти, найпопулярніші з яких це: HTML; CSS; JavaScript; React; Vue.js; Angular;

HTML - це мова розмітки для сторінок веб-браузеру.

CSS - це каскадні таблиці, які можуть змінювати стилі елементів, які вже добавлені в HTML-документах.

JavaScript - це мова програмування, яка підтримується більшістю веб-браузерів, і є найбільш поширеною мовою програмування для розробки фронтенду веб-додатків. Ця мова програмування використовує DOM-дерево для роботи з елементами веб-сторінки. DOM-дерево - це інтерфейс для роботи зі змістом HTML сторінок.

JSON - це текстовий формат, оснований на мові програмування JavaScript. Цей формат є найбільш поширеним при роботі з даними в веб-додатках.

Приклад JSON-формату:

```
{ "price": 499,  
  "unit": "KG",  
  "unit_count": 1 }
```

React, Angular, Vue - це фреймворки, за допомогою яких можна значно спростити, прискорити, та оптимізувати розробку веб-додатків, зокрема веб-сайтів. Для цієї бакалаврської роботи було обрано фреймворк React, більш детально цей фреймворк буде розглянуто в наступному розділі.

Бекенд-фахівець працює із серверною частиною, тобто логікою, прихованою від користувача. У цьому разі мова може йти як про автентифікації користувачів, так і про балансування навантаження на сервер. Бекенд-фахівці можуть взаємодіяти з адміністраторами, оскільки велике значення має працездатність та швидкодія сервера. Програміст працює backend-частиною проекту, до якої звертається frontend або інший клієнт.

Якщо фронт-енд розробники відповідають за те, як виглядає цифровий продукт, то бекенд-розробники зосереджуються на тому, як він працює. Back-End-розробник створює основну структуру веб-сайту перед тим, як підтримувати його та гарантувати, що він працює належним чином, включаючи взаємодію з базою даних, автентифікацію користувачів, конфігурацію сервера, мережі та хостингу та бізнес-логіку. Працюючи за лаштунками — або на стороні сервера — Back End-розробники стурбовані системами та структурами, які дозволяють комп'ютерним програмам працювати як потрібно.

Основна відповідальність Back-End полягає в тому, щоб забезпечити функціональність сайту, включаючи його оперативність і швидкість. Для цього розробники Back-End повинні знати, як створювати сервери з сучасними фреймворками (при розробці користувацьких API та



обслуговуванні статичних веб-сайтів і файлів), а також як керувати базами даних і даними на веб-сервері.

Як правило, Back-End розробники використовують мови програмування на стороні сервера, зокрема PHP, Ruby та Python, а також такі інструменти, як MySQL, Oracle та Git.

Fullstack-фахівець – універсальний «солдат», який відповідає за всі етапи реалізації проекту. Фулстек-програмісти поєднують обов'язки бекенд-та фронтенд-розробників. У деяких випадках fullstack-фахівці можуть виконувати функції системних адміністраторів та дизайнерів.

Fullstack Developer — це людина, яка знайома з розробкою як на фронтенді, так і на бекенді. Розробники FullStack зазвичай розуміють широкий спектр мов програмування, і через їхню універсальність їм може бути надано більшу роль лідера в проектах, ніж розробникам, які спеціалізуються вузько. Вони є універсалами, вміють працювати з кожною частиною розробки.

### **1.3 Основні фронтенд фреймворки. Angular, React, Vue.js**

Веб-розробники завжди перебувають на роздоріжжі, де їм доводиться вибирати серед низки фреймворків розробки та вибрати одну для свого проекту. Це поширена тема дебатів серед розробників щодо того, як вибрати фреймворк для свого наступного великого проекту. Деякі фреймворки, які стали найпопулярнішими серед розробників і викликають дилему, це ReactJS, VueJS і Angular.

Проста різниця між цими трьома полягає в тому, що React — це бібліотека інтерфейсу користувача, а Vue — прогресивний фреймворк. Однак Angular — це повноцінний інтерфейсний фреймворк. Згідно з опитуванням StackOverflow 2022, React є улюбленим фреймворком 42,62% розробників, Angular — 20,39%, а Vue — 18,82% розробників (рис. 3).



рис 3 - Рейтинг фреймворків StackOverflow 2022

## Angular

Angular - це інтерфейсний JavaScript-фреймворк із відкритим вихідним кодом, який найкраще використовувати для складних великомасштабних. Angular належить до стеку MEAN, одного з найпопулярніших технологічних стеків для стартапів. Це середовище розробки веб-застосунків засноване на Typescript. Angular здатний і переважно використовується для створення односторінкових веб-додатків або SPA.

### Загальна інформація

- розроблено Google;
- вперше випущений у 2010 році;
- фреймворк JavaScript на основі TypeScript;
- швидко зарекомендувала себе як основна технологія;
- в даний час близько 200 тисяч діючих веб-сайтів використовують Angular.

### Основні характеристики

- пропонує надійний набір компонентів, що спрощує написання, зміну та використання коду;
- скорочує час початкового завантаження веб-сторінки за рахунок поділу завдань на логічні фрагменти через структуру MVC;

- модель MVC допомагає зменшити кількість запитів у фоновому режимі, дозволяючи поділ уявлень;
- Angular пропонує повністю налаштований дизайн, надаючи розробникам та дизайнерам більше можливостей;
- пропонує безшовну сторонню інтеграцію для покращення функціональності продукту;
- «випереджальний компілятор» забезпечує швидке завантаження та підвищену безпеку;
- під час розробки Angular компілює HTML і TypeScript JavaScript, що прискорює компіляцію коду навіть до того, як браузер завантажить ваш веб-додаток;
- за допомогою Angular впровадження залежностей можна використовувати як компоненти, що розділяють зовнішні елементи, що робить їх повторно використовуваними, більш простими в управлінні та тестуванні.

#### Плюси Angular

- пропонує хорошу структурованість;
- проста двостороння прив'язка даних;
- архітектура MVC;
- початкове скорочення часу завантаження веб-сторінки;
- вбудована модульна система;
- велика спільнота та екосистема.

#### Мінуси Angular

- обмежені можливості SEO;
- слабка доступність для пошукових роботів;
- складний для вивчення;
- складні SPA повільні та неефективні

Існують випадки, коли цей фреймворк найкраще підходить.

Angular має сенс використовувати, якщо планується:

- розробка прогресивного веб-додатку (PWA);
- великий корпоративний додаток зі складною інфраструктурою;
- веб-сайти з динамічним контентом;
- оновлення дизайну веб-додатку;
- покращення користувацького досвіду великих веб-сайтів за рахунок редизайну.

## React

React - це інтерфейсна бібліотека JavaScript з відкритим вихідним кодом, що використовується для розробки інтерфейсів користувача. Стек MERN є одним з найпопулярніших стеків для створення високопродуктивних бізнес-додатків, включає React, орієнтований на створення інтерактивних користувацьких інтерфейсів.

Інтерфейс користувача грає вирішальну роль у веб-розробці, коли користувач бачить додаток і взаємодіє з ним через нього. Завдяки своїй простоті та орієнтації на взаємодію з користувачем React став одним із найпопулярніших фреймворків

## Загальна інформація

- розроблено Facebook;
- реліз у 2013 році;
- зовнішня бібліотека, яка використовується для створення компонентів інтерфейсу користувача з відстеженням стану і багаторазового перевикористання;
- із загальної кількості веб-сайтів розроблених з використанням React, 59,3% веб-сайтів використовують React v16, 19,1% використовують v15, 17,5% використовують React v0 та 4,1% використовують React 17;
- створює інтерфейси інтерфейсу користувача спеціально для SPA;
- приблизно 2 мільйони веб-сайтів використовують React.

## Основні характеристики

Virtual DOM пропонує ефективність та оптимізовану продуктивність для складних програм, що вимагають частих змін. Віртуальна модель DOM дозволяє вносити зміни та оновлювати лише ту частину, де це потрібно. Це змушує браузер повторно відображати лише невелику частину сторінки, а чи не всю її. React підтримує зв'язування та перемішування дерева DOM. Ці функції використовуються зниження навантаження на ресурси кінцевих користувачів. React пропонує найкращий контроль над проектом, підтримуючи односторонню прив'язку даних. React легко тестується та відстежується, що покращує весь процес розробки.

#### Плюси React

Ключові переваги, які пропонує React та завдяки яким він виділяється:

- підвищує швидкість процесу розробки, дозволяючи реагує веб-розробникам писати окремі частини як на стороні клієнта, так і на стороні сервера без зміни логіки програми;
- код React гнучкий завдяки своїй модульній структурі, заощаджуючи багато часу та коштів;
- код React легше підтримувати;
- пропонує високу продуктивність складних програм;
- React пропонує розробку мобільних програм, підтримуючи мобільні нативні програми для платформ Android та iOS.

#### Мінуси React [9]

- React не реалізує MVC, що підштовхує до використання додаткових бібліотек для реалізації станів та моделей;
- постійні оновлення React та використання супутніх бібліотек для його підтримки призводять до поганої документації;
- швидке прискорення технологій React призводить до написання неправильних інструкцій через брак часу;
- спеціалістам з React іноді складно прискорити постійні оновлення через швидкий розвиток технологій React;

– останнім часом Facebook випускає дуже багато оновлень, через що старіють деякі інструменти.

Чистий React необхідний не для кожного проекту. Але при використанні разом із Redux React стає потужним інструментом. Якщо програма досить проста, є можливість писати на JavaScript. React найкраще підходить для проектів, які включають безліч компонентів з активним або неактивним станами, елементами навігації, розгорнутими або згорнутими розділами, динамічними введеннями, активними або вимкненими кнопками, а також правами входу та доступу користувачів. Керування такими станами, що змінюються, і подання користувачам різних уявлень залежно від стану з React не викликає проблем. Якщо програма має шанс зрости в масштабі та обсязі, то React найкраще підійде, оскільки його декларативний характер компонентів дозволяє легко працювати з такими складними структурами.

## Vue

Vue - це інтерфейсний JavaScript-фреймворк Model-View-View-Model з відкритим вихідним кодом, який використовується для розробки інтерфейсів користувача і односторінкових додатків (SPA). Цей JavaScript-фреймворк доступний, високопродуктивний та універсальний. Vue сприяє оптимальній та динамічній взаємодії з користувачем в Інтернеті.

## Загальна інформація

- розроблено Еваном Ю. у 2014 році;
- реліз у 2014 році;
- Vue створений для поступового впровадження користувачами;
- понад 1 мільйон веб-сайтів, створених за допомогою Vue.

## Основні характеристики

Vue найкраще підходить для вирішення короткострокових проблем. Він може легко інтегруватись з існуючими блоками коду. Vue - виключно швидкий інструмент і володіє висококласними функціями, простою кривою навчання, високоефективним, швидким і складним односторінковим

додатком, що робить його одним з найпопулярніших JS-фреймворків згідно зі статистикою StackOverflow.

Плюси Vue:

- розмір фреймворку досить малий, що спрощує завантаження та встановлення;
- вимагає меншої оптимізації від розробників Vue через свій розмір;
- через розмір фреймворку він стимулює розробку веб-додатків Vue.js і дозволяє експертам Vue відокремлювати віртуальну модель DOM від шаблону від компілятора;
- віртуальний DOM у Vue дозволяє оновлювати елементи на веб-сторінці без рендерингу всієї DOM;
- Vue забезпечує легкий доступ до всіх своїх функцій, які підтримують налаштування;
- Vue сумісний та гнучкий, оскільки Vue використовує HTML для рендерингу об'єктів та систему шаблонів для безшовної інтеграції з існуючими програмами;
- Vue можна використовувати повторно, якщо він правильно налаштований;
- кодова база не обтяжується в міру масштабування програми.

Мінуси Vue

- Vue порівняно новий, і йому знадобиться час, щоб розширити його співтовариство;
- ринок Vue не так широко популярний у порівнянні з React та Angular;
- потрібен час, щоб побачити високий попит на вакансії для кваліфікованих розробників Vue;
- Vue не можна використовувати для великомасштабних проєктів, оскільки йому не вистачає масштабованості;
- Vue має менше плагінів у порівнянні з React та Angular;

– програми Vue мають проблеми з продуктивністю зі старими версіями мобільних браузерів Safari та iOS.

Коли використовувати Vue

Фреймворк Vue має сенс використовувати, якщо:

- у додатку повно анімації та інтерактивних елементів;
- додаток вимагає безшовної інтеграції з кількома програмами;
- необхідно створити прототип без просунутих навичок;
- потрібно швидко запустити MVP.

Продуктивність. Порівняльний аналіз.

Результати тесту JS Framework Benchmark [6] показують, що всі фреймворки досить добре працюють у більшості тестів, таких як створення або додавання рядків до таблиці (рис 4), час завантаження програми з використанням інструменту Lighthouse (рис 5), використання пам'яті додатком (рис 6).

Duration in milliseconds  $\pm$  95% confidence interval (Slowdown = Duration / Fastest)

Name Duration for...	vue- v3.2.47	angular- v15.0.1	react- v18.2.0
Implementation notes			
Implementation link	<a href="#">code</a>	<a href="#">code</a>	<a href="#">code</a>
<a href="#">create rows</a> creating 1,000 rows (5 warmup runs).	42.0 $\pm$ 0.7 (1.18)	46.7 $\pm$ 0.5 (1.29)	47.5 $\pm$ 0.6 (1.34)
<a href="#">replace all rows</a> updating all 1,000 rows (5 warmup runs).	44.8 $\pm$ 0.5 (1.16)	48.2 $\pm$ 0.6 (1.25)	52.8 $\pm$ 0.3 (1.37)
<a href="#">partial update</a> updating every 10th row for 1,000 rows (3 warmup runs). 16 x CPU slowdown.	103.5 $\pm$ 3.4 (1.18)	90.7 $\pm$ 2.5 (1.10)	128.1 $\pm$ 3.0 (1.46)
<a href="#">select row</a> highlighting a selected row. (5 warmup runs). 16 x CPU slowdown.	22.1 $\pm$ 1.3 (2.31)	15.6 $\pm$ 1.0 (1.64)	39.3 $\pm$ 0.6 (4.12)
<a href="#">swap rows</a> swap 2 rows for table with 1,000 rows. (5 warmup runs). 4 x CPU slowdown.	29.0 $\pm$ 0.8 (1.17)	166.0 $\pm$ 1.0 (6.71)	163.5 $\pm$ 0.7 (6.61)
<a href="#">remove row</a> removing one row. (5 warmup runs). 4 x CPU slowdown.	45.8 $\pm$ 1.1 (1.20)	42.3 $\pm$ 1.2 (1.10)	47.7 $\pm$ 1.3 (1.25)
<a href="#">create many rows</a> creating 10,000 rows. (5 warmup runs with 1k rows).	475.3 $\pm$ 1.5 (1.19)	474.2 $\pm$ 1.9 (1.18)	661.5 $\pm$ 2.4 (1.65)
<a href="#">append rows to large table</a> appending 1,000 to a table of 10,000 rows. 2 x CPU slowdown.	94.7 $\pm$ 0.8 (1.14)	101.7 $\pm$ 0.8 (1.23)	115.0 $\pm$ 0.7 (1.39)
<a href="#">clear rows</a> clearing a table with 1,000 rows. 8 x CPU slowdown. (5 warmup runs).	34.7 $\pm$ 1.2 (1.21)	61.1 $\pm$ 1.7 (2.13)	37.9 $\pm$ 1.0 (1.32)
<a href="#">geometric mean</a> of all factors in the table	1.27	1.59	1.87
compare: Green means significantly faster, red significantly slower	<a href="#">compare</a>	<a href="#">compare</a>	<a href="#">compare</a>

Рис. 4 - Час, необхідний для обробки операцій із таблицями



### Startup metrics (lighthouse with mobile simulation)

Name	vue-v3.2.47	angular-v15.0.1	react-v18.2.0
<b>consistently interactive</b> a pessimistic TTI - when the CPU and network are both definitely very idle. (no more CPU tasks over 50ms)	2,102.8 ± 49.7 (1.17)	2,630.9 ± 1.4 (1.46)	2,552.7 ± 1.5 (1.42)
<b>total kilobyte weight</b> network transfer cost (post-compression) of all the resources loaded into the page.	197.0 ± 0.0 (1.38)	282.8 ± 0.0 (1.99)	280.9 ± 0.0 (1.97)
<b>geometric mean</b> of all factors in the table	1.27	1.70	1.67

Рис. 5 - Час завантаження програми

### Memory allocation in MBs ± 95% confidence interval

Name	vue-v3.2.47	angular-v15.0.1	react-v18.2.0
<b>ready memory</b> Memory usage after page load.	0.9 (1.41)	1.6 (2.60)	1.1 (1.88)
<b>run memory</b> Memory usage after adding 1,000 rows.	3.7 (2.09)	4.7 (2.62)	5.0 (2.78)
<b>update every 10th row for 1k rows (5 cycles)</b> Memory usage after clicking update every 10th row 5 times	3.8 (1.95)	4.7 (2.45)	5.5 (2.83)
<b>creating/clearing 1k rows (5 cycles)</b> Memory usage after creating and clearing 1000 rows 5 times	1.2 (1.73)	2.3 (3.35)	2.0 (2.83)
<b>run memory 10k</b> Memory usage after adding 10,000 rows.	27.7 (2.42)	29.8 (2.61)	36.2 (3.17)
<b>geometric mean</b> of all factors in the table	1.89	2.71	2.66

Рис. 6 - Використання пам'яті програмою

Як можна помітити, Vue значно повільніше, ніж Angular та React при виборі та підсвічуванні рядків. З іншого боку, Angular і React не дуже ефективні при зміні порядку рядків. Це єдині істотні відмінності в тестах рендерингу, які в більшості випадків не дають результатів, що різняться. Оскільки вибір рядків є більш поширеною функцією, ніж їх заміна, можна

судити, що цей тест ставить Vue на третє місце після Angular і React, які займають лідируючу позицію.

Що стосується пам'яті та часу завантаження, React та Vue показують дуже хороші результати, але Angular працює трохи повільніше. Angular може знадобитися 150 мс для завантаження базового скрипта і потребує більше пам'яті для запуску.

## **Висновки к розділу 1**

Отже, ми розглянули концепцію веб-розробки. Провели загальний аналіз процесу веб-розробки, розібрали основні поняття. Ознайомилися із завданнями та етапами веб-розробки.

Розглянули три типи веб-розробки: frontend; backend; fullstack. Ознайомилися з основними завданнями та особливостями кожного типу.

Дослідили, що фронтенд займається клієнтською частиною програми. Бекенд відповідає за серверну частину програми. Фуллстек відповідає і за фронтенд, і за бекенд одночасно.

Розібрали основні інструменти, які застосовують розробники при роботі з фронтендом: HTML; CSS; Javascript тощо.

Розглянули основні фронтенди фреймворки: React; Angular; Vue.

Було зроблено аналіз кожного з цих фреймворків, вивчено основні особливості, переваги та недоліки при застосуванні, порівняли сфери застосування. Розглянуто доречні умови для вибору конкретного фреймворку для розробки фронтенду.

Зроблено порівняльний аналіз фреймворків. Виходячи з цього ми можемо зробити висновок, що кожен із фреймворків має свої продуктивні переваги та недоліки при певних операціях додатку.

## ГЛАВА 2. ОГЛЯД МОЖЛИВОСТЕЙ REACT & REDUX

### 2.1 Огляд React

React - це веб-фреймворк для розробки веб-додатків. Був розроблений в 2013 році Джорданом Валке, розробником компанії Facebook, перший реліз відбувся 29 травня 2013. Того ж місяця код бібліотеки був відкритим.

В лютому 2015 був розроблений фреймворк ReactNative, який базується на платформі React. Цей додаток дозволяє розробляти мобільні додатки, зокрема на базі Android та iOS.

Треба зауважити, що Реакт не обробляє даних, та не керує станом своїх компонентів. В цілому, Реакт використовується для відображення елементів веб-сторінки, але не для оперування ними.

Код React складається з сутностей, які називаються компонентами. Ці компоненти є модульними та багаторазовими. Програми React зазвичай складаються з багатьох рівнів компонентів. Компоненти рендеряться в кореневий елемент у DOM за допомогою бібліотеки React DOM. Під час візуалізації компонента значення передаються між компонентами через props, тобто властивості.. Внутрішні значення компонента називаються його станом.

Двома основними способами оголошення компонентів у React є компоненти функції та компоненти класу. Наразі, тенденція в веб-розробці є такою, що функціональний варіант розробки компонентів домінує над класовим.

За допомогою Реакт, можливо створювати односторінкові веб-додатки. Односторінкова програма – це програма, яка завантажує одну сторінку HTML і всі необхідні ресурси (наприклад, JavaScript і CSS), необхідні для роботи програми. Будь-яка взаємодія зі сторінкою або наступними сторінками не потребує повернення до сервера, тобто сторінка не перезавантажується.

Хоча ми можемо створити односторінкову програму в React, це не обов'язково. React також можна використовувати для покращення невеликих частин існуючих веб-сайтів додатковою інтерактивністю. Код, написаний у React, може мирно співіснувати з розміткою, відтвореною на сервері чимось на кшталт PHP, або з іншими клієнтськими бібліотеками.

На зорі Інтернету більшість сайтів складалася із серії сторінок, за якими користувачі могли переміщатися шляхом запитів та відкриття окремих файлів. Розташування поточного файлу або ресурсу відображалось в адресному рядку браузера. Кнопки переміщення вперед та назад працювали цілком очікуваним чином. Вміст закладок, націлених на глибини сайту, дозволяв користувачам зберігати посилання на конкретний файл, який міг бути завантажений за запитом користувача. На сайті зі сторінковою організацією, або із серверним поданням, браузерна навігація та історії переглядів просто працювали, як і планувалося.

В односторінковому додатку (single-page application, SPA) описані функції стали проблематичними. Нагадуємо, що в цьому додатку все відбувається на одній сторінці. Засоби мови JavaScript завантажують інформацію та вносять зміни до інтерфейсу користувача. Такі властивості, як історія переглядів, закладки та кнопки переміщень вперед і назад, не працюватимуть без рішень, пов'язаних із маршрутизацією. Це процес визначення кінцевих точок запитів ваших клієнтів. Ці точки працюють у зв'язку з розташуванням та об'єктами історії переглядів браузера. Вони використовуються для ідентифікації запитаного вмісту, щоб засоби JavaScript могли завантажити та вивести на екран відповідний інтерфейс користувача.

На відміну від Angular, Ember або Backbone, React не постачається зі стандартним маршрутизатором. Усвідомлюючи важливість рішень щодо маршрутизації, інженери Майкл Джексон (Michael Jackson) та Райан Флоренс (Ryan Florence) створили маршрутизатор, який назвали просто React Router.

В своїй роботі, Реакт використовує віртуальне DOM-дерево. React створює в пам'яті кеш структури даних, обчислює отримані відмінності, а потім ефективно оновлює DOM, що відображається у браузері. Цей процес називається звіркою. Це дозволяє програмісту писати код так, ніби вся сторінка рендериться при кожній зміні, тоді як React рендерить лише компоненти, які фактично змінюються. Цей вибіркового рендеринг забезпечує значне підвищення продуктивності.[1]

Однією з основних переваг роботи з React є обробка даних у деревах компонентів. Методи, якими можна скористатися при цьому, дозволяють суттєво полегшити ваше життя у довгостроковій перспективі. Якщо вдасться керувати даними з одного місця і створювати інтерфейс користувача на їх основі, то наші програми буде легше зрозуміти і масштабувати.

JavaScript відноситься до мов зі слабкою типізацією; це означає, що типи даних значень змінних можуть змінюватися. Наприклад, можна спочатку встановити для змінної JavaScript рядкове значення, а потім змінити значення на масив, і з боку мови не буде жодних заперечень. Неефективне керування типами змінних може негативно позначитися на часі, що витрачається на налагодження додатків.

Компоненти React надають спосіб вказівки та перевірки типів властивостей. Використання цієї функції суттєво скоротить час, що витрачається на налагодження програм. При наданні невірних типів властивостей видаватиметься попередження; воно допоможе виявити помилки, які за інших обставин можуть залишитися непоміченими.

JSX, або розширення синтаксису JavaScript, є розширенням синтаксису мови JavaScript. Подібний за зовнішнім виглядом до HTML, JSX надає спосіб структурувати відтворення компонентів за допомогою синтаксису, знайомого багатьом розробникам. [9]

Компоненти React зазвичай пишуться за допомогою JSX, хоча це не обов'язково (компоненти також можуть бути написані за допомогою чистого JavaScript). [4]

Команда Facebook, що займається розробкою React, випустила JSX одночасно з React, щоб надати лаконічний синтаксис для створення складних дерев DOM з атрибутами. Крім того, розробники сподівалися зробити код React читати так само легко, як код HTML і XML. У JSX тип елемента вказується за допомогою тега. Його атрибути є властивостями. Дочірні елементи до створюваного елемента можуть додаватися між тегами, що відкривають і закривають. Як дочірні можна додавати й інші JSX-елементи. Під час роботи з невпорядкованим списком елементи дочірніх записів списку додаються до нього за допомогою тегів JSX. Виглядає це дуже схоже на HTML:

```
<ul>
  <li>1 lb Salmon</li>
  <li>1 cup Pine Nuts</li>
  <li>2 cups Butter Lettuce</li>
  <li>1 Yellow Squash</li>
  <li>1/2 cup Olive Oil</li>
  <li>3 cloves of Garlic</li>
</ul>
```

JSX дозволяє додавати компоненти до інших компонентів як дочірні. Наприклад, всередині `IngredientsList` можна кілька разів відобразити ще один компонент, названий `Ingredient`:

```
<IngredientsList>
  <Ingredient />
  <Ingredient />
  <Ingredient />
</IngredientsList>
```

Візуалізація на стороні сервера (SSR) відноситься до процесу рендерингу клієнтської програми JavaScript на сервері, а не в браузері. Це може покращити продуктивність програми, особливо для користувачів із повільним з'єднанням.

За допомогою SSR початковий HTML, який надсилається клієнту, включає повністю відтворений інтерфейс користувача програми. Це дозволяє браузеру клієнта негайно відображати інтерфейс користувача, замість того, щоб чекати, поки JavaScript завантажиться та запуститься перед відтворенням інтерфейсу користувача.

React підтримує SSR, що дозволяє розробникам відтворювати компоненти React на сервері та надсилати отриманий HTML клієнту. Це може бути корисним для покращення продуктивності програми, а також для оптимізації пошукової системи.

## 2.2 Основні хуки React

React Hooks - це функції, які дозволяють розробникам «підключатися» до стану React і функцій життєвого циклу з функціональних компонентів. Примітно, що хуки не працюють всередині класів — вони дозволяють розробникам використовувати більше функцій React без класів.[2]

React надає кілька вбудованих хуків, таких як `useState`, `useContext`, `useReducer`, `useMemo` та `useEffect`. `useState` та `useEffect`, які найчастіше використовуються, призначені для контролю стану та контролю змін відповідно.

### Правила хуків

Є два правила хуків, які описують характерні моделі коду, на які спираються хуки:[3]

«Тільки виклик хуків на верхньому рівні» — не викликайте хуки з внутрішніх циклів, умов або вкладених інструкцій, щоб хуки викликалися в тому самому порядку кожного рендеру.

«Тільки виклик хуків із функцій React» — не викликайте хуки зі звичайних функцій JavaScript, щоб логіка стану залишалася в компоненті.

### Хук `useState()`.

Приклад: `const [state, setState] = useState(initialState);`

Під час початкового рендерингу стан (`state`) збігається зі значенням, переданим як перший аргумент (`initialState`).

Функція `setState` використовується для оновлення стану. Він приймає нове значення стану та ставить у чергу повторне відтворення компонента.

### Хук `useContext()`:

Приклад: `const context = useContext(Context);`

Приймає об'єкт контексту (значення, що повертається з `React.createContext`) і повертає поточне значення контексту, надане найближчим постачальником контексту для даного контексту.

Коли постачальник оновлюється, цей хук ініціює повторне відтворення з останнім значенням контексту.

### Хук `useEffect()`:

Приклад:

```
useEffect(
  () => {
    const subscription = props.source.subscribe();
    return () => {
      subscription.unsubscribe();
    };
  },
  [props.source],
);
```

Приймає функцію, яка містить імперативний, можливо ефективний код.



Функція, передана `useEffect`, запуститься після того, як візуалізація буде зафіксована на екрані. За замовчуванням ефекти запускаються після кожного завершеного візуалізації, але ви можете запустити їх лише після зміни певних значень. Передача порожнього масиву `[]` вхідних даних повідомляє React, що ваш ефект не залежить від жодних значень компонента, тому цей ефект працюватиме лише під час монтування та очищатиметься під час відключення; він не буде працювати на оновленнях.

### Хук `useReducer()`:

Приклад: `const [state, dispatch] = useReducer(reducer, initialState);`

Альтернатива `useState`. Приймає редьюсер типу (стан, дія) => newState і повертає поточний стан у поєднанні з методом відправки.

### Хук `useMemo()`:

Приклад: `const memoizedValue = useMemo(() => computeExpensiveValue(a, b), [a, b]);`

Повертає мемоізоване значення.

Передайте функцію і масив вхідних даних. `useMemo` повторно обчислить мемоізоване значення лише тоді, коли один із вхідних даних зміниться. Ця оптимізація допомагає уникнути дорогих обчислень на кожному рендері.

Якщо масив не надано, нове значення буде обчислюватися щоразу, коли новий екземпляр функції передається як перший аргумент. (З вбудованою функцією, на кожному рендері.)

### Хук `useRef()`:

Приклад: `const refContainer = useRef(initialValue);`

Цей хук повертає змінний об'єкт `ref`, властивість якого ініціалізовано переданим аргументом (`initialValue`). Повернений об'єкт зберігатиметься протягом повного життя компонента.

### Хук `useCallback()`:

Приклад: `const memoizedCallback = useCallback(  
 () => {`

```
doSomething(a, b);  
},  
[a, b],  
);
```

Повертає мемоізований зворотний виклик.

Передається зворотній виклик і масив вхідних даних. `useCallback` поверне мемоізовану версію зворотного виклику, яка змінюється, лише якщо змінився один із вхідних даних. Це корисно під час передачі зворотних викликів оптимізованим дочірнім компонентам, які покладаються на рівність посилань, щоб запобігти непотрібним візуалізаціям (наприклад, `shouldComponentUpdate`).

Коли `ReactDOM.render()` викликається повторно для компонента, React представляє новий стан інтерфейсу користувача у Virtual DOM і визначає, які частини (якщо такі є) живого DOM потрібно змінити.

В Angular наявна система двосторонньої зв'язки даних. Це, наприклад, виражається в тому, що зміни у формі елемента призводять до автоматичного оновлення стану додатка. Це ускладнює відладку і є великим мінусом даного фреймворка. При такому підході, якщо щось йде не так, програміст не може абсолютно точно знати про те, що саме стало причиною зміни стану додатка.

В React, з іншої сторони, використовується одностороння зв'язка даних. Це — великий плюс даної бібліотеки, він виражається в тому, що програміст завжди точно знає про те, що призвело до зміни стану додатків. Подібний підхід до зв'язку даних значно покращує відладку додатків.

Методи життєвого циклу для компонентів використовують форму перехоплення, яка дозволяє виконувати код у заданих точках протягом життя компонента.

Методи життєвого циклу компонентів:

**ShouldComponentUpdate** дозволяє розробнику запобігти непотрібному повторному рендерингу компонента, повертаючи `false`, якщо рендеринг не потрібний.

**componentDidMount** викликається, коли компонент «змонтовано» (компонент було створено в інтерфейсі користувача, часто шляхом його асоціювання з вузлом DOM). Це зазвичай використовується для запуску завантаження даних із віддаленого джерела через API.

**componentWillUnmount** викликається безпосередньо перед тим, як компонент буде розірвано або "розмонтовано". Це зазвичай використовується для очищення залежностей компонента, які потребують ресурсів.

**render** є найважливішим методом життєвого циклу та єдиним необхідним у будь-якому компоненті. Зазвичай він викликається кожного разу, коли оновлюється стан компонента, що має відображатися в інтерфейсі користувача.

**Життєвий цикл установки** складається з методів, що викликаються при встановленні компонента або видалення з екрана. Іншими словами, ці методи дозволяють спочатку встановлювати стан, здійснювати виклики API, запускати та зупиняти таймери, маніпулювати відображуваною DOM, ініціалізувати бібліотеки сторонніх розробників і т. д. Ці методи дозволяють вводити в обіг JavaScript, щоб допомогти ініціалізувати та знищувати компонент.

Життєвий цикл установки відрізняється залежно від того, як створюються компоненти: за допомогою синтаксису ES6 або методу `React.createClass`. У разі використання останнього для отримання властивостей компонента спочатку викликається метод `getDefaultProps`. Потім для ініціалізації стану викликається метод `getInitialState`.

Класи ES6 не мають цих методів. Замість них виходять властивості, які використовуються за умовчанням, які у вигляді аргументів передаються конструктору. Стан ініціалізується у конструкторі. Доступ до властивостей

є як у конструкторів класу ES6, так і методу `getInitialState`, і, якщо потрібно, їх можна задіяти, щоб допомогти визначити вихідний стан.

Після отримання властивостей та ініціалізації стану викликається метод `componentWillMount`. Він викликається до відображення DOM і може бути використаний для ініціалізації бібліотек, створених сторонніми розробниками, запуску анімацій, запиту даних або виконання будь-яких додаткових налаштувань, які потрібні перед відображенням компонента на екрані. З цього методу можна викликати метод `setState`, щоб змінити стан компонента безпосередньо перед тим, як він спочатку відображається на екрані.

До інших методів життєвого циклу установки компонента відносяться `componentDidMount` і `componentWillUnmount`. Перший викликається відразу після відображення компонента на екрані, а другий — безпосередньо перед видаленням його з екрана.

Метод `componentDidMount` є ще одним місцем, звідки зручно надсилати запити до API. Він викликається після відображення компонента на екрані, тому будь-які виклики `setState` з даного методу призведуть до запуску життєвого циклу оновлення та повторного відображення компонента на екрані. Метод `componentDidMount` також добре підходить для ініціалізації будь-якого коду JavaScript, створеного сторонніми розробниками, який потребує DOM. Наприклад, потрібно буде впровадити бібліотеку перетягування об'єктів або бібліотеку, яка обробляє події дотику до екрана. Зазвичай, таким бібліотекам перед їх ініціалізацією потрібен DOM. Крім того, цей метод буде корисним при запуску фонових процесів на зразок вимірювання інтервалів часу або таймерів. Будь-які процеси, запуснені в `componentDidMount` або `componentWillMount`, можуть бути припинені в `componentWillUnmount`. Від фонових процесів, що не використовуються, потрібно своєчасно позбавлятися.

Компоненти видаляються з екрана, коли видаляються своїми батьківськими компонентами або за допомогою методу

`unmountComponentAtNode` з пакету `react-dom`. Цей метод використовується видалення кореневого компонента. Коли кореневий компонент видаляється, спочатку видаляються його дочірні компоненти.

**Життєвий цикл оновлення** є послідовністю методів, що викликаються при зміні стану компонента або при отриманні від батьківського компонента нових властивостей. Цей життєвий цикл може бути використаний для впровадження коду JavaScript перед оновленням компонента або для взаємодії з DOM після оновлення. Крім того, він може служити для підвищення продуктивності програми, оскільки дозволяє скасувати непотрібні оновлення.

Життєвий цикл оновлення запускається при кожному виклику `SetState`. Виклик `setState` всередині життєвого циклу оновлення стане причиною нескінченного рекурсивного циклу, що призведе до помилки переповнення стека. Тому метод `setState` може викликатися лише у методі `componentWillReceiveProps`, що дозволяє компоненту оновлювати стан під час оновлення його властивостей.

До методів життєвого циклу оновлення входять:

- `componentWillReceiveProps(nextProps)` — викликається лише у разі передачі компоненту нових властивостей; єдиний метод, в якому може бути викликаний метод `setState`;
- `shouldComponentUpdate(nextProps, nextState)` - воротар життєвого
- циклу поновлення: предикат, здатний скасувати оновлення; може використовуватися для підвищення продуктивності, дозволяючи лише необхідні оновлення;
- `componentWillUpdate(nextProps, nextState)` - викликається безпосередньо
- перед оновленням компонента; схожий на метод `componentWillMount`, але викликається лише перед виконанням кожного оновлення;

- `componentDidUpdate(prevProps, prevState)` — викликається відразу після виконання оновлення, після виклику методу відображення `render`; схожий на метод `componentDidMount`, але викликається лише після кожного оновлення.

Методи **життєвого циклу компонента** дозволяють розширити контроль над тим, як компонент повинен відображатись на екрані або оновлений. Вони надають засіб, що дозволяє додавати функціональні можливості до та після того, як відбудеться встановлення або оновлення.

## 2.3 Основні можливості та особливості бібліотеки **Redux**

**Redux** — це невелика бібліотека з простим, обмеженим API, розробленим як передбачуваний контейнер для стану додатків. **Redux** був створений Деном Абрамовим та Ендрю Кларком у 2015 році.

Він дозволяє нам створювати додатки, які ведуть себе окремо в різних середовищах (клієнт, сервер і нативні додатки), а також просто тестуються. Можливо використовувати **Redux** разом із **React** або з будь-якою іншою бібліотекою.

Ця бібліотека працює в такому форматі. Весь глобальний стан нашої програми зберігається в дереві об'єктів в одному сховищі(store). Єдиний спосіб змінити дерево станів — це створити дію(action), об'єкт, який описує те, що сталося, і відправити його в сховище. Щоб вказати, як стан оновлюється у відповідь на дію, ви пишете чисті функції скорочення, які обчислюють новий стан на основі старого стану та дії.

**Redux** спрощує спосіб перегляду стану у додатку, вимагаючи від нас зберігати всі дані стану в одному об'єкті. Все, що потрібно знати про додаток, знаходиться в одному місці: єдине джерело істини.

Замість того, щоб змінювати стан напряму, ви вказуєте зміни, які ви хочете виконати з простими об'єктами, які називаються діями(actions).

Потім ви пишете спеціальну функцію, яка називається `reducer`, щоб вирішити, як кожна дія перетворює стан усієї програми.

Стан програми повинен зберігатися в одному незмінному об'єкті. Незмінність означає, що це об'єкт стану ні змінюватися. Час від часу він оновлюватиметься за рахунок його повної заміни. Щоб втілити цей задум у життя, нам знадобляться інструкції щодо змін. Саме це і надають дії: інструкції, що стосуються змін, що вносяться до стану програми, які супроводжуються даними, необхідні для внесення змін.

Дії є єдиним способом оновлення стану програми `Redux`. Вони надають нам інструкції про те, що має бути змінено, але ми можемо розглядати їх і як записи історії змін, що відбулися з часом.

Все наше дерево стану зберігається в одному об'єкті. У `Redux` модульність досягається за рахунок функцій. Вони використовуються для оновлення частин дерева стану. Ці функції називаються перетворювачами (`Reducers`).

Перетворювачі є функціями, які отримують поточний стан і дію у вигляді аргументів і використовують їх для створення та повернення нового стану. Розробляються оновлення конкретних частин дерева стану: або листя, або гілок. Потім перетворювачі можна збирати в один, що керує всім станом програми при будь-яких діях. Кожен перетворювач складається з функції або зводиться в одну функцію перетворювача, яка використовуватиме сховище.

У сховищі `Redux` вважається те місце, де зберігаються дані стану програми і обробляються всі оновлення стану. Хоча модель конструювання `Flux` допускає наявність безлічі сховищ, кожне з яких націлене на конкретний набір даних, в `Redux` є тільки одне.

Сховище займається оновленням стану, пропускаючи поточний стан та дію через єдиний перетворювач. Єдиним способом зміни стану вашої програми є диспетчеризація дій через сховище. У ньому є метод `dispatch`, готовий отримати дії як аргументу. При диспетчеризації за

допомогою сховища дія проводиться через перетворювачі та стан оновлюється.

У типовому додатку Redux є лише одне сховище із єдиною функцією редьюсером. У міру зростання програми можна розділити єдиний редьюсер на менші редьюсери, незалежно діючі в різних частинах дерева стану. Це точно так само, як у додатку React є лише один кореневий компонент, але він складається з багатьох маленьких компонентів.

Ця архітектура може здатися великою для простої програми, але краса цього шаблону полягає в тому, наскільки добре він масштабується для великих і складних програм. Він також дає змогу використовувати дуже потужні інструменти розробника, оскільки можна відстежити кожну зміну до дії, яка її спричинила. Ви можете записувати сеанси користувача та відтворювати їх, просто відтворюючи кожну дію.

Приклад сховища додатку[5]:

```
{
  todos: [{
    text: 'Eat food',
    completed: true
  }, {
    text: 'Exercise',
    completed: false
  }],
  visibilityFilter: 'SHOW_COMPLETED'
}
```

React Redux — бібліотека, що містить інструменти, які допомагають спростити передачу сховища через контекст. Ця бібліотека також надана Деном Абрамовим (Dan Abramov), творцем Redux. Сам Redux можна використовувати без неї. Але застосування React Redux спрощує код і допоможе прискорити створення додатків.



Бібліотека react-redux надає компонент провайдера, який служить для налаштування сховища в контексті. У провайдері можна укласти будь-який елемент React, і тоді його дочірні елементи отримають доступ до сховища через контекст.

Провайдер `provider` додає сховище до контексту та оновлює компонент `App` після диспетчеризації дій. Провайдеру потрібен один дочірній компонент:

```
const App = () => {  
  return (  
    <Provider store={store}>  
      <div className="App">  
        <Header/>  
        <Main />  
        <Footer />  
        <Cart />  
      </div>  
    </Provider>  
  );  
};
```

Провайдер вимагає передачі сховища як властивості. Він додає сховище до контексту, щоб воно могло вилучатися будь-яким дочірнім компонентом, що належить компоненту `App`. Після встановлення провайдера з'являється можливість витягувати сховище через контекст у дочірніх контейнерних компонентах.

## 2.4 Висновки к розділу 2

Отже, ми провели огляд можливостей React та Redux.

Розглянули особливості фреймворку React: технічні аспекти, можливості та сфери застосування. Проведено аналіз роботи з DOM деревом. Проаналізовано можливості застосування стилю розмітки JSX.

Було проведено аналіз хуків React(React hooks). Дано основні поняття про хуки. Детально розглянуті основні React хуки: useState, useContext, useReducer, useMemo, useEffect, useRef, useCallback тощо.

Проаналізовано основні життєві цикли компонентів та їх методи. А саме методи: ShouldComponentUpdate, componentDidMount, componentWillUnmount, render тощо.

Розглянуто життєві цикли установки та оновлення.

Зроблено аналіз можливостей та особливостей Redux. Наведено основну інформацію. Проведено огляди на сховища(stores) та редюсери(reducers), наведено їх приклади. Розглянуто бібліотеку react-redux та можливості застосування Provider-компоненти.

## ГЛАВА 3. ПРАКТИЧНА РЕАЛІЗАЦІЯ ФРОНТЕНД САЙТУ НА ОСНОВІ REACT-REDUX

### 3.1 Опис інтерфейсу сайту

Виконавши роботу по створенню каркасу додатку, ми можемо розглянути результат рис. 7.



рис 7 - Верхня частина сайту - Header та Slider

В наявності є Header частина. Вона має динамічне розташування на сторінці, це зроблено для поліпшення доступу користувача до навігаційних елементів, які знаходяться в Header. Це зроблено за допомогою властивості `position: sticky` в CSS додатку. Елемент кошику в панелі має динамічний лічильник кількості різних позицій, доданих в кошик. При цьому, кількість однієї позиції не впливає на збільшення загального лічильнику.

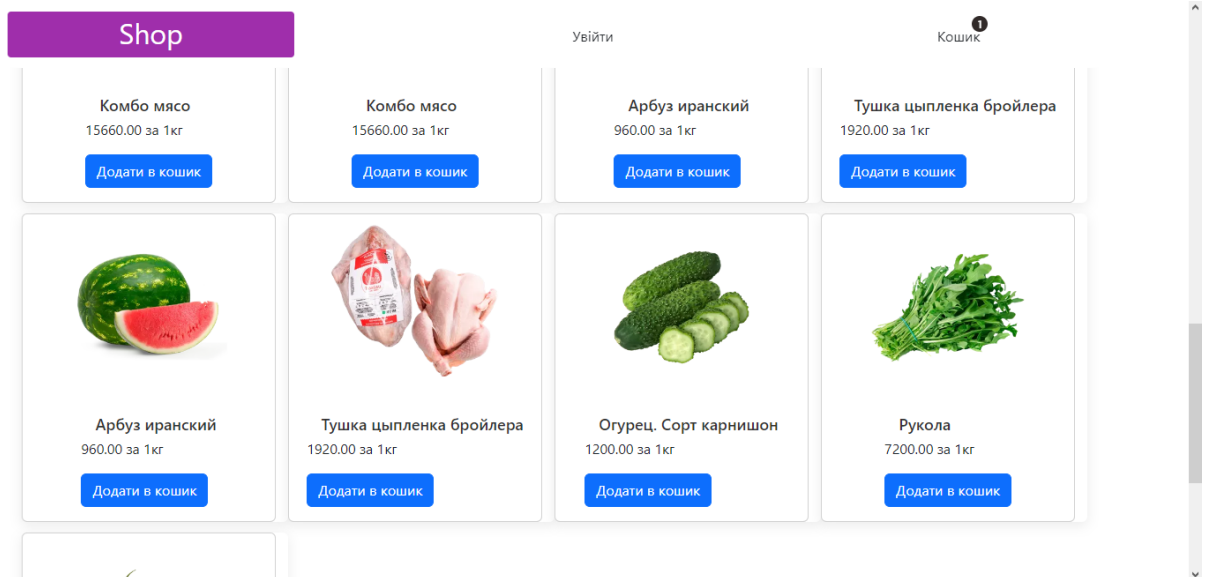


Рис 8 - Головна частина сайту з картками товарів для продажу

Нижче є список наявних товарів, які користувач може додати до кошику. Вони мають вигляд карток з назвою, ціною, та кнопкою додавання в кошик (рис. 8).

### 3.2 Реалізація головної сторінки сайту

За допомогою бібліотеки react-create-app, створено базу для нашої подальшої розробки. Ця бібліотека цілком підходить для невеликих проєктів, вона розгортає стандартний набір інструментів.

Основний функціонал інтернет-магазину полягає в можливості переглянути товар, та замовити його. Оскільки робота була зосереджена на фронтенд частині, була зроблена імітація запитів на серверну частину, шляхом збереження тестових даних(mockData) в форматі JSON безпосередньо в структурі фронтенд-частини, та подальшого їх використання.

Точкою входу в додаток є файл App.js.

Лістинг:

```
import './App.css';
import Header from './components/Header/Header';
import Main from './components/Main/Main';
```

```

import { Provider } from 'react-redux';
import store from './store/store';
import 'bootstrap/dist/css/bootstrap.min.css';
import Footer from './components/Footer/Footer';
import Cart from './components/Cart/Cart';

const App = () => {
  return (
    <Provider store={store}>
      <div className="App">
        <Header/>
        <Main />
        <Footer />
        <Cart />
      </div>
    </Provider>
  );
};

export default App;

```

Колбек-функція App() є функціональним компонентом Реакт-додатку. Оскільки в Реакт структура елементів додатку йде зверху вниз, імпортуються основні частини додатку до кореневого компоненту, а саме: Header; Main; Footer; Cart.

Із бібліотеки react-redux імпортується Provider компонент. В нього обертаються усі інші компоненти. Компонент Provider робить сховище Redux доступним для будь-яких вкладених компонентів, яким потрібен доступ до сховища Redux.

Оскільки будь-який компонент React у додатку React Redux можна підключити до магазину, більшість додатків відображатимуть <Provider> на верхньому рівні з усім деревом компонентів додатка всередині нього.

Хуки та API підключення можуть отримати доступ до наданого екземпляра магазину через механізм контексту React.

Окрім хедеру, футеру, та основної частини сайту, також імпортується компонент Cart - це компонент кошику, в який користувач зможе добавляти товари. Використання цього компоненту безпосередньо в головному компоненті зумовлено специфікою HTML-верстки, та необхідності можливості відкривання кошика з будь-якої частини додатку і коректного його візуального відображення для користувача.

Компонент Header відповідає за “шапку” сайту, в якій знаходяться інформаційні блоки з назвою магазину, контактними даними, кнопками кошику тощо.

Лістинг:

```
import React from "react";
import { useDispatch, useSelector } from "react-redux";
import { toggleCartOpen } from "../../store/mainSlice";
import './Header.css'

const Header = () => {
  const dispatch = useDispatch();
  const openCartHandler = () => {
    dispatch(toggleCartOpen(1));
  }
  const cartProducts = useSelector((state) => state.mainSlice?.cart?.products);

  return(
    <div className="Header_container">
      <div className="Header__shop_title">Shop</div>
      <div className="Header__shop_text">Увійти</div>
      <div onClick={openCartHandler} className="Header__shop_text">
        Кошик
        <div className="Header_cart_qty">{cartProducts?.length}</div>
      </div>
    </div>
  )
};

export default Header;
```

В компоненті хедеру використовуються декілька Реакт-редакс-хуків.

`useSelector` і `useDispatch` — це набір хуків для використання як альтернативи існуючому компоненту вищого порядку `connect()`. `useSelector` приймає аргумент функції, і повертає потрібну частину стану.

Створюється змінна для хуку `useDispatch()`; Цей хук потрібен для оперування діями, які знаходяться в `slice` додатку.

`openCartHandler()` - функція, яка робить диспатч `toggleCartOpen`, тобто викликає дію, яка відповідає за відкриття кошику, та передає до неї значення `true` через одиницю.

`cartProducts` - змінна, яка містить в собі селектор, який звертається до сховища, та викликає звідти список продуктів, які добавлені користувачем в кошик.

Безпосередньо в верстці є декілька `<div>` елементів, які семантично відповідають за блоки. В одному з них знаходиться `onClick` властивість, яка відповідає за клік по елементу, та в неї передана функція `openCartHandler`. В разі кліку по цьому блоку, буде викликано відповідну функцію.

Також, в одному з блоків, в середині передано значення `cartProducts?.length` - це розмір масиву продуктів, які знаходяться в кошику. Це потрібно для візуального відображення кількості продуктів в кошику, не заходячи в кошик. Лістинг компоненту `Main` представлений у додатку А.

Даний компонент відповідає за головний функціональний блок магазину, а саме відображення продуктів.

Імітується встановлення даних в сховище по типу взаємодії з сервером.

```
const productsMock = require('../mockData/products.json')
useEffect(()=>{
  dispatch(setProducts(productsMock.products));
}, [dispatch])
```

Створюється запит на JSON файл, який містить `mockData`(тестувальні) дані продуктів. Далі, через хук `useEffect` передається

функція диспатчу дії, яка встановлює отримані із `mockData` дані в глобальне сховище. В список залежностей додається змінна `dispatch`. Це робиться для того, щоб функція, яка передана в `useEffect` виконувалась лише при першому рендері сторінки, а далі - ні.

За допомогою `react-bootstrap` бібліотеки створили компонент `Carousel`, який створює компонент каруселі, в яку ми можемо встановити кілька банерів, на яких потенційно можливо розміщувати якісь акційні пропозиції, новини тощо.

Компонент `ProductItemsContainer`. Лістинг:

```
import React from "react";
import { useSelector } from "react-redux";
import ProductItem from "../ProductItem/ProductItem";
import './ProductItemsContainer.css'

const ProductItemsContainer = () => {
  const products = useSelector(state => state.mainSlice?.products)

  const renderItem = () => {
    return products?.map((product)=>{
      return(<div key={product.id}>
        <ProductItem product={product}/>
      </div>)
    })
  }

  return(
    <div className="ProductItems__container">
      {products && renderItem()}
    </div>
  )
};

export default ProductItemsContainer;
```

За допомогою селектору, отриманий доступ до списку усіх продуктів зі сховища. Якщо він не пустий, то виконується функція рендеру `renderItem`, яка для кожного з продуктів створює компонент `ProductItem`, та передає через властивості(`props`), безпосередньо конкретний продукт.



ProductItem, лістинг:

```
import React from "react";
import './ProductItem.css'
import Button from 'react-bootstrap/Button';
import Card from 'react-bootstrap/Card';
import { useDispatch } from "react-redux";
import { addCartItem } from "../../store/mainSlice";

const ProductItem = (props) => {
  const dispatch = useDispatch();
  const product = props.product;

  const addProductToCartHandler = () =>{
    dispatch(addCartItem(product))
  };

  return(
    <div className="ProductItem">
      <Card style={{ width: '18rem' }}>
        <Card.Img variant="top" src={product.product.image.s350x350} />
        <Card.Body>
          <Card.Title
            className="ProductItem_card__title">{product.product.title}</Card.Title>
          <Card.Text className="ProductItem_card__price">{product.price} за
            1кг</Card.Text>
          <Button onClick={addProductToCartHandler} variant="primary">Додати в
            кошик</Button>
        </Card.Body>
      </Card>
    </div>
  )
};

export default ProductItem;
```

Компонент ProductItem відповідає за карточку товару, яка містить оригінальні значення назви, ціни продукту тощо. Через змінну products звертаємось до props компоненту, та достаємо продукт, який переданий рівнем вище в єдиному екземплярі для кожного компоненту.

За допомогою react-bootstrap створено компонент Card, який відповідає за зовнішній вигляд картки: назву, картинку, ціну тощо. На

кнопці `Button` встановлено `onClick` подію з функцією `addProductToCartHandler`. При кліці, продукт конкретного компоненту буде добавлено до кошику через `addCartItem` дію(`action`), яка імпортується з `mainSlice` додатку.

`mainSlice` - це слайс, створений за допомогою бібліотеки `redux-toolkit`, та зокрема функції `createSlice`. Ця функція автоматично генерує творці дій і типи дій, які відповідають редюсерам і стану. Цей API є стандартним підходом для написання логіки `Redux` лістинг представлено в додатку Б.

В слайсі задано його ім'я, та `initialState`. `initialState` відповідає за початковий стан додатку при першому запуску. Оскільки дані продуктів получаются із мокових даних в компоненті `Main()`, то на початковому етапі масив продуктів є пустим, та потім в нього через дію пишуться дані, цей процес було вже розглянуто вище. Також в цьому сховищі є об'єкт кошику, який включає в себе інформацію про відкритий кошик, ціни в кошику, та масив продуктів доданих в кошик.

Дії(`actions`) слайсу. `setProducts` отримує `payload` дані, та встановлює їх безпосередньо в сховище в `products` масив.

`toggleCartOpen` відповідає за переключення стану модального вікна кошику на відкритий\закритий стан.

`addCartItem` добавляє продукт до кошику. Через метод `.find()` шукаємо, чи є продукт із `payload` вже добавленим до кошику. У випадку, якщо є - відбувається зростання `amount` властивості в товару на одиницю, та обчислюється ціна враховуючи нову кількість кожного продукта. В іншому випадку, `productToAdd` об'єкт створюється та містить в собі безпосередньо сам продукт, та його `id`, ціну, кількість, і потім додається в масив продуктів кошику через метод `.push()`. Після цього відбувається перерахунок загальної вартості всіх товарів кошику. Перерахунок відбувається враховуючи ціни за одиницю товару та кількість доданих продуктів кожної позиції.

### 3.3 Реалізація праці із корзиною

Компонент кошику.

Лістинг:

```
import React from "react";
import { useDispatch, useSelector } from "react-redux";
import './Cart.css'
import { toggleCartOpen } from '../store/mainSlice'

const Cart = (props) => {
  const dispatch = useDispatch();
  const cartIsOpen = useSelector((state) => state.mainSlice?.cart?.isOpen);
  const cartProducts = useSelector((state) => state.mainSlice?.cart?.products);

  const closeCartHandler = () => {
    dispatch(toggleCartOpen());
  }

  const renderCartProducts = () => {
    if(cartProducts?.length > 0){
      return cartProducts?.map((cartProduct)=>{
        return(
          <div key={cartProduct.id} className="cartProductItem">
            <div>{cartProduct.product.title}</div>
            <div>Кількість: {cartProduct.amount}</div>
            <div>Вартість: {cartProduct.price}</div>
          </div>
        )
      })
    } else {
      return(<div>Кошик пустий</div>)
    }
  }

  return(
    <>{cartIsOpen && (
      <div className="Cart__container">
        <div className="Cart__container_modal">
          <div className="Cart__container_wrapper">
            <div className="Cart__header">
              <h3>Кошик</h3>
              <button onClick={closeCartHandler}>Закрити</button>
            </div>

```

```

        <div>
          {renderCartProducts()}
        </div>
      </div>
    </div>
  )}
</>

)
};

export default Cart;

```

Використовуються `useDispatch` і `useSelector` хуки.

За допомогою селектору, викликається значення `cartIsOpen` зі сховища. Воно відповідає за те, чи відкритий кошик.

`renderCartProducts` функція відповідає за рендер списку продуктів в кошику. Якщо масив продуктів має продукти, відбувається перебор, інакше - інформація про пустий кошик. За допомогою методу `map()` масив продуктів перебирається, та з кожним елементом масиву проводиться операція, а саме повернення HTML елементів, в яких знаходяться дані по кожному із продуктів.

Далі функцію `{renderCartProducts()}` розміщено безпосередньо серед HTML розмітки, і вона одразу виконується в разі відображення компонента.

Безпосередньо відображення кошику кошику містить змінна `cartIsOpen`, яка включає в себе селектор зі сховища. Якщо значення `true`, то кошик відображається. Оскільки в компоненті `App` знаходиться компонент кошику, то сам компонент відображається завжди(він невидимий для користувача), але лише при правдивому значенні відбувається ре-рендер компоненти, і тоді користувач може бачити компонент кошику на екран (рис. 9).

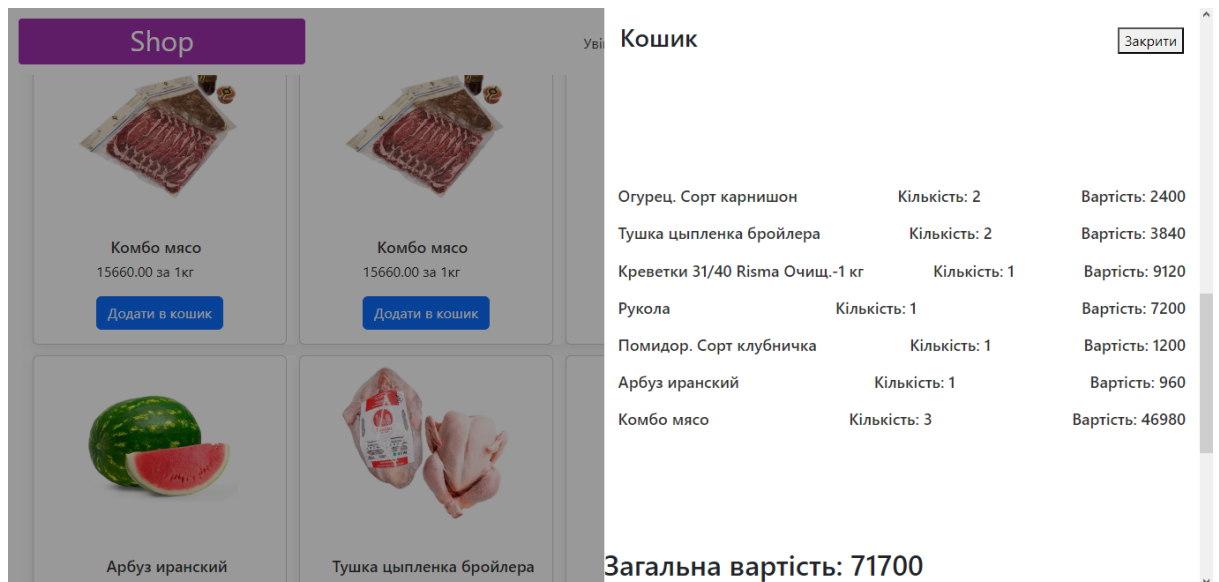


Рис. 9 - Компонент кошику з доданими товарами та вартістю

Модальне вікно кошику має в собі список всіх доданих продуктів: їх назву, кількість доданих, загальну вартість кожної позиції, розраховану з ціни за одиницю та кількості доданих. Нижче є загальна вартість всіх товарів, доданих до кошику.

### 3.4 Тестування

Для тестування виконується додавання товару до кошика. Спочатку кошик порожній, список доданих продуктів порожній (рис. 10).

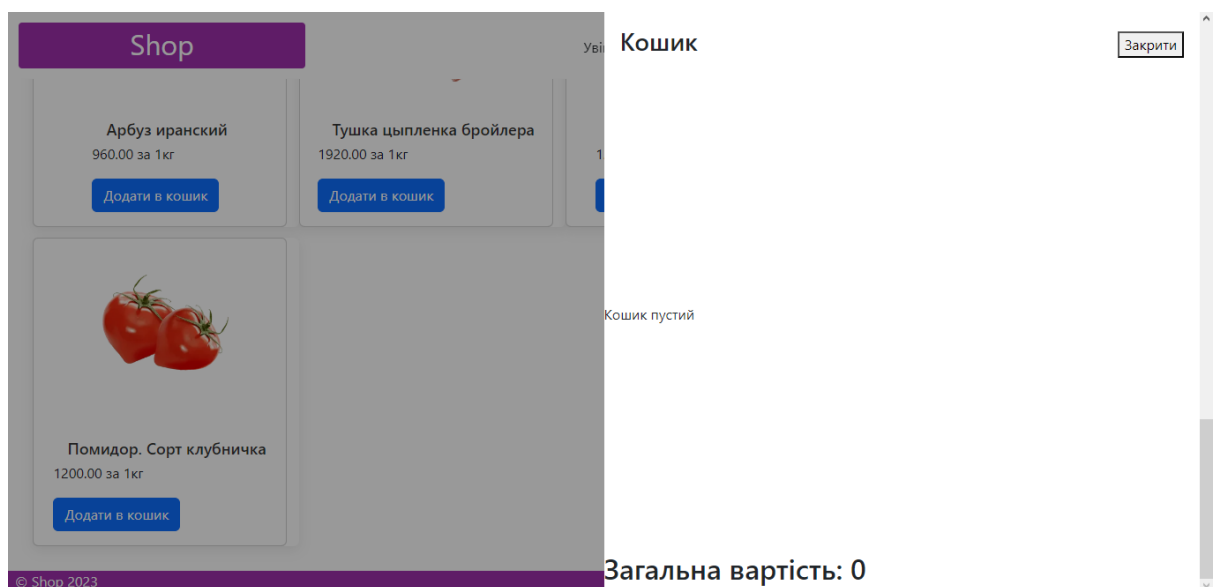


Рис 10 - Модальне вікно кошику без доданих товарів

Додається довільний товар через картку та кнопку "додати до кошика"  
рис. 11.

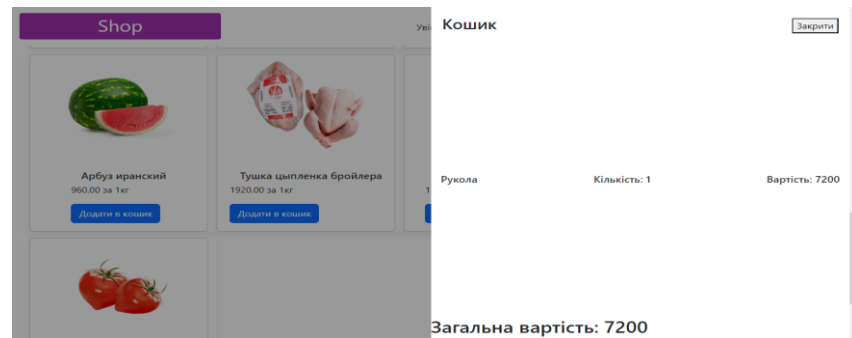


Рис 11 - Модальне вікно кошику з одним доданим товаром

Можемо спостерігати, що продукт успішно додався до кошика.  
Можемо у кошику спостерігати його назву, кількість, ціну (рис. 12).

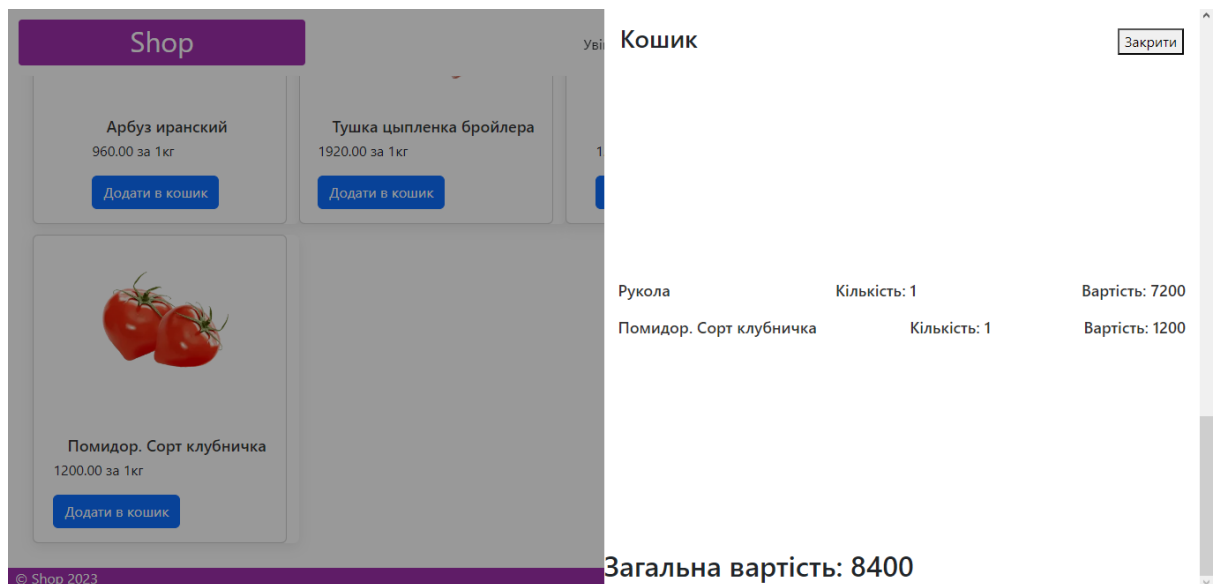


Рис. 12 - Модальне вікно кошику з двома доданими товарами

При додаванні ще одного товару система відпрацювала успішно.  
Новий товар додано до кошика. Відбулася успішна калькуляція загальної  
вартості. Товари в кількості 1 одиниці коштують 7200 та 1200 гривень  
відповідно, а загальна вартість – 8400 грн.

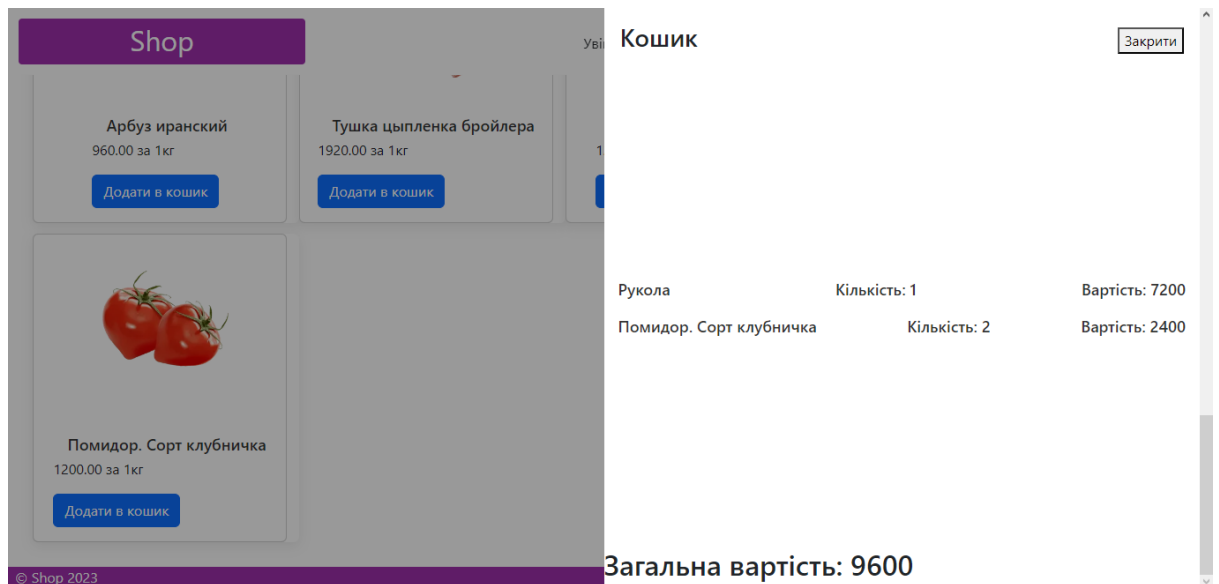


Рис. 13 - Модальне вікно кошику з двома доданими товарами, один з яких кількістю дві штуки

При додаванні одного товару двічі змінюється лише кількість товару в кошику, а не додається нова сутність. Калькуляція також успішно відбувається – 9600 грошей загальної вартості.

### 3.5 Висновки до розділу

Було розглянуто додаток інтернет-магазину на основі React & Redux. Розглянуто лістинги основних компонентів: App, Main, Header, Cart тощо. Розглянуто роботу цих компонентів.

Проведено приклади використання React хуків. Наведено приклад використання Redux сховища та доступу до нього з різних частин програми.

## ВИСНОВОК

Було розглянуті основні інструменти розробника при роботі над фронтенд частиною веб-додатків, а саме HTML, CSS, JavaScript, React\Redux. Проведено огляд фреймворків React, Vue, Angular, вивчено основні можливості. Зроблено порівняльний аналіз цих фреймворків.

Було детально розглянуто фреймворк React, та бібліотеку Redux; основні програмні прийоми, необхідні при роботі з цими бібліотеками.

В ході роботи було розроблено фронтенд частину для інтернет-магазину на основі React Redux.



## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Codecademy. React: The Virtual DOM | Codecademy. Codecademy. URL: <https://www.codecademy.com/article/react-virtual-dom> (дата звернення: 3.05.2023).
2. Hooks at a Glance – React. React. URL: <https://reactjs.org/docs/hooks-overview.html> (дата звернення: 12.04.2023).
3. Rules of Hooks – React. React. URL: <https://reactjs.org/docs/hooks-rules.html> (дата звернення: 3.05.2023).
4. JSX. URL: <https://facebook.github.io/jsx/> (дата звернення: 12.04.2023).
5. Core Concepts - Redux. README - React-Redux. URL: <https://redux.gitbook.io/docs/introduction/coreconcepts> (дата звернення: 5.04.2023).
6. Interactive Results. · GitHub Pages. URL: <https://krausest.github.io/js-framework-benchmark/current.html> (дата звернення: 05.04.2023).
7. Chinnathambi K. Learning React: A Hands-On Guide to Building Web Applications Using React and Redux (2nd Edition). Addison-Wesley Professional, 2018. 304 с.
8. Learning React: Functional Web Development with React and Redux. Sebastopol, CA : O'Reilly Media, Inc., 2018.
9. ReactJS - JSX. Online Courses and eBooks Library. URL: [https://www.tutorialspoint.com/reactjs/reactjs\\_jsx.htm](https://www.tutorialspoint.com/reactjs/reactjs_jsx.htm) (дата звернення: 11.04.2023).
10. React Components. W3Schools Online Web Tutorials. URL: [https://www.w3schools.com/REACT/react\\_components.asp](https://www.w3schools.com/REACT/react_components.asp) (дата звернення: 10.04.2023).

## ДОДАТКИ

### Додаток А Лістинг компоненту main

```
import React, {useEffect} from "react";
import ProductItemsContainer from
"./ProductItemsContainer/ProductItemsContainers";
import { useDispatch } from "react-redux";
import './Main.css'
import { setProducts } from '../store/mainSlice'
import Carousel from "react-bootstrap/Carousel";

const Main = () => {
  const dispatch = useDispatch();

  const productsMock = require('../mockData/products.json')
  useEffect(()=>{
    dispatch(setProducts(productsMock.products));
  }, [dispatch])

  return(
    <div className="Main_container">
      <Carousel className="Main_container__carousel"
variant="dark">
        <Carousel.Item>
          <img
            className="d-block w-100"
            src='data:image/svg+xml; charset=UTF-
8,<svg%20width%3D"800"%20height%3D"400"%20xmlns%3D"http%3A%2F
%2Fwww.w3.org%2F2000%2Fsvg"%20viewBox%3D"0%200%20800%20400
"%20preserveAspectRatio%3D"none"><defs><style%20type%3D"text%2Fcss"
```

```

>%23holder_18779503175%20text%20%7B%20fill%3A%23ffffff%3Bfont-
weight%3Anormal%3Bfont-family%3Avar(--bs-font-sans-
serif)%2C%20monospace%3Bfont-
size%3A40pt%20%7D%20<%2Fstyle><%2Fdefs><g%20id%3D"holder_18779
503175"><rect%20width%3D"800"%20height%3D"400"%20fill%3D"%23373
940"><%2Frect><g><text%20x%3D"289.71875"%20y%3D"221.3">First%20s
lide<%2Ftext><%2Fg><%2Fg><%2Fsvg>'
    alt="Second slide"
  />

  <Carousel.Caption>
  <h5>Banner #1 Title</h5>
  <p>Nulla vitae elit libero, a pharetra augue mollis
interdum.</p>
  </Carousel.Caption>
</Carousel.Item>
<Carousel.Item>
  <img
    className="d-block w-100"
    src='data:image/svg+xml; charset=UTF-
8,<svg%20width%3D"800"%20height%3D"400"%20xmlns%3D"http%3A%2F
%2Fwww.w3.org%2F2000%2Fsvg"%20viewBox%3D"0%200%20800%20400
"%20preserveAspectRatio%3D"none"><defs><style%20type%3D"text%2Fcss"
>%23holder_1877950317a%20text%20%7B%20fill%3A%23ffffff%3Bfont-
weight%3Anormal%3Bfont-family%3Avar(--bs-font-sans-
serif)%2C%20monospace%3Bfont-
size%3A40pt%20%7D%20<%2Fstyle><%2Fdefs><g%20id%3D"holder_18779
50317a"><rect%20width%3D"800"%20height%3D"400"%20fill%3D"%23282
c34"><%2Frect><g><text%20x%3D"251.96875"%20y%3D"221.3">Second%2
0slide<%2Ftext><%2Fg><%2Fg><%2Fsvg>'
    alt="Second slide"
  />
  <Carousel.Caption>

```

```

        <h5>Banner #2 Title</h5>
        <p>Lorem ipsum dolor sit amet, consectetur adipiscing
elit.</p>

    </Carousel.Caption>
</Carousel.Item>
<Carousel.Item>
    <img
        className="d-block w-100"
        src='data:image/svg+xml; charset=UTF-
8,<svg%20width%3D"800"%20height%3D"400"%20xmlns%3D"http%3A%2F
%2Fwww.w3.org%2F2000%2Fsvg"%20viewBox%3D"0%200%20800%20400
"%20preserveAspectRatio%3D"none"><defs><style%20type%3D"text%2Fcss"
>%23holder_1877950317d%20text%20%7B%20fill%3A%23ffffff%3Bfont-
weight%3Anormal%3Bfont-family%3Avar(--bs-font-sans-
serif)%2C%20monospace%3Bfont-
size%3A40pt%20%7D%20<%2Fstyle><%2Fdefs><g%20id%3D"holder_18779
50317d"><rect%20width%3D"800"%20height%3D"400"%20fill%3D"%23202
32a"><%2Frect><g><text%20x%3D"278.3203125"%20y%3D"221.3">Third%
20slide<%2Ftext><%2Fg><%2Fg><%2Fsvg>'
        alt="Third slide"
    />
    <Carousel.Caption>
    <h5>Banner #3 Title</h5>
    <p>
        Praesent commodo cursus magna, vel scelerisque nisl
consectetur.
    </p>
    </Carousel.Caption>
</Carousel.Item>
</Carousel>
<ProductItemsContainer/>
</div>
)

```

```
};
```

```
export default Main;
```

## Додаток Б Лістинг mainSlice

```
import { createSlice } from '@reduxjs/toolkit';
```

```
const mainSlice = createSlice({
  name: 'store',
  initialState: {
    loading: null,
    loadingProduct: null,
    products: [],
    cart: {
      isOpen: false,
      cartPrice: {
        amount: 0,
        delivery: 0,
        total: 0,
      },
      products: []
    }
  },
  reducers: {
    setProducts(state, payload) {
      state.products = payload.payload;
    },
    toggleCartOpen(state, { payload }) {
      if (typeof payload === 'boolean') {
        state.cart.isOpen = payload;
      } else {
        state.cart.isOpen = !state.cart.isOpen;
      }
    },
    addCartItem(state, { payload }) {
      const product = state.cart?.products.find(
        (product) => product.id === payload.id
      );

      if (product) {
        product.amount += 1
        product.price += Math.round(parseFloat(payload.price) * 100) / 100;
      } else {
        const productToAdd = {
          id: payload.id,
          amount: 1,
```

```

        price: Math.round(parseFloat(payload.price) * 100) / 100,
        product: payload.product,
      };
      state.cart.products.push(productToAdd);
    }
    state.cart.cartPrice.amount = state.cart.products.length;
    let totalPrice = 0;
    for(let i= 0; i < state.cart.products.length; i++){
      totalPrice += state.cart.products[i].price;
    }
    state.cart.cartPrice.total = totalPrice;

  },
},
});
export const { setProducts, toggleCartOpen, addCartItem } = mainSlice.actions;
export default mainSlice.reducer;

```

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**  
Факультет (інститут) **Навчально-науковий інститут фізики, математики**  
**та інформаційних технологій**  
(повна назва)  
Кафедра **Інформаційних технологій та систем**  
(повна назва)

**Програма та методика тестування**  
на виконання програмної розробки (ПР):  
**Створення фронтенда для онлайн магазину на основі React-Redux**



## ЗМІСТ

ТЕСТ ПЛАН.....	2
1.Вступ.....	2
1.1.Мета.....	2
1.2. Вхідні дані.....	2
1.3. Мета тестування.....	2
2.Умови тестування.....	3
3.Стратегія процесу тестування.....	3
3.1. Тестування інтерфейсу.....	3
3.1.2. функціональне тестування.....	4
4. План робіт.....	5
5. Кінцеві результати.....	5
ТЕСТОВІ ВИПАДКИ .....	6

## ТЕСТ-ПЛАН

### 1. Вступ

#### 1.1. Мета

Мета даного тест плану - це опис фронтенд додатку інтернет-магазину, розробленого на основі React Redux. Документ дозволяє отримати інформацію про план тестових робіт.

#### 1.2 Вхідні дані

Фронтенд додаток інтернет-магазину, написаний за допомогою фреймворків React та Redux.

#### 1.3 Мета тестування

Метою перевірки додатку є перевірка коректної роботи всіх його елементів. Більшу частину тестування буде відведена для тестування інтерфейсу, коректності логіки сайту. Підсумком процесу тестування будуть наступні матеріали:

- Висновок відносно загального стану, що надасть розробнику цього продукту картину відносно роботи додатку.
- Звіт про результати тестування.
- Задokumentовані помилки при виявленні.

Тестування буде проводитися вручну, методом "неформального" тестування (ad-hoc testing) з позиції кінцевого користувача програми.

Таким чином, буде протестований як інтерфейс, так і основні можливості та функціонал сайту.

Комп'ютерна система, що затверджена до перевірки:

Персональний комп'ютер із наступними характеристиками:

- Процесор Intel(R) Core(TM) i5-3230M CPU
- 8,00 ГБ оперативної пам'яті
- Тип системи: 64-розрядна операційна система

- Операційна система Windows 10 x64 професійна.

## **2. Умови тестування**

Додаток повинен правильно реагувати на дії користувача. Система додатку повинна правильно обробляти дані, згідно дій користувача.

## **3. Стратегія процесу тестування**

В процесі тестування додатку фронтенд інтернет-магазину на основі React Redux буде використано ad-hoc тестування.

Планується 2 етапи проведення процесу тестування:

- Перший етап полягає в складанні тест-плану.
- Другий етап буде присвячений тестуванню функціоналу додатку – перевірятиметься коректність відображення графічного інтерфейсу, елементів сайту, та коректність обчислювання вартості товарів, доданих в кошик.

Таким чином, буде протестований як інтерфейс додатку, так і основні функційні можливості.

Комп'ютерна система і ПО, на якому буде проводитися тестування:

- 8,00 ГБ оперативної пам'яті
- ОС Windows 10 Професійна x64.
- Процесор Intel(R) Core(TM) i5-3230M CPU

### **3.1. Тестування інтерфейсу**

мета:

- Перевірка коректного відображення головної сторінки, елементів сайту;
- Перевірка функціональності при додаванні товару в кошик;
- Перевірка функціональності обчислювання вартості товарів, доданих в кошик;

- Перевірка функціональності при додаванні в кошик одного товару декілька раз.

#### 4. План робіт

Задача	Об'єм робіт	Дата початку	Дата закінчення
Укладання тест-плану	12 годин	01.05.2023	03.05.2023
Виконання тестування	12 години	04.05.2023	06.05.2023
Аналіз тестування	6 годин	07.05.2023	09.05.2023
Підведення підсумків	6 годин	10.05.2023	12.05.2023

#### 5. Кінцеві результати

##### 5.1 Підсумок

Підсумком проведення тестування є оформлений додаток «Тест-план» (цей документ) та «Тестові випадки», які містять результат процесу тестування.

## ТЕСТОВІ ВИПАДКИ

### Тестування інтерфейсу

**Тестова задача:-** Перевірка функціональності праці додатку, та коректного графічного відображення елементів сайту.

1. Локальний запуск додатку за допомогою пакетного менеджера npm, та команди “npm start” (Рис.1).

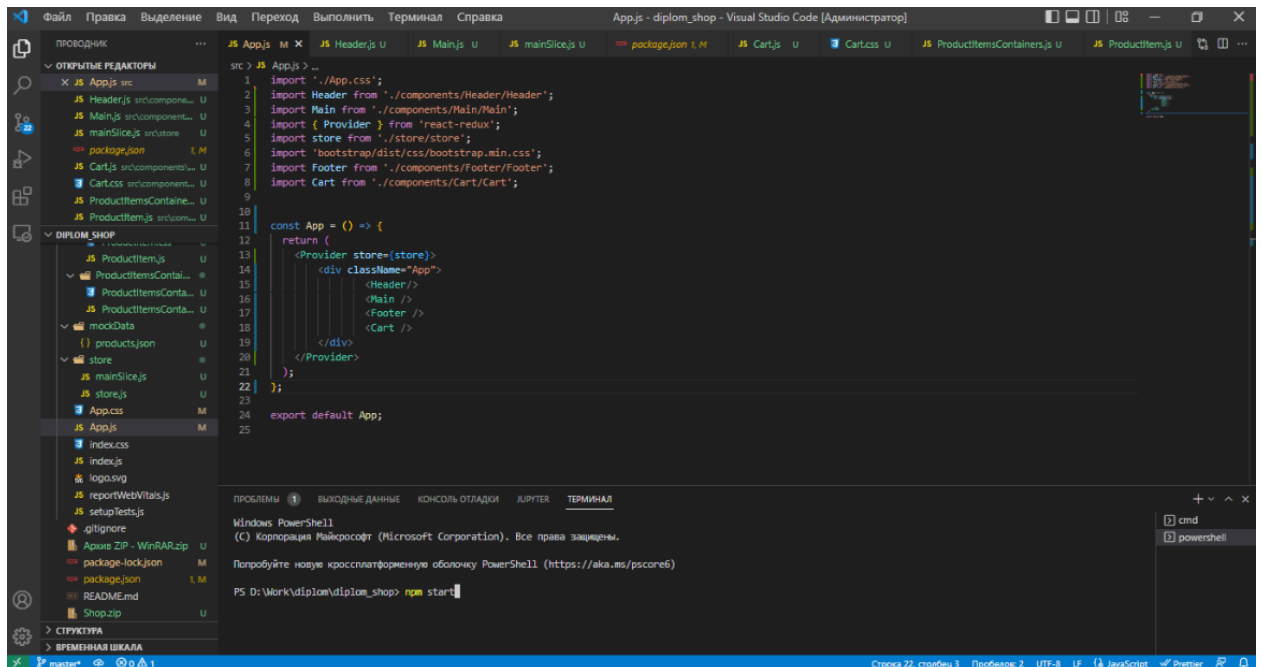


Рис.1. VS Code, запуск додатку

2. Перевірка на коректне відображення елементів додатку:



Рис.2 Верхня частина додатку.

Відображення елементі навігаційної верхівки – коректне. Коректно працює слайдер. Є елементи назви інтернет-магазину, кнопка увійти(з заглушкою), та кнопка кошика.

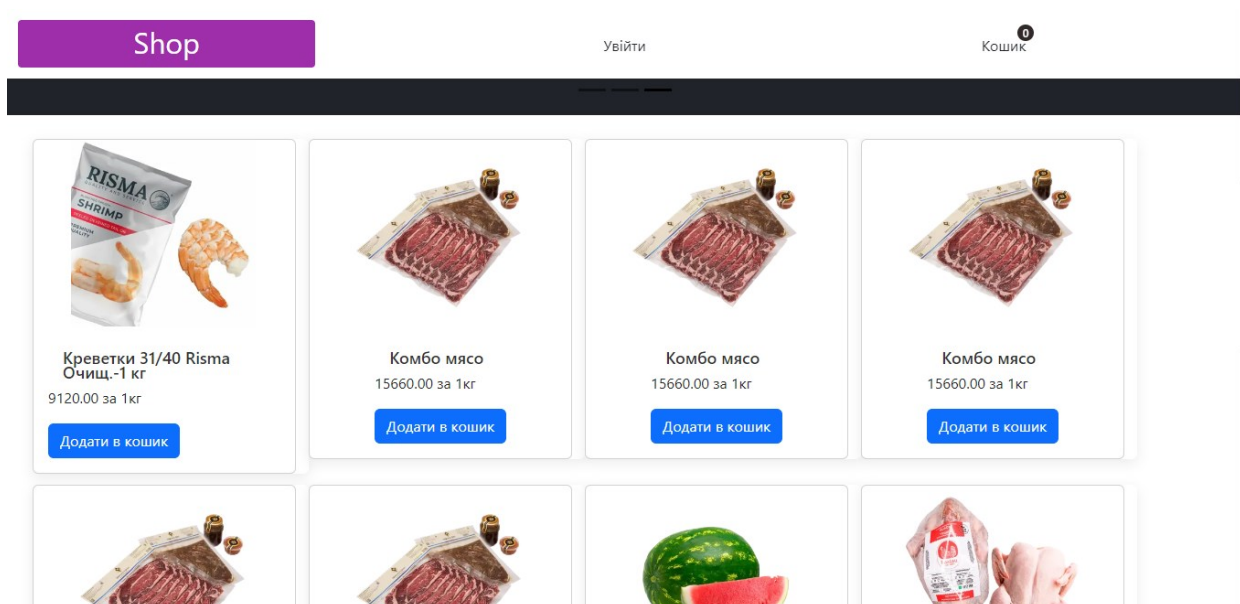


Рис.3 Головна частина сайту, картки товарів.

Правильно відображаються картки товарів, інформація про товар є: назва, ціна. Є кнопка додавання до кошика. Елемент навігаційного меню статично закріплений для комфортного доступу до його елементів будь-якої частини сторінки.

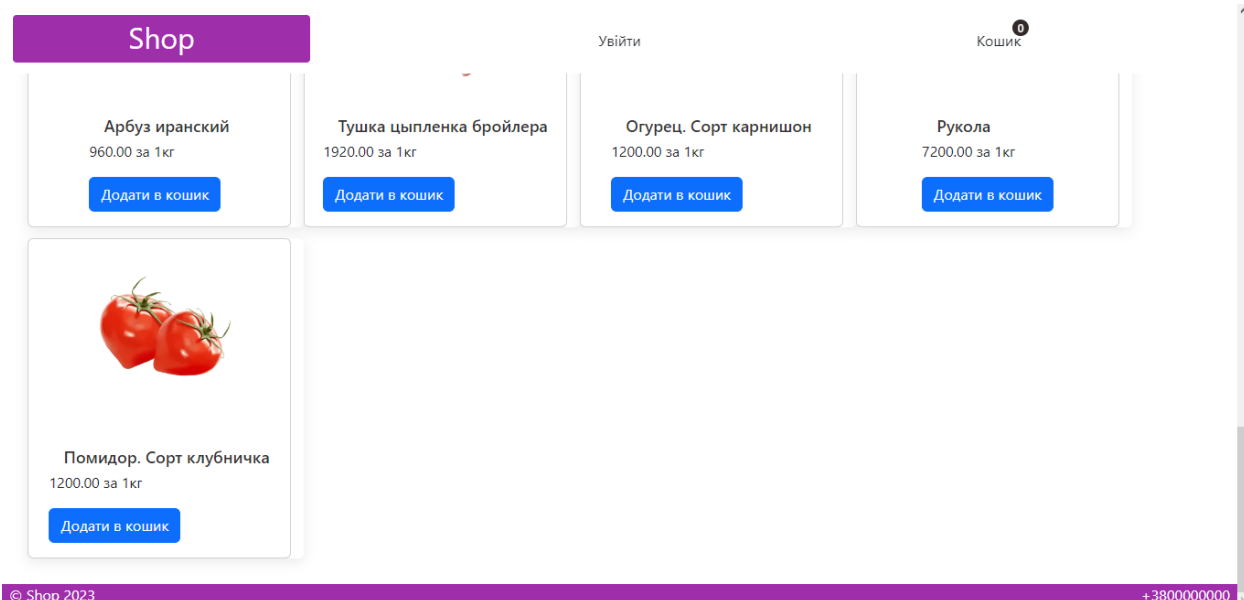


Рис.4 Нижня частина сайту(footer)

У нижній частині сайту коректно відображена інформація про авторські права та контактний номер телефону.

3. Перевірка функціональності при додаванні товару в кошик;

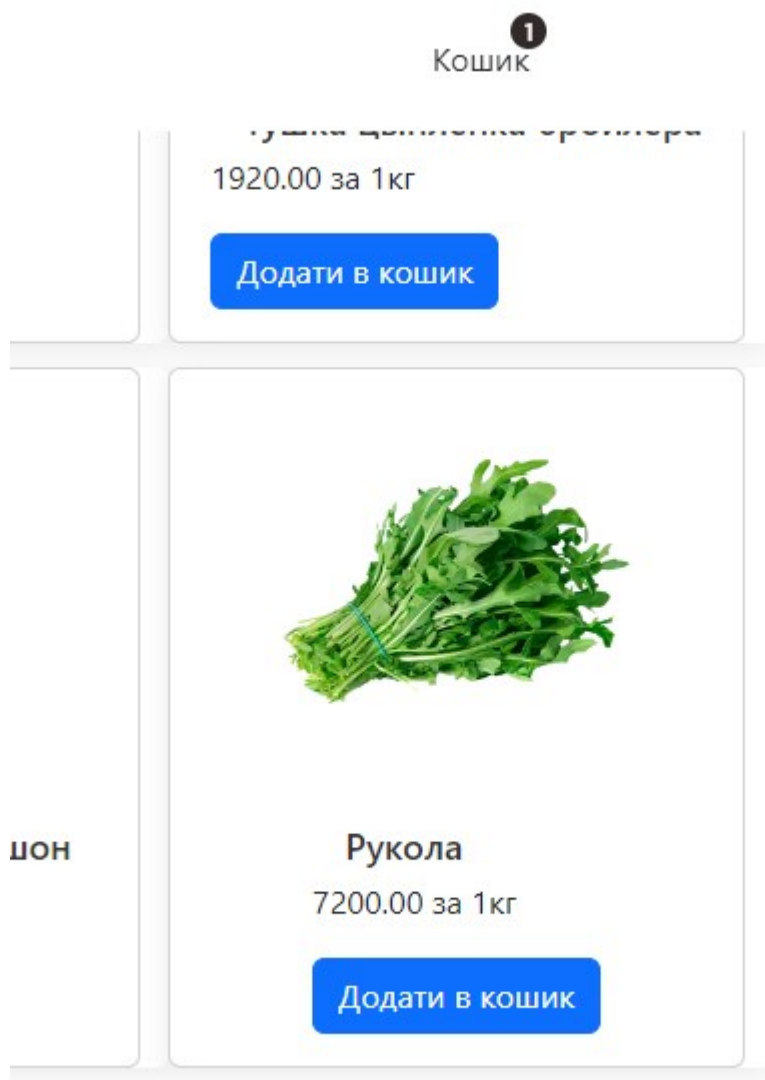


Рис.5 Картка товару та динамічний лічильник кількості товарів в кошику

При натисканні кнопки додавання до кошика коректно змінюється лічильник товарів у кошику. Це можна спостерігати на елементі кнопки кошика в навігаційному меню, є графічне відображення кількості товарів, доданих до кошика.



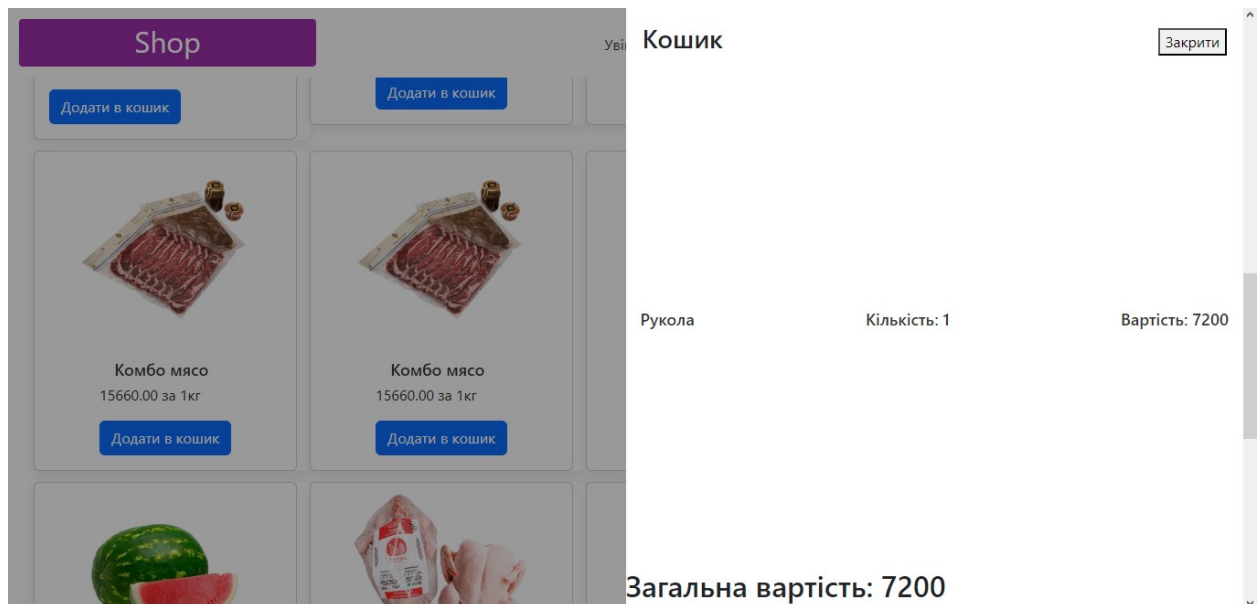


Рис.5 Модальне вікно кошику.

Після додавання товару в кошик, у модальному вікні кошика з'явиться відповідний запис про товар: його ціна, назва, кількість доданих одиниць. У нижній частині вікна відображається загальна вартість.

4. Перевірка функціональності обчислювання вартості товарів, доданих в кошик;

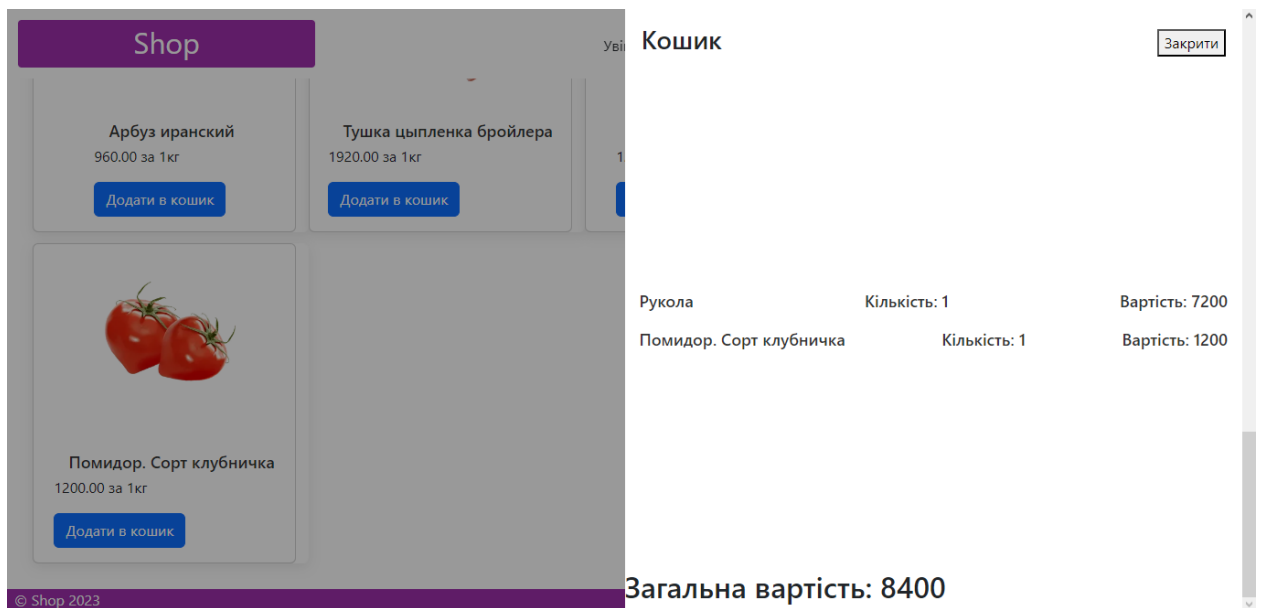


Рис.6 Додавання двох товарів в кошик

При додаванні двох товарів можемо протестувати обчислення загальної вартості. Додано два товари, вартістю 7200 та 1200. Загальна вартість обчислюється як 8400, що є коректним.

5. Перевірка функціональності при додаванні в кошик одного товару декілька раз.

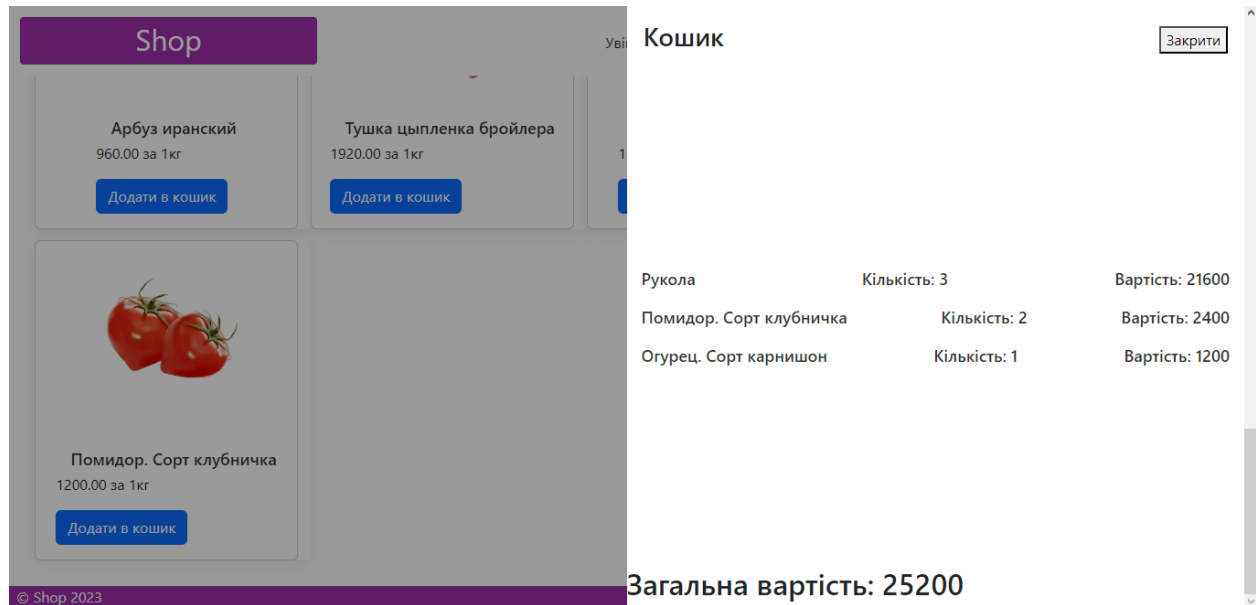


Рис.7 Додавання одного товару в кошик декілька раз

При додаванні одного й того самого товару в кошик можемо спостерігати, що нові записи не додаються, а лише змінюється параметр кількості кожного товару. Ціна товару розраховується за схемою: ціна товару за одиницю помножити кількість доданого товару. Загальна вартість усіх товарів відображається та розраховується коректно.



Рис.8 Лічильник кількості товарів в кошику, при додавання декількох товарів.

Також, лічильник кількості товарів, доданих у кошик, працює коректно та відображає лише кількість позицій у кошику, а не кількість усіх доданих товарів у сумі.

Результати тестувань додатку показали, що він відповідає усім встановленим вимогам.

**Міністерство освіти і науки України**  
**Державний заклад «Луганський національний університет**  
**імені Тараса Шевченка»**

Факультет  
(інститут)

Навчально-науковий інститут фізики,  
математики та інформаційних технологій

(повна назва)

Кафедра

Інформаційних технологій та систем

(повна назва)

**КЕРІВНИЦТВО КОРИСТУВАЧА**  
на виконання програмної розробки (ПР):  
**СТВОРЕННЯ ФРОНТЕНДА ДЛЯ ОНЛАЙН МАГАЗИНУ НА**  
**ОСНОВІ REACT-REDUX**

## **ЗМІСТ**

ВСТУП .....	3
1. Підготовка до роботи з додатком .....	4
2. Робота з додатком.....	6

## **ВСТУП**

Фронтенд інтернет-магазину на базі React Redux – це сфера розробки сайтів, націлена на розробку клієнтської частини програми, її логіки та графічного інтерфейсу. Це дає великий потенціал до масштабування та подальшого розвитку проекту, враховуючи можливість швидкої розробки та розгортання проектів на базі React. У додатку користувач може переглянути список товарів та додати їх у кошик. Масштабованість проекту дозволяє потенційно додати більший функціонал у майбутньому.

**Загальні характеристики:** Програма була розроблена на базі Реакт фреймворку, за допомогою мови програмування Javascript. Використані технології для верстки: HTML; CSS; бібліотека Bootstrap. Для покращення роботи зі станом програми та сховища використовується фреймворк Redux.

**Платформа для користування додатком:** персональний комп'ютер, браузер.

**Об'єкти додатку:** користувач, фронтенд частина додатку.

## ПІДГОТОВКА ДО РОБОТИ З ДОДАТКОМ

Мінімальні вимоги до персонального комп'ютера:

- Процесор з тактовою частотою 1.2 ГГц і більше.
- Відеокарта із 128 Мб відеопам'яті і вище.
- 256 Мб оперативної пам'яті і вище.
- 100 Мб вільного місця на накопичувачі.
- Дисплей із розширенням 1366x768 і вище.
- Клавіатура, комп'ютерна миша/тачпад.
- Мережа Internet

**Програмні вимоги:**

- Для комфортного запуску додатку, та перегляду файлів проекту та конфігураційних файлів, рекомендується до використання редактор коду Visual Studio Code.(посилання на доступ: <https://code.visualstudio.com/>)

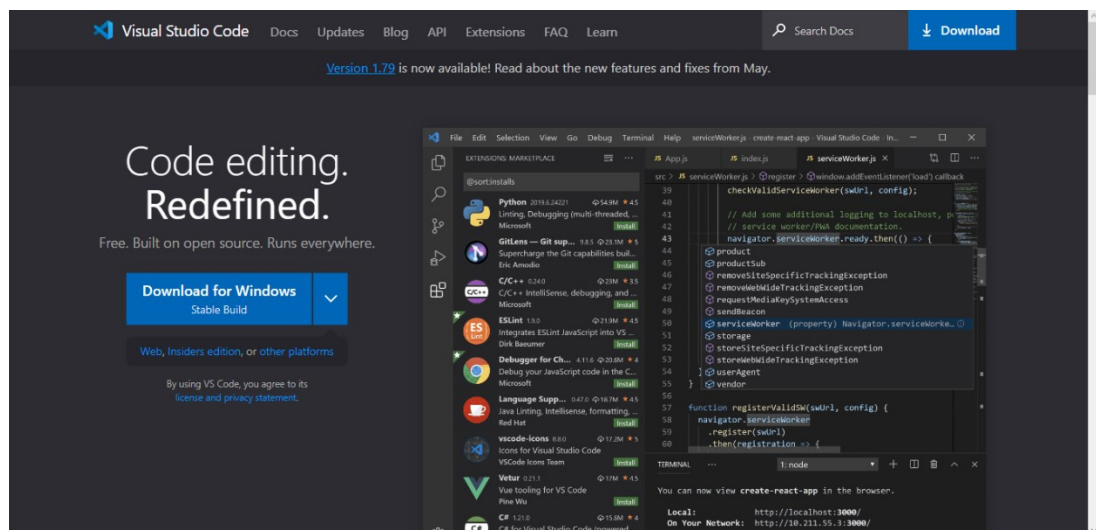


Рис.1 – сторінка завантаження редактору коду VS Code

- Для коректного запуску додатку, треба встановити програмну платформу Node(<https://nodejs.org/en/download>)

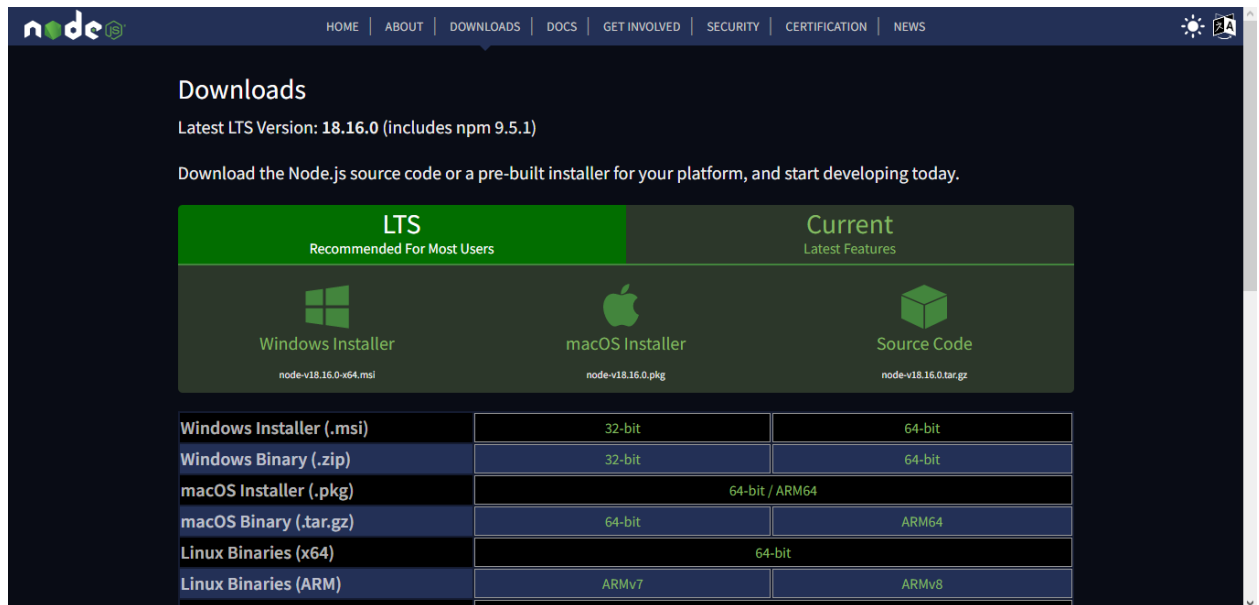


Рис. 2 – сторінка завантаження платформи Node.js

Потрібно встановити браузер для перегляду графічного інтерфейсу додатку.



## РОБОТА З ДОДАТКОМ

### Робота з додатком

Користувачеві для запуску програми необхідно відкрити папку програми в редакторі коду VS Code, яка містить файли додатку. Після цього необхідно виконати команду “npm install” в терміналі редактора коду. Ця команда встановить програмні пакети, необхідні для запуску програми.

Далі потрібно виконати команду “npm start” в терміналі редактора коду, яка і запустить додаток. Після нетривалого очікування, браузер відкриється самостійно і в ньому відкриється вкладка нашого додатка.

Керування програмою може здійснюватися виключно комп'ютерною мишею. Керування програмою може здійснюватися виключно комп'ютерною мишею. Це здійснюється через графічний інтерфейс за допомогою взаємодії із певними елементами сайту.

У разі виникнення помилок, потенційно їх можна переглянути в панелі інструментів розробника в браузері. Для цього необхідно, перебуваючи на сторінці додатку, клацнути правою кнопкою миші на будь-якому просторі сторінки. У контекстному меню знайти пункт, який відповідає за код елемента, і клацнути по ньому лівою кнопкою миші. Після відкриття панелі розробника в ній необхідно зайти у вкладку "Консоль" (“Console”). У разі помилок, більшість їх відобразиться у цьому просторі.