

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

бакалавр

(ступінь вищої освіти)

на тему ПРОГРАМНА РЕАЛІЗАЦІЯ ІНТЕРНЕТ-МАГАЗИНУ «LIKE PIZZA»
SOFTWARE IMPLEMENTATION OF THE LIKE PIZZA ONLINE STORE

(назва теми)

Виконала: студентка 4 курсу, групи КНТ-120сп

Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Інженерія програмного забезпечення

ГРЕВЦОВА О.А

(ПРИЗВИЩЕ та ініціали)

Керівник СТЕПАНЕНКО О.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ГОЛУБ Т.В.

(ПРИЗВИЩЕ та ініціали)

2023

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
« » 2023 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТКИ

Гревцової Олександри Антонівни
(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмна реалізація інтернет-магазину «Like Pizza».
Software implementation of the Like Pizza online store.

керівник проєкту (роботи) к.т.н., доцент СТЕПАНЕНКО Олександр Олексійович,
(науковий ступінь, вчене звання, ПРІЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від «4» квітня 2023 року № 87

2. Строк подання студентом проєкту (роботи) 1 червня 2023 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне
завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Характеристика об'єкта проєктування. 2. Вибір і обґрунтування
структури системи. 3. Основні рішення щодо реалізації компонентів системи.
4. Керівництво програміста. 5. Керівництво системного адміністратора.
6. Керівництво користувача.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	СТЕПАНЕНКО О.О., доцент		
Нормоконтроль	БЕЛОВА А.В., асистент		

7. Дата видачі завдання « 4 » квітня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи.	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області.	1 тиждень	Розділ 1
3	Вибір мови програмування та інших технологій розробки.	2 тиждень	Розділ 2
4	Розробка архітектури програми.	2 тиждень	Розділ 3
5	Розробка програми.	3-4 тижні	Розділ 3, 4
6	Тестування та експериментальне дослідження програмного забезпечення.	5 тиждень	Розділ 4
7	Оформлення пояснювальної записки та документів до неї.	6 тиждень	Додатки
8	Нормоконтроль та рецензування.	7 тиждень	
9	Захист роботи.	8 тиждень	

Студентка

_____ Олександра ГРЕВЦОВА.
(підпис) (Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи)

_____ Олександр СТЕПАНЕНКО
(підпис) (Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 93 с., 34 рис., 7 табл., 3 дод., 20 джерел.

ПІЦЦА, ІНТЕРНЕТ-МАГАЗИН, ПРОДАЖ, WEB-САЙТ, БАЗА ДАНИХ, ОПЕРАЦІЙНА СИСТЕМА WINDOWS, ВИТРАТИ НА РОЗРОБКУ ПРОГРАМНОГО ПРОДУКТУ, REACT.

Об'єкт розробки - Інтернет магазин із продажу різноманітних піц.

Мета роботи - розробка програмного продукту, який реалізує функції Інтернет магазину з індивідуальним наповненням та працює в мережі Інтернет.

Електронний магазин автоматизує процес вибору й покупки будь-яких піц, дозволяє користувачеві здійснювати покупки, робити замовлення, не виходячи з будинку, розширює клієнтську базу.

Вхідними даними для Інтернет-магазин є інформація про піци, які продаються в інтернет-магазині «Like Pizza».

Вихідними даними для Інтернет-магазин є сконфігуровані сторінки сайту з інформацією про категорії піц, які продаються в інтернет - магазині, й замовлення, зроблені користувачем.

Інтернет магазин розроблений мовою програмування React.js., серверна частина виконана на платформі NodeJs із застосуванням системи керування базами даних MongoDB. Електронний магазин коректно працює й відображається у всіх сучасних браузерях таких як Google Chrome, Safari, Opera і Firefox.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 93 pages, 34 figures, 7 tables, 3 appendixes, 20 sources.

PIZZA, ONLINE STORE, SALES, WEBSITE, DATABASE, SQL-QUERIES, WINDOWS OPERATING SYSTEM, COSTS OF SOFTWARE DEVELOPMENT, REACT.

The object of development is an online store selling a variety of pizzas.

The purpose of the work is the development of a software product that implements the functions of an online store with individual content and works on the Internet.

The electronic store automates the process of choosing and buying any pizzas, allows the user to make purchases, place orders without leaving home, and expands the customer base.

The input data for the online store is information about the pizzas sold in the Like Pizza online store.

The initial data for the online store are configured website pages with information about the categories of pizzas sold in the online store and orders made by the user.

The online store is developed in the React.js programming language, the server part is made on the NodeJs platform using the MongoDB database management system. The electronic store works correctly and is displayed in all modern browsers such as Google Chrome, Safari, Opera and Firefox.

ЗМІСТ

	С.
Перелік скорочень та умовних позначок	8
Вступ	9
1 Характеристика об'єкта проєктування.....	11
1.1 Поняття інтернет-магазину	11
1.2 Основні способи ведення бізнесу за допомогою системи онлайн-покупок	12
1.3 Основні функціональні можливості інтернет-магазину.....	13
1.4 Висновки за розділом	14
2 Вибір і обґрунтування структури системи.....	15
2.1 Вибір середовища програмування для серверної частини сайту.....	15
2.2 Вибір середовища програмування для клієнтської частини сайту	18
2.3 Обґрунтування вибору середовища розробки системи	25
2.4. Обґрунтування вибору середовища функціонування системи	26
2.5 Функціональна схема роботи інтернет-магазину	29
2.6 Інформаційні потоки	31
2.7 Висновки за розділом	32
3 Основні рішення щодо реалізації компонентів системи	33
3.1 Основні відомості про розробку Інтернет-магазину	33
3.2. Принцип роботи кошика товарів у інтернет-магазині	33
3.3 Алгоритм роботи блока, що містить посилання на кошик	38
3.4 Алгоритм роботи головної сторінки інтернет-магазину	40
3.5 Алгоритм роботи блока, що містить інформацію про властивості товарів.....	42
3.6 Головний файл в React	43
3.7 Алгоритм роботи кореневого компоненту в інтернет-магазині	44
3.8 Висновки за розділом	44
4 Керівництво програміста	46

4.1 Призначення й умови використання Інтернет-магазину з продажу піц	46
4.2 Характеристики розробленого продукту	47
4.3 Структура Json-файлу	49
4.4. Структурна схема компонентів інтернет-магазину	51
4.5.Висновки за розділом	53
5 Керівництво системного адміністратора	55
5.1 Розгортання інтернет-магазину	55
5.2 Вхідні і вихідні дані	55
5.3 Створення віртуального хоста на сервері	55
5.4 Висновки за розділом	56
6 Керівництво користувача	57
6.1 Призначення сайту	57
6.2 Виконання програми	57
6.3 Висновки за розділом	60
Висновки	61
Перелік джерел посилання	62
Додаток А Технічне завдання	64
Додаток Б Текст програми	67
Додаток В Слайди презентації	86

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

API – Application Programming Interface;

SPA – Single Page Application;

HTML – HyperText Markup Language;

HTTP – HyperText Transfer Protocol;

SPA – Single Page Application

REACTJS – React JavaScript;

DOM – Document Object Model;

VS CODE – Visual Studio Code;

БД – база даних;

НД – нормативна документація;

ОС – операційна система.

ВСТУП

Було б не вірно, якщо б ми не відзначили, що в наш час Інтернет відіграє велику роль у житті майже кожної людини.

Інтернет необхідний не тільки для роботи, комунікацій та ігор, але також і для оплати різних послуг. Люди зберігають багато часу, здійснюючи потрібні платежі через всесвітню мережу.

Велику роль відіграє інформативна частина Інтернету. З його допомогою люди з легкістю дізнаються багато нового і мають уявлення про останні події, як світу, так і рідного міста чи навіть селища. Але перше місце все частіше відводиться саме комунікативній ролі: електронна пошта, чати, месенджери або спілкування в соціальних мережах. Інтернет дає можливість спілкування між людьми з різних куточків світу. Тепер кожен може підтримувати постійний зв'язок із своїми рідними чи друзями незалежно від свого розташування. Сучасні люди спілкуються, працюють, заробляють та знайомляться за допомогою Інтернету. Ця мережа допомагає багатьом навіть у пошуках зниклих людей.

Інтернет справді може бути рівним орбіті, навколо якої обертається життя. Не менш важливу роль відіграють GPS-навігатори. За їх допомогою ви можете легко розпізнати, де знаходиться та чи інша вулиця, будівля у своєму або в іншому місті. Велику роль Інтернет в наш час відіграє і у освітньому процесі. Наразі велика частина навчальних закладів підключені до всесвітньої мережі. Інтернет надає доступ до великої кількості онлайн-курсів, які можуть бути відкриті для будь-якого бажаючого. Багато університетів також пропонують онлайн-курси та програми дистанційного навчання, які дозволяють студентам отримати вищу освіту, не виходячи з дому. Щодо бізнесу, Інтернет дозволяє створювати та просувати веб-сайти компаній, що дає можливість привернути більше клієнтів та розширити свій бізнес. Крім того, Інтернет дозволяє підприємцям знаходити нових партнерів,

клієнтів та постачальників, а також дізнаватися про нові тенденції та інновації у своїх галузях.

Інтернет став невід'ємною частиною життя. Люди все частіше використовують Інтернет для комунікації, пошуку інформації, розваг та покупок.

Web-сайт здійснює швидкий перегляд різних товарів, надає можливість отримання потрібної інформації, має функції пошуку даних за потрібним параметром.

Отже, Інтернет забезпечує людям зручність та швидкість в багатьох аспектах життя, тому його використання стало необхідністю для багатьох людей у сучасному світі.

1 ХАРАКТЕРИСТИКА ОБ'ЄКТА ПРОЄКТУВАННЯ

1.1 Поняття інтернет-магазину

Інтернет-магазин- це електронна платформа, яка дозволяє людям купувати товари та послуги через Інтернет. У інтернет-магазинах можна знайти та придбати все, що можна купити у звичайному магазині.

Переваги інтернет-магазину полягають у зручності та швидкості процесу покупки. Клієнт може зробити замовлення в будь-який час, з будь-якого місця з доступом до Інтернету. Вони не потребують фізичного візиту до магазину, що дозволяє заощадити час та зусилля.

Інтернет-покупки - процес замовлення та купівлі товарів або послуг через Інтернет. Вони стають все популярнішими, оскільки дозволяють зручно та швидко придбати потрібні товари.

Інтернет-магазини існують вже більше двадцяти років та їх популярність з кожним роком зростає.

Електронна комерція – процес купівлі і продажу товарів, послуг, інформації чи інших продуктів через інтернет .

Інтернет-магазини є частиною "Е-Commerce", що означає електронна комерція.

Покупки в Інтернеті відбуваються, коли клієнт купує через цифрову платформу. Прикладом такої платформи є Asos.com, онлайн-магазин одягу та взуття з Британії. Вам не потрібно жити в цій країні, щоб придбати один із товарів, які продає цей магазин. Глобальне покриття є однією з головних переваг онлайн-покупок.

Отже, інтернет-магазини та онлайн-замовлення у сучасному світі набирають обертів та стають все більш популярними та зручними ніж прості магазини [1].

1.2 Основні способи ведення бізнесу за допомогою системи онлайн-покупок

Система онлайн — покупок надає можливість підприємствам розширювати власний бізнес та просувати на нові ринки, забезпечуючи доступ до широкого кола потенційних клієнтів. Основні способи ведення бізнесу за допомогою систем онлайн-покупок включають:

- створення власного інтернет-магазину: підприємство може створити свій власний інтернет-магазин, на якому будуть розміщені всі товари та послуги, які вони пропонують. Це дає можливість клієнтам широко та зручно здійснювати покупки через Інтернет;
- розміщення товарів на інтернет – платформах: підприємство може розміщувати свої товари на різних інтернет-платформах, таких як Amazon, eBay та інші. Це дає можливість залучити нових клієнтів та розширити свій бізнес на нові ринки;
- використання соціальних медіа: підприємство може використовувати соціальні медіа, такі як Instagram, TikTok, Facebook та інші, для просування своїх товарів та послуг. Це дає можливість залучати нових клієнтів та збільшувати свою клієнтську базу;
- використання мобільних додатків: підприємство може створювати мобільні додатки для зручності клієнтів за збільшення продажів. Це дає можливість клієнтам здійснювати покупки зі своїх мобільних пристроїв та забезпечує легкий доступ до інформації про товари та послуги;
- залучення до онлайн-маркетингу: підприємці можуть використовувати онлайн-маркетинг для просування своїх товарів та послуг, включаючи рекламу на пошукових системах, соціальних мережах, блогах та інших сайтах [2].

1.3 Основні функціональні можливості інтернет-магазину

Інтернет-магазин є вебсайтом в якому розроблені функціональні можливості для автоматизації інтернет покупок користувача. При розробці інтернет-магазину розробник повинен звертати увагу на функціональність, зручність користування, і надійність. В розробленому інтернет магазині реалізовані наступні можливості:

- а) відображення списку піс доступних до покупки;
- б) зручний та швидкий пошук інформації за заданими критеріями;
- в) можливість придбання товару та додавання його у кошик;
- г) редагування інформації за:
 - 1) алфавітом;
 - 2) категорією;
 - 3) популярністю;
 - 4) ціною;
- д) формування чеку;
- е) редагування товарів у кошику.

Це перелік ключових можливостей інтернет-магазину.

Взагалі ринок сфери інформаційних технологій дуже стрімко розвивається і на сьогоднішній день коли існують такі технології як наприклад безконтактний спосіб оплати чи ідентифікація користувача за допомогою відбитку пальця, чи ідентифікації обличчя, перехід фізичного магазину до ІТ є найлогічнішим рішенням. Оскільки описані вище можливості дають змогу не тільки конкурувати з фізичним магазином, а й вирватися уперед. Принцип функціонування інтернет-магазину дає такі можливості:

- залишити відгуки про товари;
- ставити через електронну форму питання та отримати відповідь;
- замовлення товару в будь який день так будь який час;

- економити власнику на оренді та зарплаті персоналу;
- робити продаж товару без фактичної наявності.

Виходячи з того що продаючи товар ми робимо послугу для людей що його купують, а в послугі головне зручність та корисність, ми можемо зробити висновок що за допомогою перелічених переваг та можливостей, клієнту більше не потрібно спочатку шукати де йому потрібно купити той чи інший товар, потім стояти в черзі, та порівнювати ціни одного товару з іншими, чи питати відгуки у знайомих, а потрібно всього лиш перейти за інтернет-посиланням у зручний для користувача час, та ознайомитися з різноманітним асортиментом, за потреби поставити запитання в режимі онлайн чи прочитати відгуки, й врешті решт оформити замовлення обрав найзручніший спосіб оплати та доставки, знаходячись у будь-якій точці світу.

1.4 Висновки за розділом

Було проведено повноцінне дослідження об'єкта проектування, сформовано набір термінів, які б характеризували основні процеси електронної комерції та принципи функціонування інтернет-магазинів. Були розглянуті основні етапи ведення бізнесу за допомогою систем онлайн-покупок.

Також були розглянуті основні функціональні можливості створення інтернет-магазину, які в свою чергу дають перевагу над звичайними магазинами. На основі вищесказаного, було вирішено створити інтернет-магазин через впершу чергу актуальність, зручність та ефективність ведення бізнесу саме у режимі онлайн та можливість масштабування існуючого бізнесу за допомогою існуючих інструментів онлайн розробки.

2 ВИБІР І ОБҐРУНТУВАННЯ СТРУКТУРИ СИСТЕМИ

2.1 Вибір середовища програмування для серверної частини сайту

Середовищем функціонування сервера було обрано Node.js.

Нижче наведена таблиця 2.1 порівняння середовища функціонування Node.js та інших з деякими іншими середовищами та фреймворками.

Таблиця 2.1 – Порівняння середовища функціонування Node.js

Функція/особливість	Node.js	Python(Django)	Ruby
Середовище виконання	JavaScript заснований на Chrome V8	Python	Ruby
Розширення мови	Контекстне	Динамічне	Динамічне
Асинхронність	Підтримує асинхронну обробку	Підтримує асинхронну обробку	Підтримує асинхронну обробку
Спільнота	Велика активна	Велика активна	Велика активна
Швидкодія	Висока	Помірна	Помірна
Веб-фреймворки	Express, Koa, Napi	Django, Flask	Ruby on Rails
Складність вивчення	Легкий	Легкий	Помірно складний

З глобальним ростом маркетплейсів масштабованість для серверних додатків стала важливою як ніколи. Ось чому мій вибір пав на Node.js в якості платформи для серверного програмного забезпечення. У веб-розробці програма повинна взаємодіяти з кількома користувачами та обробляти великі обсяги даних. Для того щоб програма яка виконується на платформі Node.js була стійкою до відмов та масштабованою, критично важливо щоб розробники слідували стратегії не блокуючого вводу/виводу.

Основною відмінністю Node.js від інших технологій є використання однопотокової асинхронної моделі з циклом подій. В асинхронній моделі завдання виконуються по чергово в одному потоці. Це простіше, ніж у випадку потоків, оскільки програміст впевнений, що коли одне завдання виконується, в цей же час інше завдання чекає своєї черги. Хоча в однопроцесорній системі в потоковій моделі завдання все-одно будуть виконуватись по чергово, програміст, що використовує потоки, має мислити в термінах кількох незалежно виконуваних процесів. Натомість однопотокова асинхронна система завжди буде чергувати виконання завдань навіть на багатопроцесорній системі. Механізм подій допомагає серверу опрацьовувати запити не блокуючим способом і забезпечує високу масштабованість, в порівнянні з традиційними серверами, які для кожного нового запиту користувача створюють новий потік, що підвищує споживання пам'яті та ресурсів процесора.

Використання однопотокової моделі дозволяє Node.js обслуговувати набагато більше запитів, ніж традиційні сервери, такі як HTTP-сервер Apache. Всі бібліотеки Node.js є асинхронними. Це означає, що всі функції Node.js не блокують потік, а виконуються в фоновому режимі. Дана асинхронна модель краще підходить для програм, де багато користувачів одночасно виконують деякі дії, що не навантажують процесор. Тому сервери, написані на Node.js, підходять тільки для виконання завдань, які не є важкими обчислювальними процесами.

Сервери на Node.js добре виконують роль посередників в клієнт-серверній системі. На основі цих факторів можна зробити висновок що на даний час платформа Node.js є однією з найкращих серверних платформ для написання програм на JavaScript [3].

Переваги Node.js:

- висока продуктивність: Node.js використовує механізм однопотокової обробки подій, що дозволяє високої продуктивності, забезпечуючи швидке виконання коду;

- JavaScript: JavaScript є однією з найпоширеніших мов програмування, і Node.js дозволяє використовувати його як на клієнтській стороні, так і на сервері, що спрощує розробку веб-застосунків;
- обширна бібліотека: Node.js має велику бібліотеку модулів, які спрощують розробку застосунків, таких як Express, Socket.IO, Mongoose та багато інших;
- кросплатформність: Node.js може бути запущений на будь-якій операційній системі, що робить його зручним для розробки та розгортання застосунків на різних платформах;
- розширюваність: Node.js має велику гнучкість та можливості розширення, що дозволяє розробникам створювати власні модулі та бібліотеки, що спрощує процес розробки.

З Node.js пов'язано ще декілька цікавих особливостей:

- Non-blocking I/O: Node.js використовує асинхронну модель вводу-виводу(I/O), що дозволяє додаткам виконувати інші задачі, поки вони чекають на результати вводу-виводу, що забезпечує ефективне використання ресурсів сервера;
- Event-driven architecture: заснований на подіях, що дозволяє реагувати на події, такі як вхідні запити, вивід результатів, помилки і т.д., що дозволяє створювати швидкі та ефективні серверні застосунки;
- модульна структура: має модульну структуру, що дозволяє розробникам легко використовувати сторонні модулі та бібліотеки, зменшуючи час розробки та сприяючи повторному використанню коду;
- широке використання: використовується у багатьох веб-фреймворках, таких як Express, Meteor, Коа та багатьох інших, і має велику спільноту розробників, що забезпечує підтримку та розвиток платформи;
- розширені можливості: може бути розширений за допомогою додаткових модулів та бібліотек, що дозволяє створювати складні серверні застосунки, такі як чат-боти, системи обробки платежів, системи розподіленої обробки даних та багато інших.

Загалом, вибір Node.js може бути корисним для розробки високопродуктивних веб-застосунків, особливо для тих, хто вже знайомий з мовою JavaScript [4].

Webpack - це інструмент збірки модулів для JavaScript, що дозволяє об'єднувати окремі файли JavaScript у більш великі та оптимізовані файли. Він також підтримує роботу з іншими типами файлів, такими як CSS, зображення, HTML і т.д.

Webpack може бути використаний як в браузері, так і в середовищі Node.js. Використання Webpack в Node.js дозволяє збирати та оптимізовувати серверний додаток. Основна ідея Webpack полягає в тому, щоб об'єднати всі залежності додатку в один файл, який можна запустити на сервері. Це дозволяє зменшити кількість запитів, які необхідно виконувати на сервері, щоб отримати всі необхідні файли. З Webpack в Node.js можна використовувати різні типи модулів та залежностей, такі як CommonJS, ES6, AMD, CSS, зображення, JSON і т.д. Завдяки цьому ви можете збирати всі ваші модулі та залежності в один файл.

Крім того, Webpack дозволяє застосовувати різні типи оптимізації до вашого коду, такі як зменшення розміру файлів, мінімізація коду, додавання кешування та інші. Це допомагає зменшити час завантаження додатку та забезпечити більш швидку роботу.

Отже, Webpack є потужним інструментом збірки модулів для JavaScript, який може бути використаний як в браузері, так і в Node.js. Використання Webpack в Node.js дозволяє збирати та оптимізовувати ваш серверний додаток та забезпечити його більш швидку роботу.

2.2 Вибір середовища програмування для клієнтської частини сайту

Для розробки проекту було прийнято рішення використовувати Single Page Application (SPA).

Single Page Application - скорочено SPA, в перекладі означає «Додаток однієї сторінки». Іншими словами SPA – це web-додаток, розміщене на одній webсторінці, яка для забезпечення роботи завантажує весь необхідний код разом із завантаженням самої сторінки [5].

Плюси Single Page Application.

Першим плюсом варто відзначити той факт, що додатки на SPA відмінно працюють на пристроях як стаціонарних, так і мобільних. «Великі» комп'ютери, планшети, смартфони, і, в кінці-кінців, прості телефони (деякі) можуть безперешкодно працювати з сайтами побудованих за принципом SPA. Отже, перший «плюс» - робота на великій кількості пристроїв, а значить, створивши один додаток, ми отримуємо набагато більшу аудиторію користувачів ніж при використанні стандартного підходу.

Далі другий «плюс» – великий користувацький інтерфейс, так званий User Experience. Так як web-сторінка одна, побудувати багатий, насичений користувацький інтерфейс набагато простіше. Простіше зберігати інформацію про сеанс, управляти станами уявлень (views) і керувати анімацією (в деяких випадках).

Третій «плюс» – SPA істотно скорочує так звані «ходіння по колу», тобто завантаження одного і того ж контенту знову і знову. Так як наш портал (сайт) використовує шаблон, то разом з основним змістом будь-якої сторінки відвідувач сайту обов'язково завантажує розмітку шаблону. Так, кешування даних на даному етапі розвитку WWW досягло найвищих результатів, але якщо нічого кешувати, то і час, і ресурси на це не витрачаються.

Нижче в таблиці 2.2 приведені порівняння SPA та традиційних веб-додатків.

Таблиця 2.2 – Порівняння SPA та традиційних веб-додатків

Особливість	Односторінковий додаток	Традиційний веб-додаток
Навігація	Переходить між сторінками без перезавантаження	Кожна дія користувача викликає повторне завантаження сторінки
Взаємодія з сервером	Використовує API для отримання та відправки даних	Сервер забезпечує генерацію та відправку HTML
Швидкодія	Зменшена потреба у відправці та завантаження даних з сервера	Потребує повторного завантаження сторінки для кожної дії
Переносимість	Може працювати на різних пристроях та платформах	Залежить від підтримки браузерів та стандартів
Розробка	Вимагає використання фреймворків або бібліотеки, таких як React, Angular, Vue.js	Використовує традиційні мови програмування, такі як PHP, Java, .NET
Взаємодія з користувачем	Більш інтерактивний та близький для десктопних додатків	Зазвичай заснований на формах та обміні даними з сервером

Для реалізації SPA проекту необхідно використовувати «роутери». Для веб-проектів react-router-dom надає `<BrowserRouter>` і `<HashRouter>`. Основна відмінність між ними полягає в способі зберігання URL і взаємодії з сервером. `<BrowserRouter>` використовує звичайні URL. В цьому випадку URL виглядають як зазвичай, але потрібна певна настройка сервера. Зокрема, сервер повинен обслуговувати всі сторінки, які використовуються на клієнті.

Create React App підтримує це з коробки в режимі розробки і містить інструкції для правильного налаштування сервера [6]. На рисунку 2.2 зображена структура роутеру фільтрації товарів за категоріями.

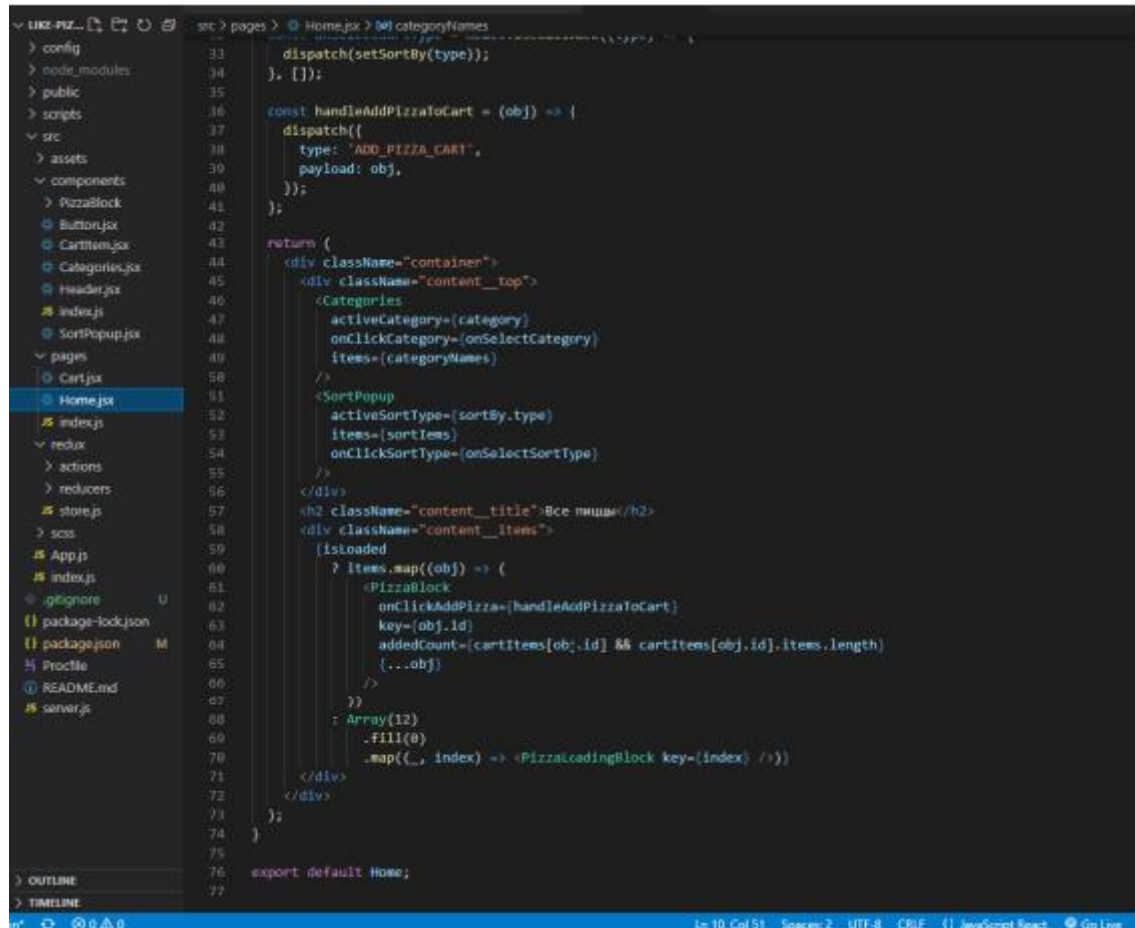


Рисунок 2.1 – Структура роутеру фільтрації товарів

На зображенні вище видно наступні теги роутерів:

- а) `<BrowserRouter>` - оголошує зону дії роутерів;
- б) `<Switch>` - оголошує вибірку першого збігу адреси і відображає контент з єдиного блоку route;
- в) `<Route>` - блоки які містять умову (збіг адреси з path) та дію яка буде виконана у випадку правдивості умови (тіло route);
 - 1) path - властивість route, яка зберігає в собі адресу для перевірки на збіг;

2) `exist` - властивість `route`, яка перевіряє `EndPoint` на точний збіг з адресою вказаною у властивості `path`;

г) `<Redirect>` - тег відповідає за переадресацію сайту, в разі відсутності збігів з усіма `route`;

1) `to` - властивість `redirect`, яка задає шлях переадресації [7].

Для розробки веб-сайту була використана бібліотека JavaScript React.

React – це бібліотека JavaScript, що використовується для розробки інтерактивних користувацьких інтерфейсів для веб-сайтів та веб-додатків.

React використовує віртуальну DOM (Document Object Model) для швидкого та ефективного оновлення UI, без необхідності повного оновлення сторінки [8].

DOM (Document Object Model) – це структура, що представляє структуру та контент сторінки веб-сайту як об'єкти, що можуть бути змінені та взаємодіяти з JavaScript -кодом. Кожен елемент сторінки, такий як заголовок, текст, зображення або кнопка, представлений як об'єкт, який можна маніпулювати через JavaScript. DOM складається з ієрархії об'єктів, де кожен елемент є нащадком об'єкту, що містить його, а кореневий елемент є об'єктом «document», що представляє всю сторінку [9].

DOM є важливим інструментом для веб-розробки та є частиною стандартів, таких як HTML та XML [9].

React також підтримує використання JSX – розширення мови JavaScript, що дозволяє писати HTML- подібний код в JavaScript- файлах, що спрощує розробку та робить код більш читабельним. Також дозволяє легко інтегруватися з іншими технологіями, такими як Redux, для управління станом додатка та Axios, для роботи з API. React використовується для побудови компонентів, які є незалежними блоками коду, що можуть бути перевикористані в різних частинах додатка [9].

Розроблюваний інтернет-магазин буде функціонувати на протоколі HTTP.

HTTP – широко поширений протокол передачі даних, спочатку призначений для передачі гіпертекстових документів (тобто документів, які можуть містити посилання, що дозволяють організувати перехід до інших документів).

Абревіатура HTTP розшифровується як HyperText Transfer Protocol, «протокол передачі гіпертексту». Протокол HTTP припускає використання клієнт-серверної структури передачі даних. Клієнтську програму формує запит і відправляє його на сервер, після чого серверне програмне забезпечення обробляє цей запит, формує відповідь і передає його назад клієнтові. Після цього клієнтську програму може продовжити відправляти інші запити, які будуть оброблені аналогічним чином. Сам по собі протокол HTTP не передбачає використання шифрування для передачі інформації. Проте, для HTTP є поширене розширення, яке реалізує упаковку переданих даних в криптографічний протокол SSL або TLS. Назва цього розширення - HTTPS (HyperText Transfer Protocol Secure). Для HTTPS-з'єднань зазвичай використовується TCP-порт 443.

HTTPS широко використовується для захисту інформації від перехоплення, а також, як правило, забезпечує захист від атак виду man-in-the-middle - в тому випадку, якщо сертифікат перевіряється на клієнті, і при цьому приватний ключ сертифіката ні скомпрометований, користувач не підтверджував використання непідписаного сертифіката, і на комп'ютері користувача не були впроваджені сертифікати центру сертифікації зловмисника [10]. На рисунку 2.2 зображена структура компонента Link, який містить посилання на інформацію про під.

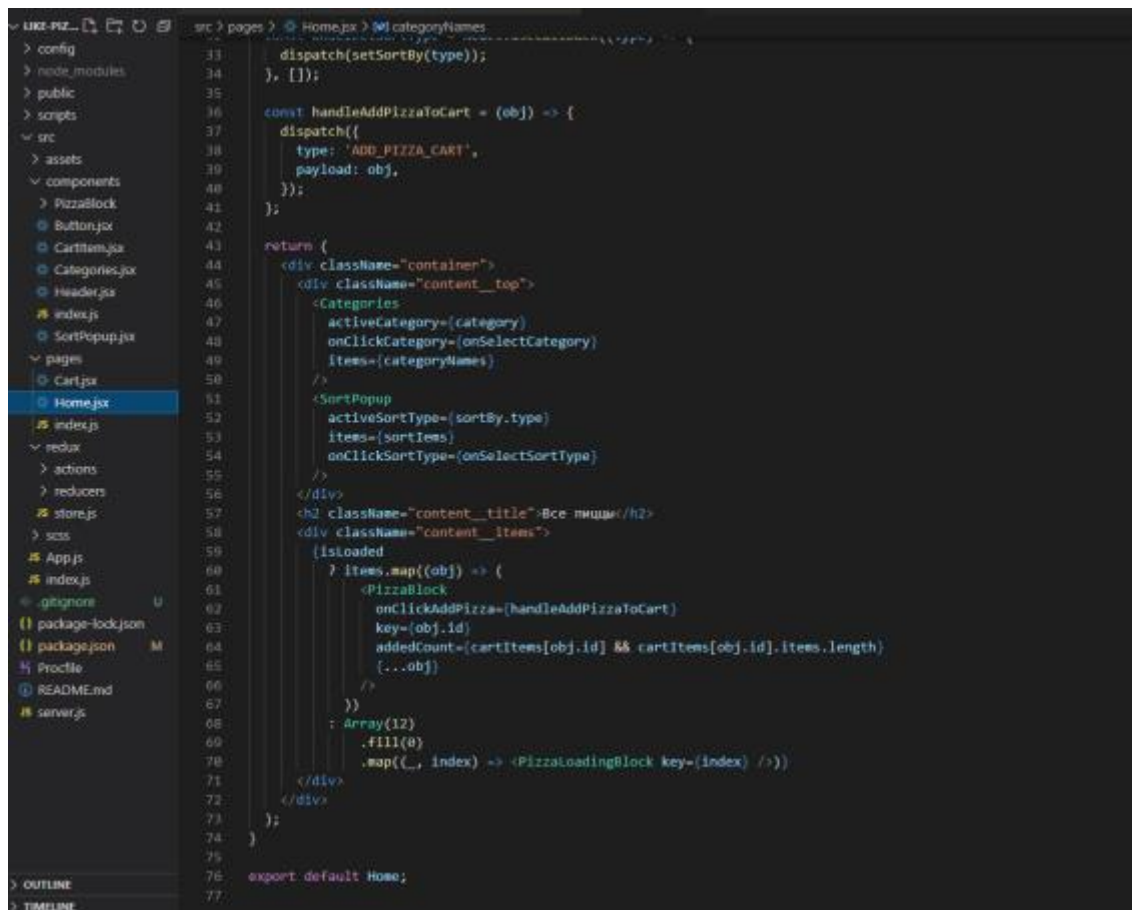


Рисунок 2.2 – Структура компонента Link

При розробці проекту неодноразово використовуються теги HTML.

Базові теги:

– `<!DOCTYPE>` – показує браузеру тип документа, повідомляє його версію і мову;

– `<Html> </html>` – кореневий тег, який повідомляє браузеру, що це HTML-документ. Всі інші елементи містяться всередині нього;

– `<Head> </head>` – контейнер, в який поміщаються метадані документа, не видимі користувачам, але зчитуються пошуковими роботами: наприклад, `<title>` або `<style>`;

– `<Meta>` – тег для оформлення метаданих документа, використовуваних браузером для обробки сторінки, а пошуковими системами - для індексації;

– `<Body> </body>` – тег, що обрамляє видиму користувачам частину документа. Все, що ви вкажете всередині цього контейнера, відобразиться на сторінці;

Форматування тексту:

– `<H1> </h1>.....<h6></h6>` – теги заголовків, від найбільшого до найменшого;

– `<B></b>` - жирний текст без додання важливості виділеного фрагмента;

– `<p></p>`- контейнер для абзацу;

– `<Br>`- переносить текст на інший рядок без створення абзацу;

– `<a></a>`- додавання гіперпосилання в текст. Має обов'язковий атрибут `href`, в якому вказується посилання або якір. Всередині контейнера міститься текст, при натисканні на який відбувається перехід на іншу сторінку або інше місце на цій же сторінці [11].

2.3 Обґрунтування вибору середовища розробки системи

Для зручної та більш продуктивної розробки проєкту була обрана програма Visual Studio Code (VS Code).

Visual Studio Code є популярним редактором вихідного коду, який багато розробників використовують для розробки веб-додатків. Основні його особливості включають:

– кросплатформеність: VS Code підтримує роботу на Windows, Linux і MacOS, що дозволяє розробникам працювати на будь-якій обраній платформі;

– легкість та продуктивність: Редактор відомий своєю швидкістю та ефективністю роботи. Запускається швидко та має малі вимоги до ресурсів комп'ютера.

– вбудовані інструменти: VS Code поставляється з вбудованим вкладчиком, інструментами для роботи з Git (версійним контролером), підсвічуванням синтаксису та засобами для рефакторингу коду;

– відкритий вихідний код: VS Code розробляється як програмне забезпечення з відкритим кодом, що дозволяє розробникам внести свої внески, вивчити його роботу та пристосувати до своїх потреб.

Нижче наведена таблиця 2.3 порівняння Visual Studio Code з іншими редакторами.

Таблиця 2.3 – Порівняння Visual Studio Code з іншими редакторами

Редактор	Кросплатформеність	Відкритий вихідний код	Підтримка Git	Вбудований відкладчик
VS Code	Так	Так	Так	Так
Sublime Text	Так	Ні	Так	Ні
Atom	Так	Так	Так	Ні
Visual Studio	Так	Ні	Так	Так

2.4 Обґрунтування вибору середовища функціонування системи

Для середовища функціонування будь-якого веб-додатку використовуються веб-браузери. Сьогодні на кожному ПК, ноутбучі чи іншому пристрої з можливістю виходу в інтернет є браузер.

Веб-браузер - це програмне забезпечення, що дозволяє переглядати веб-сторінки в Інтернеті. Він є ключовим компонентом Інтернету, оскільки

дозволяє користувачам швидко і легко отримувати доступ до сайтів, взаємодіяти з ними та спілкуватися в Інтернеті (месенджери, пошта).

Середовищем функціонування програми є Windows, Mac OS X, Linux, OS / 2 та інші операційні системи при наявності браузера. Створюваний проект буде розрахований на найпопулярніші браузери в даний момент, а саме Google Chrome, Safari, Opera і Firefox.

Google Chrome – це веб-браузер, який дозволяє переглядати веб-сторінки в Інтернеті. Він є найпопулярнішим серед інших браузерів. Був розроблений компанією Google. Chrome має декілька корисних функцій, які допомагають користувачам більш комфортно користуватися Інтернетом:

- швидкість: Chrome працює дуже швидко, завантажуючи веб-сторінки за декілька секунд;
- зручний пошук: в Chrome є вбудована пошукова строка, яка дозволяє здійснювати пошук в Інтернеті безпосередньо з браузера;
- закладки: користувачі можуть зберігати свої улюблені веб-сторінки у вигляді закладок і повертатися до них у будь-який час;
- синхронізація: користувачі можуть синхронізувати свої закладки, історію та інші дані між різними пристроями, які вони використовують для користування браузером;
- захист від шкідливих програм: має вбудований захист від шкідливих програм та сайтів, що допомагає захистити користувача від можливих загроз.

Опера – це веб-браузер, який дозволяє переглядати веб-сторінки в Інтернеті. Був розроблений компанією Opera Software і також став одним з популярних серед користувачів.

Основні функції Опера:

- швидкість: Опера працює дуже швидко і може завантажувати веб-сторінки за декілька секунд;
- безпека: Опера має вбудований блокувальник реклами та захист від відстеження;

- підтримка відео та аудіо: Opera підтримує відео та аудіо високої якості;
- режим Turbo: Opera має функцію «Турбо-режим», яка дозволяє стиснути веб-сторінки та скоротити час їх завантаження;
- VPN: Opera має вбудований VPN, який дозволяє користувачам захищено переглядати веб-сторінки;
- менеджер завантажень: Opera має вбудований менеджер завантажень, який дозволяє користувачам легко завантажувати файли з Інтернету [12].

Mozilla Firefox – також веб-браузер. Був розроблений Фондом Mozilla.

Safari – це веб-браузер, який розробляється компанією Apple і постачається з пакетом програм для операційної системи macOS та iOS.

Основні функції:

- швидкість: працює швидко і може завантажувати веб-сторінки за кілька секунд;
- читання: має вбудовану функцію «Читати», що дозволяє користувачам зручно читати статті в Інтернеті, видаляючи зайві елементи зі сторінки та пристосовуючи розмір тексту до власних потреб;
- Airplay: користувачі можуть використовувати функцію Airplay для стримінгу відео та аудіо з Safari на телевізор або будь-який інший пристрій;
- iCloud: Safari інтегрований з iCloud, що дозволяє користувачам зберігати свої закладки, паролі та історію перегляду на серверах Apple та отримувати до них доступ з будь-якого пристрою;
- відео: підтримує відео високої якості та має вбудований плеєр для відтворення відео безпосередньо в браузері.

Safari є надійним та швидким браузером, який добре інтегрується з операційною системою macOS та iOS. Він має багато корисних функцій, що допомагають користувачам зручно переглядати веб-сторінки та захищати свою приватність.

Існують версії для Windows, Linux, OS/2, Mac OS X, Solaris й інших операційних систем.

Нижче в таблиці 2.4 представлена інформація про можливості браузерів.

Таблиця 2.4 – Функціональність браузерів

Можливості	Safari	Mozilla FireFox	Opera	Chrome
Вкладки	Так	Так	Так	Так
Блокування pop-up	Так	Так	Так	Так
Блокування Банерів	Так	Так	Частково	Частково
Перевірка Орфографії	Так	Так	Так	Так
Панель пошуку	Так	Так	Так	Так
Менеджер Закачувань	Так	Так	Так	Так

2.5 Функціональна схема роботи інтернет-магазину

Функціональна схема роботи інтернет-магазину складається з наступних компонентів:

- веб-сторінка, на якій клієнти можуть переглядати меню, обирати страву, здійснювати замовлення та оплату;
- сервер, який зберігає дані про товари, замовлення, користувачів та обробляє запити клієнтів;
- Json-файли, в яких зберігається інформація про товари, замовлення, користувачів, та інші дані, пов'язані з роботою магазину;
- платіжна система, яка забезпечує безпеку платежів за замовлення;
- система доставки, яка забезпечує доставку товарів до клієнтів;

- система управління запасами, яка допомагає відстежувати наявність товарів на складі та керувати запасами товарів.

Всі ці компоненти працюють разом, щоб забезпечити зручне та швидке замовлення та доставку товарів користувачам інтернет-магазину.

Алгоритм роботи інтернет-магазину складається з наступних дій:

- покупець входить на веб-сторінку інтернет-магазину, обирає якийсь товар, та потім або одразу реєструється;
- далі товар додається у кошик і формується замовлення;
- оператор передзвонює для підтвердження замовлення;
- товар оплачується.

Це алгоритм функціонування інтернет-магазинів з продажу будь-якої їжі.

Якщо це інтернет-магазин з якимось іншим товаром, наприклад спортивні товари, або одяг та взуття, то відбувається трохи інший алгоритм, але схожий на попередній:

- також йде реєстрація та перегляд товарів на сайті;
- товар додається у кошик та формується замовлення;
- замовлення було прийнято магазином;
- товар резервується на складі та формується до відправки після оплати.

На рисунку 2.3 наведеному нижче наведена функціональна схема роботи інтернет-магазину з продажі піц.

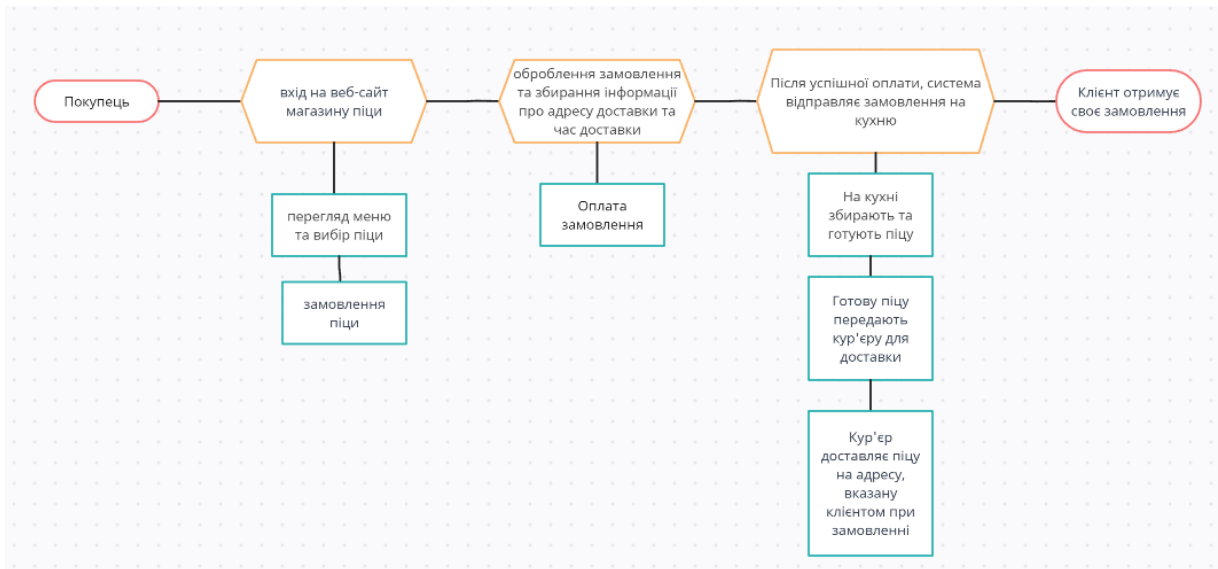


Рисунок 2.3 – Функціональна схема роботи інтернет-магазину з продажі піц

2.6 Інформаційні потоки

При роботі будь-якого сайту існують два типи інформаційних потоків: зовнішні та внутрішні.

Зовнішні потоки пов'язані з користувачем, який взаємодіє з сайтом, наприклад, шляхом роботи з формами, кнопками та іншими елементами на сторінках сайту. Користувачі можуть надсилати запити на сервер, щоб отримати різну інформацію, наприклад, сторінки, файли або відповіді на запити.

З іншого боку, внутрішні потоки пов'язані з сервером, який обробляє запити користувачів та надсилає відповіді назад. Користувач, який отримує дані з сервера, може бачити сторінки, виконувати різні дії та взаємодіяти з сайтом.

Отже, внутрішні та зовнішні інформаційні потоки є важливими компонентами роботи будь-якого сайту, які дозволяють користувачам взаємодіяти з сайтом та отримувати необхідну інформацію [13].

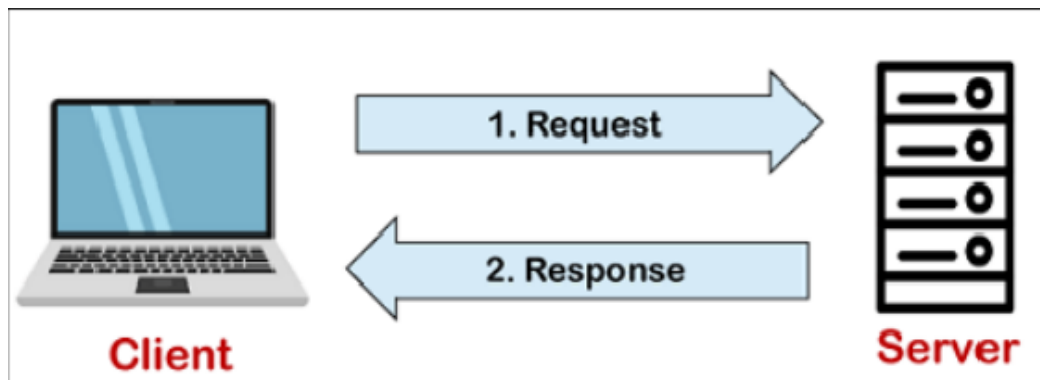


Рисунок 2.4 – Інформаційні потоки

2.7 Висновки за розділом

В данному розділі було обрано середовище розробки, що відповідає критеріям, зазначеним при постановці завдання, та сформований необхідний набір інструментів для вирішення проєктного завдання.

Було проведено обґрунтування вибору мови програмування та середовища функціонування системи. Для розробки проєкту було обрано платформу Node.js для розробки серверної частини сайту, React, HTML та CSS було обрано для клієнтської частини інтернет магазину. В якості бази даних було обрано Json-файли.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Основні відомості про розробку Інтернет-магазину

Інтернет-магазин з продажу піц дає можливість покупцям здійснювати замовлення будь-яких піц у режимі онлайн, а адміністраторам веб-сайту додавати та редагувати страви також онлайн.

У ході розробки веб-сайту було використано такі мови програмування як ReactJS, Node.js, також неодноразово використовувалась мова розмітки HTML з таблицями каскадних стилів CSS.

Також для будь-якого інтернет-магазину необхідна база даних, то в моєму випадку використовувались JSON- файли.

Плюси використання таких файлів:

- простота використання (легкість в читанні та написанні, редагуванні даних);
- не має залежності від якихось модулів. Не треба якийсь модуль, який працює в якості провайдера між базою даних та кодом;
- працює нативно на рівні вбудованого JS API роботи з JSON – файлами;
- компактність (дозволяють зберігати більше даних на обмін);
- JSON формат часто використовується для передачі даних з сервера на клієнт у веб-розробці, що дозволяє легко взаємодіяти з даними на веб-сторінці.

3.2 Приципи роботи кошика товарів у інтернет-магазині

Кошик в різноманітних інтернет-магазинах є важливою функцією, яка дозволяє користувачам збирати товари, які вони хочуть придбати, перед оформленням замовлення. Це дозволяє покупцю зручно обирати товари та порівнювати їх, не переходячи зразу до оформлення замовлення.

Основні переваги кошика в інтернет-магазинах:

- зручність: користувачі можуть додавати товари до кошика в будь-який час, доки не будуть готові зробити покупку. Вони можуть також змінювати кількість товарів, видаляти їх або додавати;
- час: кошик зберігає товари, які користувачі додав до нього, що дозволяє зберігати час при повторному відвідуванні магазину та замовленні товарів;
- контроль витрат: кошик дозволяє користувачам відстежувати загальну вартість товарів, що вони додали до кошика, допомагаючи контролювати свої витрати та уникнути непередбачуваних витрат.

Створення кошика було створено таким чином та представлений на даному фрагменті програми:

```
import React from 'react';
import { useSelector, useDispatch } from
'react-redux';
import { Link } from 'react-router-dom';
import      cartEmptyImage      from
'../assets/img/empty-cart.png';
import  {  CartIt,  Button  }  from
'../components';
import { clearCart, removeCartIt, plusCartIt,
minusCartIt }
from '../redux/actions/cart';
function Cart() { const dispatch =
useDispatch();
  const { totalPrice, totalCount, items } =
useSelector(({ cart }) => cart);
  const      addedPizzas      =
Object.keys(items).map((key) => { return
items[key].items[0]; });
  const onClearCart = () =>
```

```
{if (window.confirm(Ви дійсно хочете
очистити корзину?)) {
  dispatch(clearCart()); }
};
const onRemoveItem = (id) =>
{if (window.confirm('Ви дійсно хочете
видалити?')) { dispatch(removeCartIt(id));
}
};
const onPlusItem = (id) => {
  dispatch(plusCartIt(id));
};
const onMinusItem = (id) =>
{ dispatch(minusCartIt(id)); };
const  onClickOrder  =  ()  =>  {
  console.log('ВАШЕ ЗАМОВЛЕННЯ',
items); };
  return (
<div className “container container—
cart”>
```

```
{totalCount ? (
<div className = "cart">
<div className = "cart_top">
<h2 className = "content__title">
<svg
Width="18"
Height = "18"
viewBox="0 0 18 18"
fill-"none"
xmlns="http://www.w3.org/2000/svg">
<path
D = "M6.33333"
16.3333C7.06971 16.3333 7.66667 15.7364
7.66667 15C7.66667 14.2636 7.06971
13.6667 6.33333 13.6667C5.59695 13.6667
5 14.2636 5 15C5 15.7364 5.59695 16.3333
6.33333 16.3333Z"
stroke="white"
strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M14.3333
16.3333C15.0697 16.3333 15.6667 15.7364
15.6667 15C15.6667 14.2636 15.0697
13.6667 14.3333 13.6667C13.597 13.6667
13 14.2636 13 15C13 15.7364 13.597
16.3333 14.3333 16.3333Z"
stroke="white"
strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M4.78002
```

```
4.99999H16.3334L15.2134
10.5933C15.1524 10.9003 14.9854 11.176
14.7417 11.3722C14.4979 11.5684 14.1929
11.6727 13.88 11.6667H6.83335C6.50781
11.6694 6.1925 11.553 5.94689
11.3393C5.70128 11.1256 5.54233 10.8295
5.50002 10.5067L4.48669
2.82666C4.44466 2.50615 4.28764 2.21182
4.04482 1.99844C3.80201 1.78505 3.48994
1.66715 3.16669 1.66666H1.66669"
stroke="white"
strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round" />
Кошник
<h2/>
<div className = "cart_clear">
<svg
Width = "20"
Height ="20"
viewBox= "0 0 20 20"
fill="none"
xmlns="http://www.w3.org/2000/svg">
d ="M2.5 5H4.16667H17.5"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M6.66663
5.00001V3.33334C6.66663 2.89131
6.84222 2.46739 7.15478 2.15483C7.46734
1.84227 7.89127 1.66667 8.33329
1.66667H11.6666C12.1087 1.66667
12.5326 1.84227 12.8451 2.15483C13.1577
```

```

2.46739    13.3333    2.89131    13.3333
3.33334V5.00001M15.8333
5.00001V16.6667C15.8333    17.1087
15.6577 17.5326 15.3451 17.8452C15.0326
18.1577    14.6087    18.3333    14.1666
18.3333H5.83329C5.39127    18.3333
4.96734 18.1577 4.65478 17.8452C4.34222
17.5326    4.16663    17.1087    4.16663
16.6667V5.00001H15.8333Z"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M8.33337 9.16667V14.1667"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M11.6666 9.16667V14.1667"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"/>
</svg>
<span
onClick={onClearCart} Очистити
кошик</span>
<div>
</div>
<div
className="content_items">
{addedPizzas.map((obj) =>
( <CartIt

```

```

key={obj.id}
id={obj.id}
name={obj.name}
type={obj.type}
size={obj.size}
totalPrice={items[obj.id].totalPrice}
totalCount={items[obj.id].items.length}
onRemove={onRemoveItem}
onMinus={onMinusItem}
onPlus={onPlusItem}/> )) }
<div className="cart_bottom">
<div className="cart_bottom-details">
<span>
Всього піц: <b>{totalCount}</b> шт.</b>
</span>
<span>
Сума замовлення: <b> {totalPrice} ₴</b>
</span>
</div>
<div className="cart_bottom-buttons">
<a href = "/" className="button button—
outline button—addgo-back-btn">
<svg
width="8"
height="14"
viewBox="0 0 8 14"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M7 13L1 6.93015L6.86175 1"
stroke="#D3D3D3"
strokeWidth="1.5"
strokeLinecap="round"
strokeLinejoin="round"
/>

```

```

</svg>
<Link to= "/">
<span> Повернутися назад</span>
</Link>
</a>
<Button
onClick={onClickOrder} className= "pay-
btn"
<span> Оплатити зараз</span>
</button>
</div>
</div>
</div>
): (
<div className ="cart cart--empty">
<h2>
Кошик порожній<i> 😞 </i>
</h2>
<p>
Наймовірніше всього, ви не замовляли
ще піцу.
<br/>
Для того, щоб замовити піцу перейдіть на
головну сторінку.
</p>
<Link to ="/" className= "button button--
black">
<span>Повернутися назад</span>
</Link>
</div>
)}
</div>
);
}

export default Cart;

```

Оформлення замовлення в магазині «Like Pizza» починається з вибору тіста та розміру обраної піци. Клієнт може вибрати одну або декілька піц.

Блок-схему роботи кошика зображено нижче на рисунку 3.1.



Рисунок 3.1 – Схема роботи кошику інтернет-магазину

3.3 Алгоритм роботи блока що містить посилання на кошик

Компонент, який містить посилання на кошик в інтернет магазині, є важливою частиною функціональності будь-якого інтернет-магазину. Він дозволяє користувачам легко додавати товари до кошика та переглядати їх кількість та загальну вартість, а також здійснювати операції замовлення. Цю функцію виконує компонент «Button».

Цей компонент містити іконку кошика та посилання на окрему сторінку з кошиком, де користувач може переглянути вміст свого кошика та здійснити оплату. Компонент «Button» розміщується у верхній частині сторінки, щоб користувачі могли легко знайти його та бути в курсі свого поточного стану замовлення.

Компонент «Button» включає в себе логіку для зберігання даних про товари в кошику та їх кількість, а також логіку для додавання, видалення та оновлення товарів у кошику. Цей компонент також містить зовнішній вигляд, який включає іконку кошика, що залежить від потреб інтернет-магазину.

Отже, компонент «Button», який містить посилання на кошик, є важливою складовою функціональності будь-якого інтернет-магазину, і він

дозволяє користувачам легко керувати своїми покупками та здійснювати замовлення в магазині.

Приклад коду роботи компонента «Button» наведений нижче:

```
import React from 'react';
import PropTypes from 'prop-types';
import classNames from 'classnames';
const Button = ({ onClick, className, outline, children }) => {
  return (
    <button
      onClick={onClick}
      className={classNames('button',
        className, {
          'button--outline': outline,
        })}
    >
      {children}
    </button>
  );
};
Button.propTypes = {
  onClick: PropTypes.func,
};
export default Button;
```

Блок-схема роботи компонента, що містить посилання на кошик приведена на рисунку 3.2:



Рисунок 3.2 – Блок-схема роботи компонента «Button»

3.4 Алгоритм роботи головної сторінки інтернет-магазину

При запуску сайту починає працювати компонент «Home», який містить в собі головний блок магазину. Інтернет-магазин виконує код цього компоненту та передає результат користувачу у вигляді скомпільованої сторінки.

Компонент «Header» містить в собі «дах» сайту, з логотипом.

На початку виконання компонента «Home» проводиться певний аналіз адресного рядка, робиться запит на EndPoint, який зчитує контент файлу та відправляє його на клієнт та формується сторінка на основі інформації з JSON-файлів. Схематична блок-схема генерації головної сторінки зображено на рисунку 3.3.

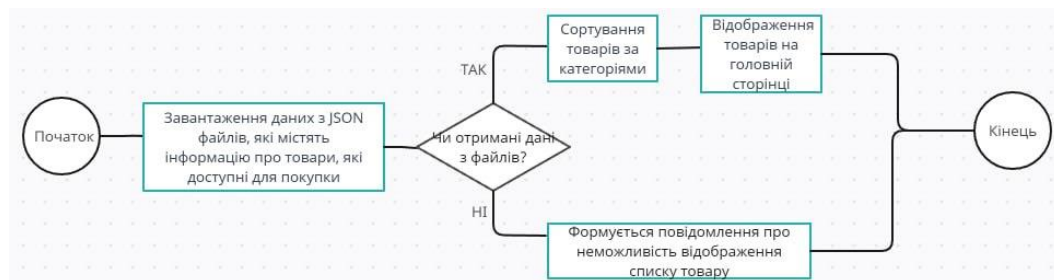


Рисунок 3.3 – Схема роботи головної сторінки інтернет-магазину

Нижче приведений код роботи компоненту «Home», який відповідає за алгоритм роботи головної сторінки.

```

import React from 'react';
import { useSelector, useDispatch } from 'react-redux';
import { Categories, SortPop, PizaBlock, PizzaLoadingBlock }
from '../components'; import { setCategory,
setSortBy }
from '../redux/actions/filters'; import {
fetchPizzas }
from '../redux/actions/pizzas';

```

```

import { addPizzaToCart } from
'../redux/actions/cart';
const categoryNames = ['М'ясні',
'Вегетаріанські', 'Гострі'];
const sortItems = [
{ name: 'популярності', type: 'popular',
order: 'desc' },
{ name: 'ціни', type: 'price', order: 'desc' },
{ name: 'алфавіт', type: 'name', order: 'asc'
}, ];

```

```

function Home() { const dispatch =
useDispatch();
const items = useSelector(({ pizzas }) =>
pizzas.items);
const cartIts = useSelector(({ cart }) =>
cart.items);
const isLoading = useSelector(({ pizzas })
=> pizzas.isLoading);
const { category, sortBy } = useSelector(({
filters }) => filters);
React.useEffect(() => {
dispatch(fetchPizzas(sortBy, category)); },
[category, sortBy]); const onSelectCategory
= React.useCallback((index) =>
{ dispatch(setCategory(index)); }, []); const
onSelectSortType =
React.useCallback((type) => {
dispatch(setSortBy(type)); }, []);
const handleAddPizzaToCart = (obj) =>
{ dispatch({ type: 'ADD_PIZZA_CART',
payload: obj, }); };
return (
<div className="container">
<div className= "content__top">
<Categories activeCategory={category}
onClickCategory={onSelectCategory}
items={categoryNames} />
<SortPop
activeSortType={sortBy.type}
items={sortItems}
onClickSortType={onSelectSortType} />
</div>
<h2 className ="content_title" Всі піци
</h2>
<div className= " content_items">
{isLoading ? items.map((obj) => (
<PizzaBlock
onClickAddPizza={handleAddPizzaToCart}
key={obj.id}
addedCount={cartIts[obj.id] &&
cartIts[obj.id].items.length} {...obj} /> ))
: Array(12) .fill(0) .map((_,index)=> )}
); } export default Home;

```

3.5 Алгоритм роботи блока, що містить інформацію про товари

Компонент «Categories» містить в собі інформацію про властивості товарів. Такі як розмір, склад категорії.

Блок-схема роботи компонента «Categories» зображений на рисунку 3.4:



Рисунок 3.4 – Блок-схема компонента «Categories»

Нижче приведений код роботи компоненту «Categories», який відповідає за алгоритм роботи блока, що містить інформацію про властивості товарів.

```

import React from 'react';
import PropTypes from 'prop-types';
const Categories = React.memo(function Categories({ activeCategory, items, onClickCategory })
{
  return
  <div className = "categories">
  <ul>
  <li
  className={activeCategory === null ? 'active' : ""}
  onClick={() =>
  onClickCategory(null)}> Все </li>
  {items && items.map((name, index) => (
  <li
  className={activeCategory === index ? 'active' : ""}
  onClick={() => onClickCategory(index)}
  key={`_${name}_${index}`}> {name}
  ))}
  );
  });
Categories.propTypes = { items:
PropTypes.arrayOf(PropTypes.string).isRequired, onClickCategory: PropTypes.func.isRequired,
};
Categories.defaultProps={ activeCategory: null, items: [] };
export default Categories;

```

3.6 Головний файл в React

Файл «index.js» – це точка входу для додатку React. Він є головним файлом, який виконується при запуску додатку. У цьому файлі зазвичай імпортується кореневий компонент веб-сайту та рендерується в HTML сторінку.

У стандартному додатку React, створеному за допомогою Create React App, файл знаходиться у папці «src».

Метод «ReactDOM.render()» відповідає за відображення нашого кореневого компоненту у HTML сторінці. У цьому методі ми передаємо кореневий компонент «App» як дочірній елемент методу «React.StrictMode», який використовується для виводу повідомлень про застарілі методи React у консоль, щоби ми могли їх виправити. Ми також передаємо HTML-елемент з ідентифікатором «root», який відповідає за місце, де наш додаток буде рендеритись.

Таким чином, «index.js» відповідає за налаштування і рендерінг кореневого компоненту React в HTML сторінці.

Приклад коду головного файлу розробки інтернет-магазину.

```
import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './redux/store';
import './scss/app.scss';
import App from './App';
ReactDOM.render(
  <Router>
    <Provider store = {store}>
      <App/>
    </Provider>
  </Router>
  document.getElementById('root'), );
```

3.7 Алгоритм роботи кореневого компоненту інтернет-магазину

«App.js» - це кореневий компонент React додатку, який містить весь додаток та який буде відображатися на сторінці. В цьому файлі міститься код, який використовується для відображення компонентів та організації логіки вашого додатку.

Зазвичай, в «App.js» ми імпортуємо необхідні компоненти та бібліотеки, використовуючи директиву «import», після чого ми визначаємо функцію-компонент, яку ми експортуємо за замовчуванням. Ця функція-компонент повинна повернути JSX - який є мовою розмітки для React компонентів.

Таким чином, «App.js» відіграє ключову роль у відображенні та організації логіки React додатку, і він буде постійно редагуватися залежно від потреб додатку.

Приклад коду головного файлу розробки інтернет-магазину.

```
import React from 'react';
import { Header } from './components';
import { Home, Cart } from './pages';
import { Route } from 'react-router-dom';
function App() {
  return (
    <div className="wrapper">
      <Header/>
      <div className="content">
        <Route path="/"
          component={Home} exact />
        <Route path="/cart"
          component={Cart} exact />
      </div>
    </div>
  );
}
export default App;
```

3.8 Висновки за розділом

У цьому розділі наведений опис основної інформації про розробку інтернет-магазину.

Розроблені основні алгоритми, що були використані під час роботи з розробки веб-сайту – а саме алгоритм роботи кошика, алгоритм роботи компонента, що містить посилання на кошик, алгоритм роботи компонента, що містить властивості товарів та алгоритм роботи головної сторінки магазину. Також було розглянуто значення головного файлу та кореневого компоненту в інтернет-магазині.

Зроблено загальну характеристику створеного додатку та розглянуті основні процедури та блоки виконання.

4 КЕРІВНИЦТВО ПРОГРАМІСТА

4.1 Призначення й умови використання Інтернет-магазину з продажу піц

Веб-сайт призначений для відображення даних про піци та створення замовлень.

Внесення змін в проєкт відбуваються в через адміністративну панель, розроблену з допомогою запитів HTTP.

Для запуску в режимі розробки треба відкрити проєкт в Visual Studio Code.

Веб-сайт є односторінковим.

На головній сторінці ви можете знайти:

- логотип;
- інформацію про піци ;
- кошик.

На сторінці кошику ви можете знайти:

- інформацію про замовлені піци;
- інтерфейс для зміни кількості замовлених піц;
- кнопка «на головну».

Серверна частина проєкту була розроблена за допомогою платформи NodeJS. У якості БД використаний json-файл з внесеним до нього переліком піц.

Умовами експлуатації сайту є наявність надійного з'єднання з мережею інтернет, та наявність усіх модулів веб-застосунку та бази даних на веб-сервері.

Для роботи веб-застосунку потрібен комп'ютер або ноутбук пристроями введення та виведення інформації – клавіатура та миша, або будь-який інший пристрій з можливістю відображення веб-сторінок, та обов'язково з виходом до мережі інтернет, з попередньо встановленим браузером: Google Chrome, Safari, Opera, Mozilla Firefox та інші.

4.2 Характеристики розробленого продукту

Сайт може виконувати наступні функції:

- перегляд інформації про піци;
- створення замовлення;
- редагування інформації у кошику.

Для розробки даного web-додатку використано дві моделі даних: об'єктноорієнтована та документо-орієнтована. Документо-орієнтовану модель було побудовано шляхом перетворення об'єктно-орієнтованої моделі бази даних.

Об'єктно-орієнтована модель – це підхід до програмування, в якому дані та функції, що їх обробляють, групуються в об'єкти. Кожен об'єкт має свої властивості та може взаємодіяти з іншими об'єктами. Вони можуть бути створені на основі абстрактних даних, і методи можуть бути використані для взаємодії з іншими об'єктами та навколишнім світом. Об'єктно-орієнтована модель дозволяє розробникам більш ефективно та гнучко працювати з даними та функціями у програмах [14].

Об'єкт – це структура даних, яка складається з набору властивостей, що описують характеристики цього об'єкту. Крім того, об'єкт може мати унікальний ідентифікатор `_id`, який служить для ідентифікації конкретного об'єкта в системі.

Колекція – це сукупність об'єктів, які мають однакову структуру та назву, і зберігаються разом в базі даних. При цьому лише один об'єкт належить лише одній колекції.

Властивість – це сукупність назви і відповідного йому типу і значення [15].

Типи властивостей – ціле число, рядок, число з плаваючою точкою, об'єкт, масив, бінарна рядок, символ, байт, дата, `boolean`, `null`.

Підтримуються операції вибірки, вставки, зміни та видалення даних. Об'єкти не мають зв'язків між собою, але можуть зберігати інші об'єкти у

властивостях. Індокси можуть бути накладені на властивості вкладених об'єктів.

В якості унікального ідентифікатора використовується 12-байтне унікальне число, що генерується на клієнті. Це дозволяє уникнути проблем з синхронізацією реплік та переповненням цілого числа, а також забезпечує збереження посилань на статті після перевтілення бази даних.

В процесі проектування даного програмного продукту для побудови об'єктно-орієнтованої моделі було виділено наступні об'єктні множини: «Categories», «SortPop»

Документ «SortPop» (таблиця 4.1, рисунок 4.1) зберігає інформацію про блок сортування.

Таблиця 4.1 – Опис документу «SortPop»

Параметр	Тип значення	Опис
activeSortType	String	Визначає обране значення для сортування
items	Array	Вибирає властивості продукту з серверу
onClickSortType	Function	Визначає яке значення обране і сортує дані

```
SortPopup.propTypes = {
  activeSortType: PropTypes.string.isRequired,
  items: PropTypes.arrayOf(PropTypes.object).isRequired,
  onClickSortType: PropTypes.func.isRequired,
};
```

Рисунок 4.1 – Схема документу «SortPop»

Документ «Categories» (таблиця 4.2, рисунок 4.2) зберігає інформацію про категорії на сайті.

Таблиця 4.2 – Опис документу «Categories»

Параметр	Тип значення	Опис
Items	Array	Вибирає властивості продукту з серверу
onClickCategory	Func	Визначає, яка категорія була обрана і сортує дані

```
Categories.propTypes = {
  items: PropTypes.arrayOf(PropTypes.string).isRequired,
  onClickCategory: PropTypes.func.isRequired,
```

Рисунок 4.2 – Схема документу «Categories»

4.3 Структура Json-файлу

В основі розробки інтернет-магазину, лежить Json-файл, структура якого зображена на рисунку 4.3.



Рисунок 4.3 – Структура зображення Json-файл

Json-файл містить вміст даних розробленого проєкту, а саме:

- категорії піц;
- назви піц;
- опис піц;
- зображення піц.

JSON-файл, що містить товари, зберігається на сервері в каталозі з даними. Коли користувач додає новий товар, редагує існуючий товар або видаляє товар з файлу, дані зберігаються в JSON-файлі. При пошуку товару в файлі, інформація про товари витягується з JSON-файлу та відображається на сторінці магазину. Для виведення списку всіх товарів інформація з JSON-файлу також витягується та відображається у вигляді таблиці з деталями про кожен товар.

На рисунку 4.4 наведена блок-схема збереження JSON-файлу у збереженні товарів в інтернет-магазині.

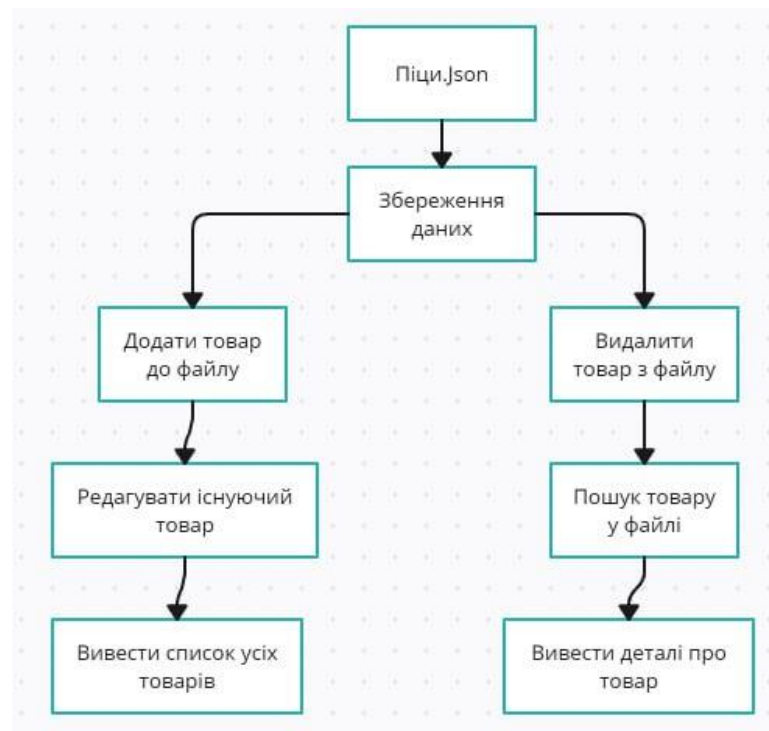


Рисунок 4.4 – Блок-схема роботи JSON-файлу у збереженні товарів в інтернет-магазині

```

[ {
  "id": 3,
  "title": "Сирна",
  "price": "150",
  "sizes": [26, 30, 40]
},

```

```
{
  "id":2,
  "title":"Курча барбекю ",
  "price":"150",
  "sizes":[26,30,40]
}]
```

4.4 Структурна схема компонентів інтернет-магазину

У проєкті інтернет-магазину «Like Pizza» міститься 7 компонентів, їх перелік і функції описані нижче в таблиці 4.3.

Таблиця 4.3 – Опис компонентів проєкту

Назва компоненту	Опис компоненту
LoadingBlock	Генерує вигляд завантаження інформації
Button	Містить посилання на кошик
CartItem	Містить інформацію про товари у кошику
Categories	Містить інформацію про властивості товарів
Header	Містить у собі «дах» сайту, з логотипом
SortPop	Дозволяє сортувати товари
PizaBlock	Генерує та повертає блок на кожен елемент масиву
Cart	Містить кошик
Home	Містить головний блок магазину

React-компонент - це один з базових будівельних блоків у розробці веб-додатків на React. Компонент можна розглядати як самостійний фрагмент інтерфейсу, який може бути повторно використаний на сторінках додатку.

Кожен React-компонент має свій власний стан, властивості (props) та методи, що визначають його поведінку та вигляд. Зазвичай компоненти створюються з метою відображення певної інформації або реалізації певної функціональності на сторінці додатку.

Компоненти можуть бути класовими (class components) або функціональними (functional components). Класові компоненти можуть мати свій внутрішній стан, властивості та методи, а функціональні компоненти зазвичай використовуються для простіших функцій відображення.

React-компоненти можна розглядати як блоки з будівельного набору, які можна зібрати разом, щоб створити складніші інтерфейси та функціональність. React-компоненти дозволяють розбити додаток на менші, більш керовані та зрозуміліші частини, що полегшує розробку та підтримку додатку в майбутньому.

Інтернет-магазин можна розглядати як веб-сайт, який складається з двох основних частин: адміністративної та клієнтської. Адміністративна частина призначена для адміністратора магазину, який має доступ до бази даних і може додавати та редагувати інформацію про товари. Клієнтська частина доступна для всіх відвідувачів магазину, які можуть шукати та переглядати товари, розміщені в різних категоріях. Клієнт може оформити замовлення на товар, обрати спосіб оплати та доставки. Така структура інтернет-магазину значно спрощує процес пошуку та купівлі товарів для користувачів.

«Home» є головним компонентом, який містить в собі всі необхідні блоки та інформацію про піци.

«Header» відповідає за відображення верхньої частини сайту з логотипом та навігаційним меню.

«SortPop» - компонент, який дозволяє сортувати піци за критеріями, такими як ціна, популярність та інші.

«Categories» містить інформацію про властивості піц, такі як розмір, склад, категорії та інші.

«PizzaBlock» - компонент, який генерує та повертає блок для кожного товару, включаючи зображення, ціну, властивості та інші деталі.

«LoadingBlock» - компонент, який генерує вигляд завантаження інформації поки товари завантажуються з сервера.

«Button» містить посилання на кошик, що дозволяє користувачам додавати товари до кошика.

«Cart» - компонент, який містить інформацію про піци в кошику та дозволяє користувачам здійснювати покупки.

«CartItem» - компонент, який містить інформацію про окремий товар в кошику, такі як зображення, ціну, кількість та інші деталі.

Структурна схема взаємозв'язку компонентів веб-сайту зображено на рисунку 4.5.

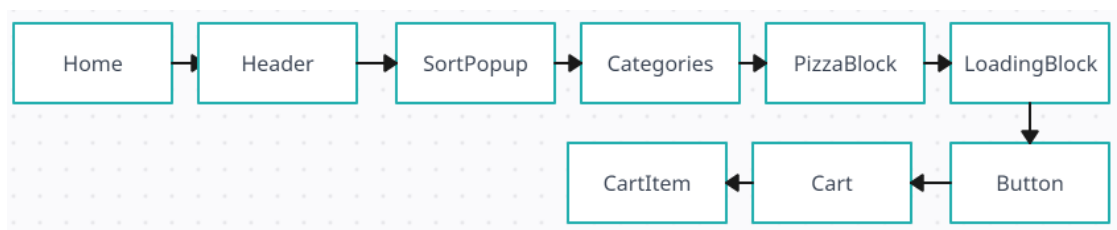


Рисунок 4.5 – Схема компонентів веб-сайту

4.5 Висновки за розділом

У цьому розділі було розглянуто основне призначення й умови використання розроблюваного інтернет-магазину. На рисинку 4.6 зображена структурна схема компонентів інтернет-магазину.

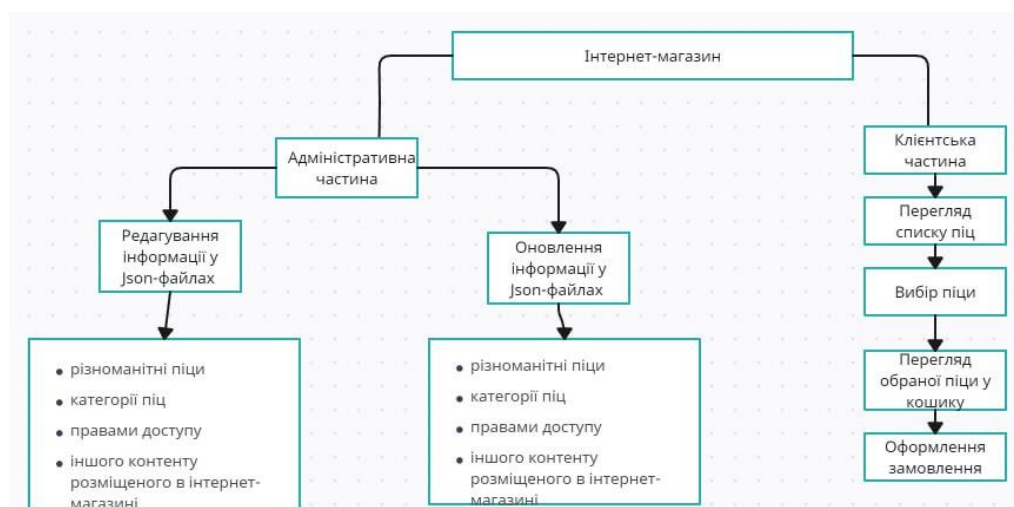


Рисунок 4.6 – Структурна схема компонентів інтернет-магазину

Описані основні логічні блоки що були розроблені у процесі розробки інтернет-магазину, та деякі логічні функції що наявні у логічних блоках.

Було розглянуто структуру Json-файлу. Також було розглянуто структурну схему компонентів інтернет-магазину, яка демонструє, як компоненти магазину взаємодіють між собою, що дозволяє забезпечити ефективну роботу магазину та зручне користування ним для клієнтів.

Створено структуру Json-файлу інтернет-магазину.

Описано структурну схему розробленої системи.

5 КЕРІВНИЦТВО СИСТЕМНОГО АДМІНІСТРАТОРА

5.1 Розгортання інтернет-магазину

Щоб почати роботу з сайтом необхідно скопіювати проект з Git-репозиторію (GrevtsovaAlex/LikePiza (github.com)) на власний пристрій і відкрити необхідні файли у будь-якому текстовому редакторі. Після цього можна змінювати відповідну програмну частину проекту.

Після завершення роботи результат редагування необхідно зберегти. Для того, щоб зміни були застосовані для всіх користувачів, треба замінити програмну частину на сервері на оновлену.

5.2 Вхідні і вихідні дані

Вхідними даними до сайту є файл json який виконує роль бази даних. Вихідними даними сайту є візуальна частина, тобто React-відповідь.

5.3 Створення віртуального хоста на сервері

Код починається з імпорту бібліотеки «json-server» та створення екземпляру серверу за допомогою функції «create()». Далі, встановлюється шлях до файлу з даними в форматі JSON, який буде використовуватися сервером, за допомогою функції «jsonServer.router('./public/db.json')».

Встановлюється порт, на якому буде працювати сервер та встановлюється значення за замовчуванням «3000».

Після цього, застосовуються проміжні програми та маршрутизатор а допомогою функції «use()», після чого сервер запускається за допомогою функції «listen()» з виведенням повідомлення про його запуск у консолі.

Таким чином, цей код дозволяє створити Mock-сервер з використанням «json-server», який буде використовувати файл з даними в форматі JSON для підтримки API, а також підтримувати статичні файли.

Приклад створення хостингу на сервері.

```
const jsonServer = require('json-server');  
const server = jsonServer.create();  
const router = jsonServer.router('./public/db.json');  
const middlewares = jsonServer.defaults({ static: './build', });  
const PORT = process.env.PORT || 3000;  
server.use(middlewares);  
server.use(router);  
server.listen(PORT, () => { console.log('Server is running');  
});
```

Mock-сервер- це сервер, який емулює поведінку реального сервера, але не містить реальних даних. Він може бути використаний для тестування або розробки клієнтських додатків, коли реальний сервер ще не готовий або важко доступний.

У веб-розробці, Mock -сервери використовуються для емуляції взаємодії з реальним API, тестування клієнтських додатків та багато іншого.

5.4 Висновки за розділом

У даному розділі було надано опис призначення та умови експлуатації інтернет-магазину. Розглянуто вхідні та вихідні дані.

Описаний процес створення віртуального хостингу на сервері та процес розгортання електронного магазину.

6 КЕРІВНИЦТВО КОРИСТУВАЧА

6.1 Призначення сайту

Веб-сайт призначений для відображення даних про піци та створення замовлень.

Внесення змін в проект відбуваються в через адміністративну панель, розроблену з допомогою запитів HTTP.

Для запуску в режимі розробки треба відкрити проект в Visual Studio Code. Веб-сайт є односторінковим.

Серверна частина сайту була розроблена за допомогою платформи Node.js, зовнішня оболонка за допомогою фреймворку React, мови гіпертекстової розмітки HTML. У якості бази даних були використані JSON-файли.

Умовою експлуатації сайту є обов'язково наявність надійного з'єднання з мережею Інтернет, а також наявність всіх модулів на вебсервері.

Для роботи з сайтом потрібен будь-який ПК з пристроями введення інформації (клавіатура та комп'ютерна миша) або ноутбук, або якийсь інший пристрій з можливістю відображення веб-сторінок, та виходом до мережі інтернет, з попередньо встановленим веб-браузером: Google Chrome, Opera, Mozilla Firefox або інший.

6.2 Виконання програми

Для початку роботи необхідно відкрити проект за допомогою Visual Studio Code. Запустити консоль Terminal-New terminal (Термінал – Новий термінал).

Завантажити сайт за допомогою команди «npm run start» та сервер за допомогою команди «npm run json-server».

Після завантаження сайту перейти за адресою <http://localhost:3000/>.

На рисунку 6.1 зображено вхід до веб-сайту з браузера.

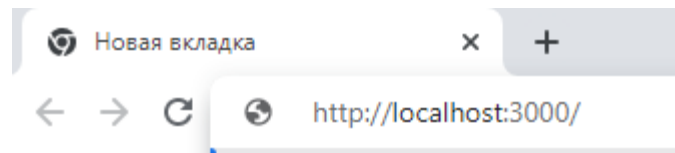


Рисунок 6.1 – Вхід до веб-сайту з браузера

Далі, після того як був здійснений перехід за посиланням до веб-сайту, користувач потрапляє на головну сторінку магазину, де розташовані потрібні різновиди піц, серед яких користувачу потрібно обрати саме ту, яка йому потрібна. Головна сторінка веб-сайту «Like Pizza» продемонстрована на рисунку 6.2.

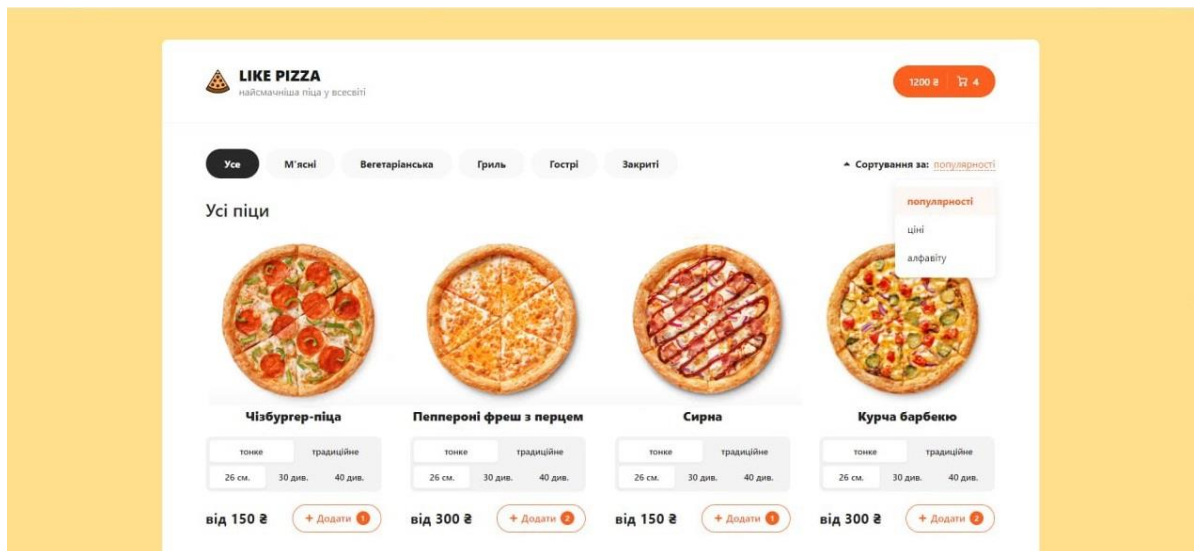


Рисунок 6.2 – Головна сторінка сайту

На головній сторінці інтернет-магазину «Like Pizza» також можна побачити сортування та вибір категорій різноманітних піц. Це продемонстровано на рисунку 6.3

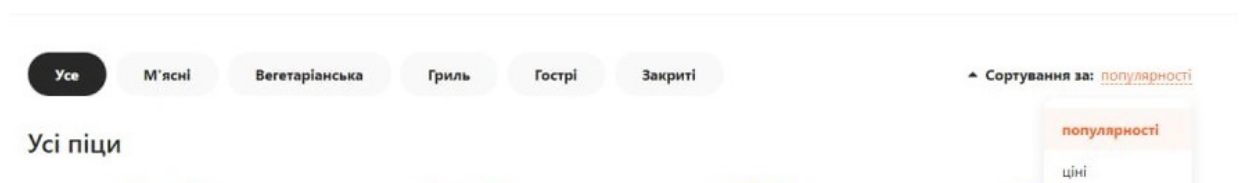


Рисунок 6.3 – Приклад сортування на головній сторінці

Далі обрати товар за допомогою сортування чи вибору товару в певній категорії, обрати кількість піц, обрати тісто та розмір. Після цього обрані користувачем піци будуть відображені у кошику. Кнопка обраних замовлень зображена на рисунку 6.4.



Рисунок 6.4 – Кнопка обраних замовлень

Після обраних товарів є можливість перейти в кошик та подивитися замовлення.

На рисунку 6.5 зображена сторінка кошику «Like Pizza» без замовлень.

Кошик порожній 😞
Найімовірніше, ви не замовляли ще піцу.
Щоб замовити піцу, перейди на головну сторінку.

Повернутися назад



Рисунок 6.5 – Кошик сайту «Like Pizza» без замовлень

На рисунку 6.6 зображена сторінка кошику «Like Pizza» із замовленням.

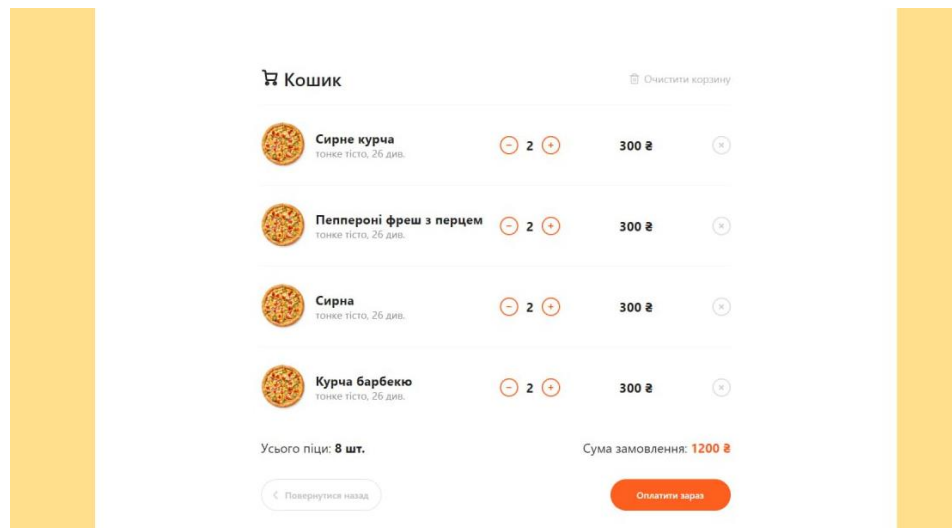


Рисунок 6.6 – Сторінка кошику «Like Pizza» із замовленням

На рисунку 6.7 зображена сторінка оформлення замовлення.

Рисунок 6.7 – Сторінка оформлення замовлення

6.3 Висновки за розділом

В розділі було описано процес взаємодії користувача з інтернет-магазином, способи сортування, пошуку, замовлення та оплати товару.

Інтерфейс розробленого веб-сайту є типовим для будь-якого користувача і не повинен викликати тих чи інших складнощів при використанні за основним призначенням.

ВИСНОВКИ

У процесі виконання дипломної кваліфікаційної роботи було розроблено інтернет-магазин «Like Pizza» із продажу піц.

Після тестування та налагодження функціонування веб-додатку можна зробити наступні висновки:

- функціонал інтернет-магазину повністю відповідає вимогам технічного завдання;
- інтернет-магазин з продажу піц має зручний інтерфейс та діалог з користувачем;
- інтернет-магазин забезпечує коректність введення та виведення даних;
- інтернет-магазин має необхідну інформацію про процеси роботи системи (допомога користувачеві) та коректно відображається у всіх сучасних браузерях.

Для розробки проєкту було обрано фреймворк ReactJS для розробки клієнтської частини сайту, для розробки серверної частини було обране середовище Node.js.

Як базу даних було обрано JSON-файли. Описано блок-схему роботи JSON-файлів.

Також проаналізоване середовище розробки, що відповідає зазначеним у постановці задачі критеріям, та необхідний набір інструментів для вирішення проєктної задачі.

Обґрунтовано вибір мови програмування та середовища функціонування системи.

В результаті випробувань розроблений інтернет-магазин дозволяє відображати інформацію про різноманітні піци на сайті та продавати ці продукти через Інтернет.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Пономаренко Л. А. Електронна комерція [Текст] : підруч. для студ. екон. спец. / Л. А. Пономаренко, В. О. Філатов ; Київський національний торговельно-економічний ун-т. - К. : КНТЕУ, 2002. - 443 с.
2. Краус К. М. Електронна комерція та інтернет-торгівля [Текст] : навч.-метод. посіб. / К. М. Краус, Н. М. Краус, О. В. Манжура. - Київ : Аграр Медіа Груп, 2021. - 454 с.
3. Node.js: що це таке [Електрон. ресурс]. – Режим доступу: <https://nodejs.org/uk/about>
4. Node.js: причини чому варто вивчити цю платформу [Електрон. ресурс]. – Режим доступу: <https://chiacademy.software/blog/node-js-3-golovni-prychyny-chomu-varto-vyvchyty>
5. Single Page Application: що таке SPA додатки [Електрон. ресурс]. – Режим доступу: <https://wezom.com.ua/ua/blog/что-такое-spa-prilozheniya>
6. Monocubed / [Electronic resource]. – Access mode: <https://www.monocubed.com/blog/top-single-page-application-frameworks/>
7. NPM / [Electronic resource]. – Access mode: <https://www.npmjs.com/package/react-router-dom>
8. React: знайомство з React [Електрон. ресурс]. – Режим доступу: <https://uk.legacy.reactjs.org/tutorial/tutorial.html>
9. DOM: що таке DOM і для чого він потрібен [Електрон. ресурс]. – Режим доступу: <http://xn--80adth0aefm3i.xn--j1amh/DOM>
10. HTTP: що це таке [Електрон. ресурс]. – Режим доступу: <https://hostiq.ua/wiki/ukr/http-https/>
11. IBEF / [Electronic resource]. – Access mode: <https://www.ibef.org/>
12. Браузери: що це таке і яке він має призначення [Електрон. ресурс]. – Режим доступу: <https://futurenow.com.ua/shho-take-brauzer-i-yake-vin-maye-pryznachennya/>

13. Інформаційні потоки: сутність і значення [Електрон. ресурс]. – Режим доступу: <https://referatss.com.ua/work/informacijni-potoki-sutnist-i-znachennja/>
14. Об'єктно-орієнтована модель: що це таке [Електрон. ресурс]. – Режим доступу: https://stud.com.ua/53305/informatika/obyektno-oriyentovana_model
15. Властивість: що це таке [Електрон. ресурс]. – Режим доступу: <http://www.kievoit.ippo.kubg.edu.ua/kievoit/2013/3/3.html>
16. React: розробка інтернет-магазину for junior [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=bziVFvq8cLQ&list=PL0FGkDGJQjJFMRmP7wZ771m1Nx-m2_qXq
17. React: розробка інтернет-магазину for junior [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=xvuDzujtxN4&list=PL0FGkDGJQjJFMRmP7wZ771m1Nx-m2_qXq&index=2
18. React: розробка інтернет-магазину for junior [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=SZma4oQN8SY&list=PL0FGkDGJQjJFMRmP7wZ771m1Nx-m2_qXq&index=3
19. React: розробка інтернет-магазину for junior [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=IMBAK-DftVM&list=PL0FGkDGJQjJFMRmP7wZ771m1Nx-m2_qXq&index=4
20. React: розробка інтернет-магазину for junior [Електрон. ресурс]. – Режим доступу: https://www.youtube.com/watch?v=5XKs7dvVQD8&list=PL0FGkDGJQjJFMRmP7wZ771m1Nx-m2_qXq&index=5

ДОДАТОК А
Технічне завдання

A.1 Підстава для розробки

Підставою для розробки послужило завдання на дипломну роботу на тему «Програмна реалізація інтернет-магазину «Like Pizza», затверджений 4 квітня 2023 року № 87 по Національному університету «Запорізька політехніка» .

A.2 Призначення розробки

Мета роботи – розробка програмного продукту, який реалізує функції інтернет-магазину з індивідуальним наповненням та працює в мережі Інтернет, та створення індивідуального наповнення веб-додатку відповідно до певної предметної області.

A.3 Основні вимоги до розроблювальної програми

A.3.1 Вимоги до надійності

Програма повинна забезпечувати коректну обробку помилок у наступних ситуаціях:

- при введенні користувачем неприпустимих значень;
- при невірній роботі з файлами (спробі доступу до файлу неприпустимого формату, спробі запису на захищений від запису диск та ін.).

A.3.2 Умови експлуатації

Під час роботи програми не рекомендується перезавантажувати ПК або вимикати живлення. Вихід із програми необхідно здійснювати послідовною зупинкою клієнтського й серверного сервісів.

Експлуатація програмного продукту здійснюється відповідно до експлуатаційної документації на розроблену програму, що відповідає

стандартам і містить інформацію, необхідну для освоєння й експлуатації програмного продукту.

A.3.3 Вимоги до складу й параметрів технічних засобів

Необхідними вимогами до складу й параметрів технічних засобів, які повинні бути застосовані для роботи інтернет-магазину, є:

- Оперативна пам'ять - не менш 1Гб;
- HDD (жорсткий диск)- не менш 40Гб;
- відео адаптер: DirectX 9 або пізніша версія з драйвером WDDM 1.0;
- процесор - не менш 1 ГГц;
- мережний адаптер.

A.3.4 Вимоги до маркування й пакування

Програма може бути записана як на Flash-накопичувач, так і на компакт-диску. На пакуванні повинна бути назва програми - «Програмна реалізація інтернет-магазину «Like Pizza».

A.3.5 Вимоги до транспортування й зберігання

Вимоги до транспортування й зберігання аналогічні вимогам, які пропонуються до носія, на якому записана програма.

A.3.6 Вимоги до програмної документації

Програма повинна поставлятися з "Керівництвом оператора".

ДОДАТОК Б
Текст програми

При розробці коду програми було використано підходи та рекомендації, запропоновані в [16] - [20].

SERVER PART

Server.js

```
const jsonServer = require('json-server');
const server = jsonServer.create();
const router = jsonServer.router('./public/db.json');
const middlewares = jsonServer.defaults({ static: './build', });
const PORT = process.env.PORT || 3000; server.use(middlewares);
server.use(router);
server.listen(PORT, () => { console.log('Server is running'); });
```

CLIENT PART

PizaBlock.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import classNames from 'classnames';
import Button from '../Button';
function PizaBlock({ id, name, imageUrl, price, types, sizes, onClickAddPizza,
addedCount }) {
  const availableTypes = ['тонке', 'традиційне'];
  const availableSizes = [26, 30, 40];
  const [activeType, setActiveType] = React.useState(types[0]);
  const [activeSize, setActiveSize] = React.useState(0)
  const onSelectType = (index) => { setActiveType(index); };
  const onSelectSize = (index) => { setActiveSize(index); };
  const onAddPizza = () => { const obj = {
    id,
    name,
    imageUrl,
    price,
    size: availableSizes[activeSize],
    type: availableTypes[activeType], };
    onClickAddPizza(obj); };
  return (
    <div className="pizza-block">
      <img className="pizza-block__image" src={imageUrl} alt="Pizza" />
      <h4 className="pizza-block__title">{name}</h4>
      <div className="pizza-block_selector">
        <ul>
          {availableTypes.map((type, index) => (
            <li
              key={type} onClick={() => onSelectType (index)}
            </li>
          )
        </ul>
      </div>
    </div>
```

```

className={classNames({
  active: activeType === index,
  disabled: !types.includes(index), })}>
  {type}
</li> )}}
</ul>
<ul>
  {availableSizes.map((size, index) => (
    <li
      key={size}
      onClick={()=>onSelectSize(index)}
      className={classNames({
        active: activeSize === index,
        disabled: !sizes.includes(size), })}>
        {size} см.
      </li>))}
  </ul>
</div>
<div className="pizza-block__bottom">
  <div className="pizza-block__price">{price}
  &lt;/div> // [https://github.com/GrevtsovaAlex/LikePiza]
  <Button onClick={onAddPizza}
    className="button--add" outline>
    <svg
      Width = "12" height="12" viewBox="0 0 12 12" fill="none"
      xmlns="http://www.w3.org/2000/svg">
        <path
          d="M10.8 4.8H7.2V1.2C7.2 0.5373 6.6627 0 6 0C5.3373 0 4.8 0.5373 4.8
          1.2V4.8H1.2C0.5373 4.8 0 5.3373 0 6C0 6.6627 0.5373 7.2 1.2 7.2H4.8V10.8C4.8
          11.4627 5.3373 12 6 12C6.6627 12 7.2 11.4627 7.2 10.8V7.2H10.8C11.4627 7.2
          12 6.6627 12 6C12 5.3373 11.4627 4.8 10.8 4.8Z"
          fill="white"/>
        </svg>
        <span>Додати</span>
        {addedCount}&& <i>{addedCount}</i>}
      </Button>
    </div>
  </div> ); }
PizzaBlock.propTypes = {
  name: PropTypes.string,
  imageUrl: PropTypes.string,
  price: PropTypes.number,

```

```

types: PropTypes.arrayOf(PropTypes.number),
sizes: PropTypes.arrayOf(PropTypes.number),
onClickAddPizza: PropTypes.func,
addedCount: PropTypes.number,};

```

```

PizzaBlock.defaultProps = {
  name: '---',
  price: 0,
  types: [],
  sizes: [], };
export default PizzaBlock;

```

LoadingBlock.jsx

```

import React from 'react';
import ContentLoader
from 'reactcontent-loader';
function LoadingBlock() {
  return (
    <ContentLoader
      className="pizza-block"
      speed={2}
      width={280}
      height={460}
      viewBox="0 0 280 460"
      backgroundColor="#f3f3f3"
      foregroundColor="#ecebeb">
      <circle cx= "132" cy = "142" r = "115"/>
      <rect x="0" y="273" rx="6" ry="6" width="280" height="26"/>
      <rect x="0" y="310" rx="6" ry="6" width="280" " height="84" />
      <rect x="0" y="418" rx="6" ry="6" width="91" height="31" />
      <rect x="137" y="408" rx="25" ry="25" width="140" " height="46"/>
    </ContentLoader>
  ); }/[https://github.com/GrevtsovaAlex/LikePiza]
export default LoadingBlock;

```

Button.jsx

```

import React from 'react';

import PropTypes from 'prop-types';
import classNames from 'classnames';

const Button = ({ onClick, className, outline, children }) => {
  return (
    <button onClick={onClick}
      className={classNames('button',
        className, {

```

```
'button--outline': outline,}}}>
{children}
</button>); };

```

```
Button.propTypes = {
onClick: PropTypes.func, };
export default Button;

```

CartIt.jsx

```
import React from 'react';
import Button from './Button';
const CartIt = ({ id, name, type, size, totalPrice, totalCount, onRemove, onMinus,
onPlus }) => {
const handleRemoveClick = () => { onRemove(id); };
const handlePlusItem = () => { onPlus(id); };
const handleMinusItem = () => { onMinus(id); };
return (
<div className="cart__item">
<div className="cart__item-img">

</div>
<div className="cart__item-info">
<h3>{name}</h3>
<p>{type} тiкто, {size} см.</p>
</div>
<div className="cart__item-count">
<div
onClick={handleMinusItem
className="button button—outline
button--circle cart__item-count-minus">
<svg
width="10"
height="10"
viewBox="0 0 10 10"

```

```

fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001
9.6C4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001
3.84V0.96C4.00001 0.42984 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984
5.92001 0.96V3.84Z"
fill="#EB5A1E"
/>
<path
d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.92001 -
2.29533e-05 5.49017 -2.29301e-05 4.96001C2.2907e-05 4.42985 0.429817
4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998
4.00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017
9.17014 5.92001 8.63998 5.92001L5.75998 5.92001Z"
fill="#EB5A1E"/>
</svg>
</div>
<b>{totalCount}</b>
<div
onClick={handlePlusItem}
className="button button—outline
button--circle cart__item-count-plus">
<svg
width="10"
height="10"
height="10"
height="10"
xmlns="http://www.w3.org/2000/svg">
<path
d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001
9.6C4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001
3.84V0.96C4.00001 0.42984 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984
5.92001 0.96V3.84Z"
fill="#EB5A1E"

```

```

/>
<path
d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.92001 -
2.29533e-05 5.49017 -2.29301e-05 4.96001C2.2907e-05 4.42985 0.429817
4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998
4.00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017
9.17014 5.92001 8.63998 5.92001L5.75998 5.92001Z"
fill="#EB5A1E"
/>
</svg>
</div>
</div>
<div className="cart__item-price">
<b>{totalPrice} €</b>
</div>
<div className="cart__item-remove">
<Button
onClick={handleRemoveClick}
className="button--circle" outline>
<svg
width="10"
height="10"
viewBox="0 0 10 10"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M5.92001 3.84V5.76V8.64C5.92001 9.17016 5.49017 9.6 4.96001
9.6C4.42985 9.6 4.00001 9.17016 4.00001 8.64L4 5.76L4.00001
3.84V0.96C4.00001 0.42984 4.42985 0 4.96001 0C5.49017 0 5.92001 0.42984
5.92001 0.96V3.84Z"
fill="#EB5A1E"//[https://github.com/GrevtsovaAlex/LikePiza]
/>
<path
d="M5.75998 5.92001L3.83998 5.92001L0.959977 5.92001C0.429817 5.92001 -
2.29533e-05 5.49017 -2.29301e-05 4.96001C2.2907e-05 4.42985 0.429817
4.00001 0.959977 4.00001L3.83998 4L5.75998 4.00001L8.63998
4.00001C9.17014 4.00001 9.59998 4.42985 9.59998 4.96001C9.59998 5.49017
9.17014 5.92001 8.63998 5.92001L5.75998 5.92001Z"
fill="#EB5A1E"
/>
</svg>
</Button>
</div>
</div>
); };

```

```
export default CartIt;
```

Categories.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
const Categories = React.memo(function Categories({ activeCategory, items,
onClickCategory })
{ return
<div className = "categories">
<ul>
<li
className={activeCategory === null ? 'active' : ''}
onClick={() =>
onClickCategory(null)}> Bce </li>
{items && items.map((name, index) => (
<li
className={activeCategory === index ? 'active' : ''}
onClick={() => onClickCategory(index)}
key={`_${name}_-${index}`}> {name}
))}
);
});
Categories.propTypes = { items:
PropTypes.arrayOf(PropTypes.string).isRequired, onClickCategory:
PropTypes.func.isRequired, };
Categories.defaultProps={ activeCategory: null, items: [] };
export default Categories;
```

Header.jsx

```
import React from 'react';
import { Link } from 'react-router-dom';
import { useSelector } from 'react-redux';
import logoSvg from '../assets/img/pizzalogo.svg';
```

```

import Button from './Button';
function Header() {
  const { totalPrice, totalCount } = useSelector(({ cart }) => cart);
  return (
    <div className="header">
      <div className="container">
        <Link to = "/">
          <div className="header__logo">
            <img width="38" src={logoSvg} alt="Pizza logo" />
          </div>
          <h1>Like Pizza</h1>
          <p>найсмачніша піца у всесвіті</p>
        </div>
      </div>
      </Link>
      <div className="header__cart">
        <Link to = "/cart">
          <Button className="button--cart">
            <span>{totalPrice} ₪</span>
            <div className="button__delimiter">
              </div>
              <svg
                width="18"
                height="18"
                viewBox="0 0 18 18"
                fill="none"
                xmlns="http://www.w3.org/2000/svg">
                <path
                  d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C7.66667 14.2636 7.06971 13.6667 6.33333 13.6667C5.59695 13.6667 5 14.2636 5 15C5 15.7364 5.59695 16.3333 6.33333 16.3333Z"
                  stroke="white"
                  strokeWidth="1.8"
                  strokeLinecap="round"

```

```

strokeLinejoin="round"/>
<path
d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C15.6667
14.2636 15.0697 13.6667 14.3333 13.6667C13.597 13.6667 13 14.2636 13 15C13
15.7364 13.597 16.3333 14.3333 16.3333Z"
stroke="white"
strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round"/>
<path
d="M4.78002 4.99999H16.3334L15.2134 10.5933C15.1524 10.9003 14.9854
11.176 14.7417 11.3722C14.4979 11.5684 14.1929 11.6727 13.88
11.6667H6.83335C6.50781 11.6694 6.1925 11.553 5.94689 11.3393C5.70128
11.1256 5.54233 10.8295 5.50002 10.5067L4.48669 2.82666C4.44466 2.50615
4.28764 2.21182 4.04482 1.99844C3.80201 1.78505 3.48994 1.66715 3.16669
1.66666H1.66669"
stroke="white"
strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round"/>
</svg>
<span>{totalCount}</span>
</Button>
</Link>
</div>
</div>
</div>); }
export default Header;
SortPop.jsx
import React from 'react';

```

```

import PropTypes from 'prop-types';
const SortPop = React.memo(function SortPop({ items, activeSortType,
onClickSortType }) {
const [visiblePopup, setVisiblePopup] = React.useState(false);
const sortRef = React.useRef();
const activeLabel = items.find((obj) =>
obj.type === activeSortType).name;
const toggleVisiblePopup = () => {
setVisiblePopup(!visiblePopup); };
const handleOutsideClick = (event) => {
const path = event.path || (event.composedPath&&event.composedPath());
if (!path.includes(sortRef.current)) {
setVisiblePopup(false); } };
const onSelectItem = (index) => {
if (onClickSortType) {
onClickSortType(index);}
setVisiblePopup(false);};
React.useEffect(() => {
document.body.addEventListener('click', handleOutsideClick);
}, []);/[https://github.com/GrevtsovaAlex/LikePiza]
return (
<div ref={sortRef} className="sort">
<div className="sort__label">
<svg
className={visiblePopup ? 'rotated' : ""}
width="10"
height="6"
viewBox="0 0 10 6"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M10 5C10 5.16927 9.93815 5.31576 9.81445 5.43945C9.69075 5.56315
9.54427 5.625 9.375 5.625H0.625C0.455729 5.625 0.309245 5.56315 0.185547
5.43945C0.061849 5.31576 0 5.16927 0 5C0 4.83073 0.061849 4.68424 0.185547
4.56055L4.56055 0.185547C4.68424 0.061849 4.83073 0 5 0C5.16927 0 5.31576
0.061849 5.43945 0.185547L9.81445 4.56055C9.93815 4.68424 10 4.83073 10
5Z"
fill="#2C2C2C"/>
</svg>
<b>Сортування за: </b>
<span
onClick={toggleVisiblePopup}>{activeLabel}
</span>
</div>
{visiblePopup && (

```

```

<div className="sort__popup">
  <ul>
    {items && items.map((obj, index) => (
      <li
        onClick={()=>onSelectedItem(obj)}
        className={activeSortType=== obj.type ? 'active' : ''}
        key={`_${obj.type}_${index}`}>
        {obj.name}
      </li>
    ))}
  </ul>
</div>
)}
</div>
); });
SortPop.propTypes = {
  activeSortType:
    PropTypes.string.isRequired,
  items:
    PropTypes.arrayOf(PropTypes.object).isRequired,
  onClickSortType: PropTypes.func.isRequired,
};
SortPop.defaultProps = {
  items: [],};
export default SortPop;

```

Cart.jsx

```

import React from 'react'; import { useSelector, useDispatch }
from 'react-redux';
import { Link } from 'react-router-dom';
import cartEmptyImage from '../assets/img/empty-cart.png';
import { CartIt, Button } from '../components';
import { clearCart, removeCartIt, plusCartIt, minusCartIt } from
'../redux/actions/cart';
function Cart() {
  const dispatch = useDispatch();
  const { totalPrice, totalCount, items } = useSelector(({ cart }) => cart);
  const addedPizzas=Object.keys(items).map((key) => {
    return items[key].items[0];
  });
  const onClearCart = () => {
    if(window.confirm('Ви дійсно хочете очистити кошик?')) {
      dispatch(clearCart());} };
  const onRemoveItem = (id) => {
    if(window.confirm('Ви дійсно хочете видалити?')) {
      dispatch(removeCartIt(id)); } };

```

```

const onPlusItem = (id) => {
  dispatch(plusCartIt(id)); };
const onMinusItem = (id) => {
  dispatch(minusCartIt(id)); };
const onClickOrder = () => {
  console.log('БАШЕ ЗАМОБЛЕХНЯ', items);
};
return (
  <div className="container container--cart">
    {totalCount ? (
      <div className="cart">
        <div className="cart__top">
          <h2 className="content__title">
            <svg
              width="18"
              height="18"
              viewBox="0 0 18 18"
              fill="none"
              xmlns="http://www.w3.org/2000/svg">
              <path
                d="M6.33333 16.3333C7.06971 16.3333 7.66667 15.7364 7.66667 15C7.66667 14.2636 7.06971 13.6667 6.33333 13.6667C5.59695 13.6667 5 14.2636 5 15C5 15.7364 5.59695 16.3333 6.33333 16.3333Z"
                stroke="white"
                strokeWidth="1.8"
                strokeLinecap="round"
                strokeLinejoin="round"
              />
              <path
                d="M14.3333 16.3333C15.0697 16.3333 15.6667 15.7364 15.6667 15C15.6667 14.2636 15.0697 13.6667 14.3333 13.6667C13.597 13.6667 13 14.2636 13 15C13 15.7364 13.597 16.3333 14.3333 16.3333Z"
                stroke="white"
                strokeWidth="1.8"
                strokeLinecap="round"
                strokeLinejoin="round"
              />[/[https://github.com/GrevtsovaAlex/LikePiza]
              <path
                d="M4.78002 4.99999H16.3334L15.2134 10.5933C15.1524 10.9003 14.9854 11.176 14.7417 11.3722C14.4979 11.5684 14.1929 11.6727 13.88 11.6667H6.83335C6.50781 11.6694 6.1925 11.553 5.94689 11.3393C5.70128 11.1256 5.54233 10.8295 5.50002 10.5067L4.48669 2.82666C4.44466 2.50615 4.28764 2.21182 4.04482 1.99844C3.80201 1.78505 3.48994 1.66715 3.16669 1.66666H1.66669"
                stroke="white"

```

```

strokeWidth="1.8"
strokeLinecap="round"
strokeLinejoin="round"
/>
</svg>Кошик
</h2>
<div className="cart__clear">
<svg
width="20"
height="20"
viewBox="0 0 20 20"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M2.5 5H4.16667H17.5"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"
/>
<path
d="M6.66663 5.00001V3.33334C6.66663 2.89131 6.84222 2.46739 7.15478
2.15483C7.46734 1.84227 7.89127 1.66667 8.33329 1.66667H11.6666C12.1087
1.66667 12.5326 1.84227 12.8451 2.15483C13.1577 2.46739 13.3333 2.89131
13.3333 3.33334V5.00001M15.8333 5.00001V16.6667C15.8333 17.1087 15.6577
17.5326 15.3451 17.8452C15.0326 18.1577 14.6087 18.3333 14.1666
18.3333H5.83329C5.39127 18.3333 4.96734 18.1577 4.65478 17.8452C4.34222
17.5326 4.16663 17.1087 4.16663 16.6667V5.00001H15.8333Z"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"
/>
<path
d="M8.33337 9.16667V14.1667"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"
strokeLinejoin="round"
/>
<path
d="M11.6666 9.16667V14.1667"
stroke="#B6B6B6"
strokeWidth="1.2"
strokeLinecap="round"

```

```

strokeLinejoin="round"
/>
</svg>
<span
onClick={onClearCart}>Очистити кошик</span>
</div>
</div>
</div>
className="content__items">
{addedPizzas.map((obj) => (
<CartItem
key={obj.id}
id={obj.id}
name={obj.name}
type={obj.type}
size={obj.size}
totalPrice={items[obj.id].totalPrice}
totalCount={items[obj.id].items.length}
onRemove={onRemoveItem}
onMinus={onMinusItem}
onPlus={onPlusItem}/> ))}
</div>
<div className="cart__bottom">
<div className="cart__bottom-details">
<span>
Всього піц:<b>{totalCount}</b>шт.</b>
</span>
<span>
Сума замовлення:<b>{totalPrice}</b>€
</span>
</div>
<div className="cart__bottom-buttons">
<a href="/" className="button button--outline button--add go-back-btn">
<svg
width="8"
height="14"
viewBox="0 0 8 14"
fill="none"
xmlns="http://www.w3.org/2000/svg">
<path
d="M7 13L1 6.93015L6.86175 1"
stroke="#D3D3D3"
strokeWidth="1.5"
strokeLinecap="round"
strokeLinejoin="round"/>

```

```

</svg>
<Link to="/">
<span>Повернутися назад</span>
</Link></a>
<Button onClick={onClickOrder} className="pay-btn">
<span>Оплатити зараз</span>
</Button>
<div class="by_adres" data-toggle="adres">
<form id="deliv" autocomplete="off" action=method="POST" class="deliv_to"
v-on:submit.prevent="submit($event)">
<span class="title">ОФОРМЛЕННЯ ЗАКАЗУ</span>
<div class="linec">
<label>Ім'я</label>
<input type="text" name="name" class="valid">
<label>ТЕЛЕФОН</label>
<input type="tel" name="phone" class="valid">
</div>
<div>
<label>АДРЕСА</label>
<input type="text" name="stre" class="valid">
<label>БУДИНОК</label>
<input type="text" name="hom" class="valid">
<label>КВ/ОФ</label>
<input type="text" name="rom" class="valid">
<label>ПІД'ЇЗД</label>
<input type="text" name="pid" class="valid">
<label>ДОМОФОН</label>
<input type="radio" name="rad" checked value="rad" class="cheeked" data-
time="active">
<label>КІД НА ДВЕРІ</label>
<input type="radio" name="rd" class="cheeked" value=" rd " data-type="yes"
data-time="active">
</div><div>
<label>ДОСТАВИТИ<br>ЯК МОЖНА ШВИДШЕ</label>
<input type="radio" name="deliv_rad" value="now" class="cheeked" checked
data-time="active">
<label> ДОСТАВИТИ <br>до визначеного часу</label>
<input type="radio" name="deliv_rad" value="now" class="cheeked" checked
data-time="active">
</div><div>
<label>З ЯКОЇ СУМИ ПІДГОТУВАТИ ЗДАЧУ</label>
<span class="cheeked_in"></span>
<div class="flex">
<input type="text" name="change" placeholder="300"
v-model="inputs.change">

```

```

<span>Вкажіть суму</span>
</div>
<div class="text">
<label>КОМЕНТАРІЙ ДО ЗАКАЗУ</label>
<textarea name="comment"
v-model="inputs.comment"
</div>
<div>
<p>Якщо ви хочете, щоб кухар не клав певний інгредієнт, напишіть у
коментарі.Є можливість покласти певні інгредієнти лише у половину
піци.</p>
<button type="submit" class="red">Оформити замовлення</button>
</div></div>
</div>
<div className="cart cart--empty">
<h2>
Корзина пуста <i>☹️</i>
</h2>
<p>
Найммовірніше всього, ви не замовляли ще піцу.
<br/>
Найммовірніше всього, ви не замовляли ще піцу.
</p>
<Link to "/" className="button button--black">
<span>Повернутися назад</span>
</Link>
</div>)}
</div>); }
export default Cart;

```

Home.jsx

```

import React from 'react';

import { useSelector, useDispatch } from 'react-redux';
import { Categories, SortPop, PizaBlock, PizzaLoadingBlock }
from '../components'; import { setCategory, setSortBy }
from '../redux/actions/filters'; import { fetchPizzas }
from '../redux/actions/pizzas';

import { addPizzaToCart } from '../redux/actions/cart';

const categoryNames = ['М'ясні', 'Вегетаріанські', 'Гострі'];
const sortItems = [
{ name: 'популярністю', type: 'popular', order: 'desc' },

```

```

{ name: 'ціні', type: 'price', order: 'desc' },
{ name: 'алфавіт', type: 'name', order: 'asc' }, ],
function Home() { const dispatch = useDispatch();
const items = useSelector(({ pizzas }) => pizzas.items);
const cartIts = useSelector(({ cart }) => cart.items);
const isLoading = useSelector(({ pizzas }) => pizzas.isLoading);
const { category, sortBy } = useSelector(({ filters }) => filters);
React.useEffect(() => { dispatch(fetchPizzas(sortBy, category)); },
[category, sortBy]); const onSelectCategory = React.useCallback((index) =>
{dispatch(setCategory(index)); }, []); const onSelectSortType =
React.useCallback((type) => { dispatch(setSortBy(type)); },[]);
const handleAddPizzaToCart = (obj) =>
{ dispatch({ type: 'ADD_PIZZA_CART', payload: obj, }); };
return (
<div className="container">
<div className= "content__top">
<Categories activeCategory={category}
onClickCategory={onSelectCategory} items={categoryNames} />
<SortPop activeSortType={sortBy.type}
items={sortItems}
onClickSortType={onSelectSortType} />
</div>
<h2 className = "content_title" Все пиццы </h2>
<div className= " content_items">
{isLoading ? items.map((obj) => ( <PizzaBlock
onClickAddPizza={handleAddPizzaToCart} key={obj.id}
addedCount={cartIts[obj.id] && cartIts[obj.id].items.length} {...obj} /> ))
: Array(12) .fill(0) .map((_,index)=> )} ); }
export default Home;
App.js
import React from 'react';

```

```

import { Header } from './components';
import { Home, Cart } from './pages';
import { Route } from 'react-router-dom';
function App() {
return (
<div className="wrapper">
<Header/>
<div className="content">
<Route path="/"
component={Home} exact />
<Route path="/cart"
component={Cart} exact />
</div>
</div>); }
export default App;

```

index.js

```

import React from 'react';
import ReactDOM from 'react-dom';
import { BrowserRouter as Router } from 'react-router-dom';
import { Provider } from 'react-redux';
import store from './redux/store';
import './scss/app.scss';
import App from './App';
ReactDOM.render(
<Router>
<Provider store = {store}>
<App/>
</Provider>
</Router>
document.getElementById('root'), );

```

ДОДАТОК В
Слайди презентації

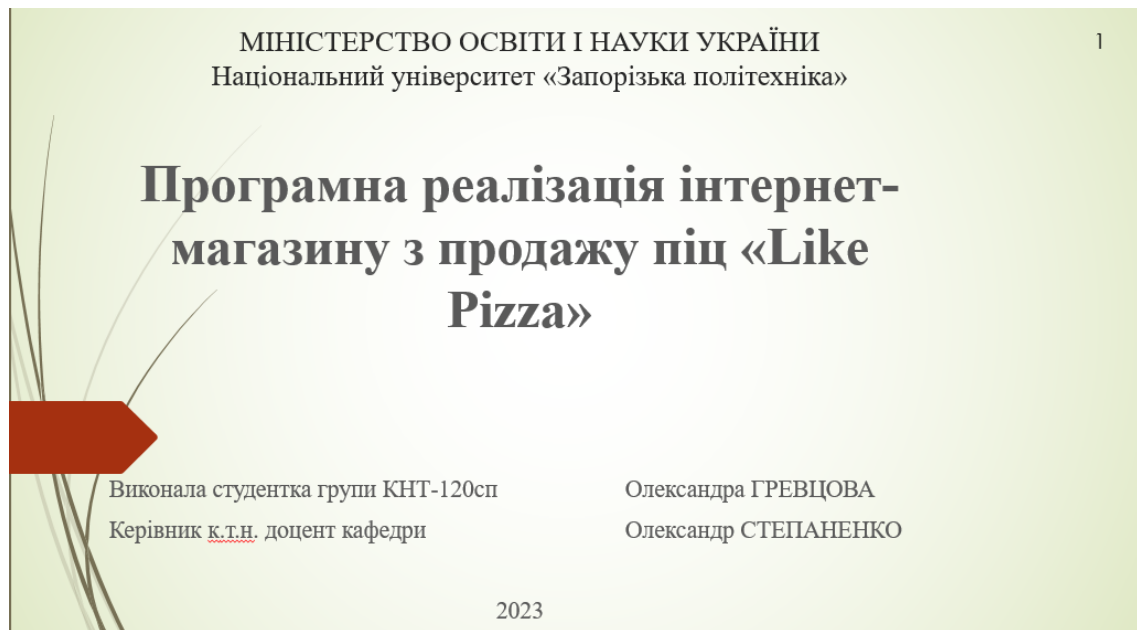


Рисунок В.1 – Слайд №1

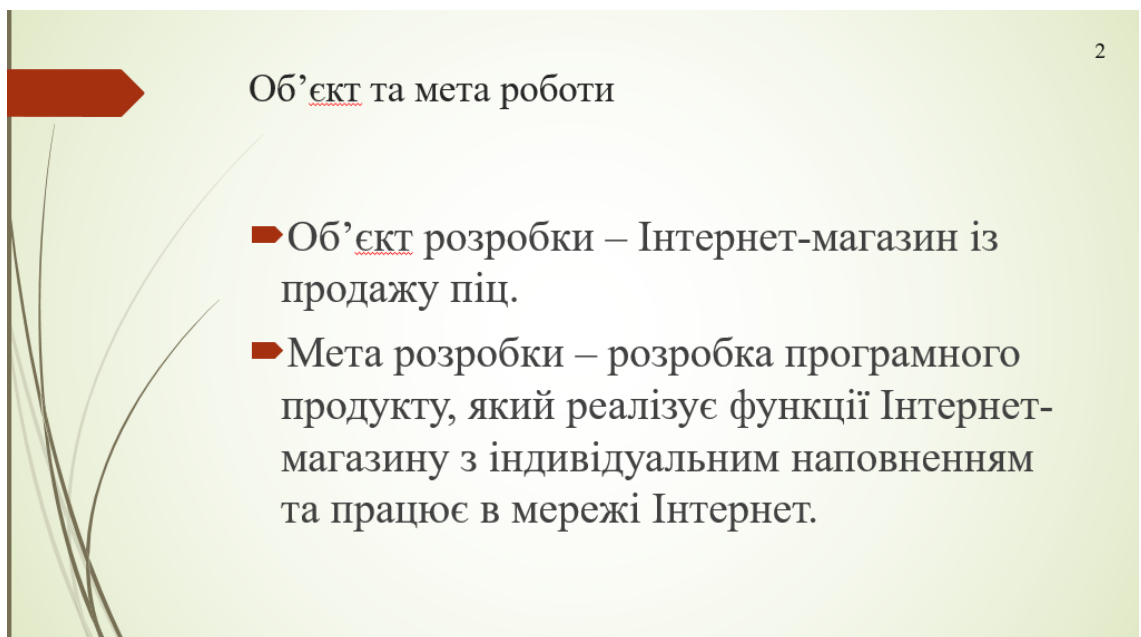


Рисунок В.2 – Слайд №2



Задачі

- Проаналізувати об'єкт розробки;
- Проаналізувати середовище розробки;
- Розробити структуру об'єкту дослідження;
- Розробити структуру Json-файлу;
- Розробити основні програмні модулі;
- Протестувати та відлагодити розроблену програму.

Рисунок В.3 – Слайд №3



Матеріали, методи та технічні засоби

- Мова програмування React;
- Програмна платформа Node.js та Json-файли які виконують функцію бази даних;
- Мова гіпертекстової розмітки HTML;
- Таблиця каскадних стилів CSS;
- Персональний комп'ютер з процесором Intel Pentium Gold під управлінням операційної системи Microsoft Windows 10;

Рисунок В.4 – Слайд №4

Структурна схема компонентів інтернет-магазину

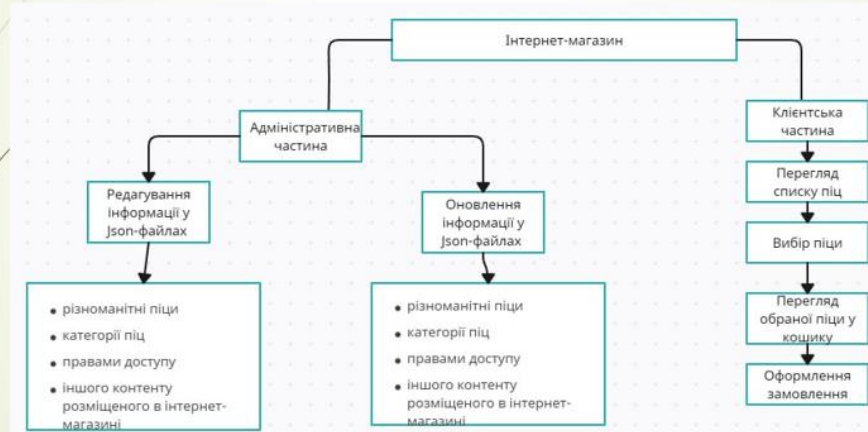


Рисунок В.5 – Слайд №5

Функціональна схема роботи інтернет-магазину

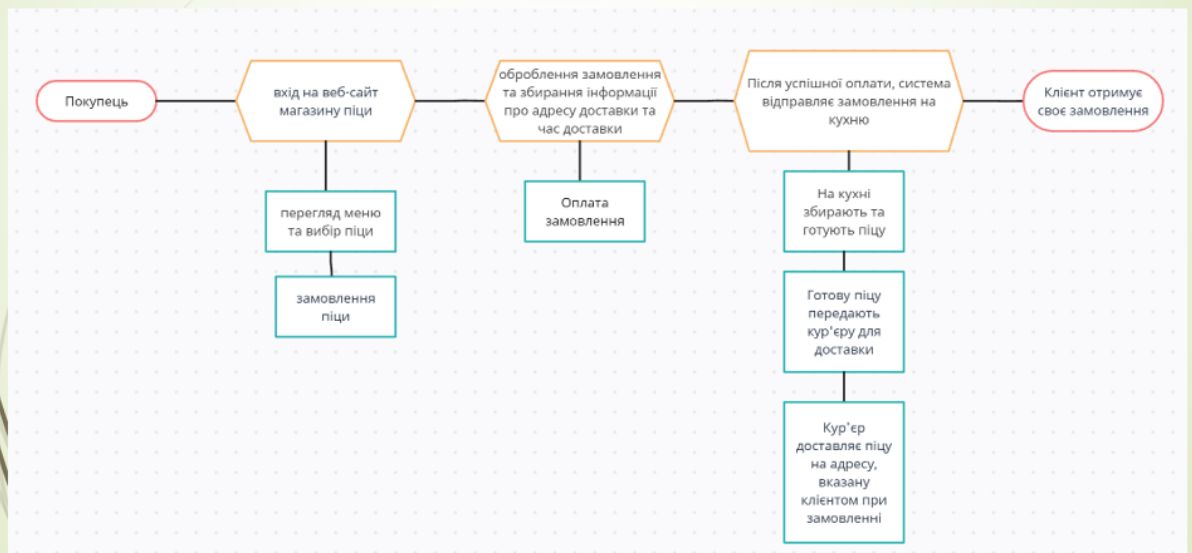


Рисунок В.6 – Слайд №6

Блок-схема кошика



Рисунок В.7 – Слайд №7

Блок-схема роботи головної сторінки магазину

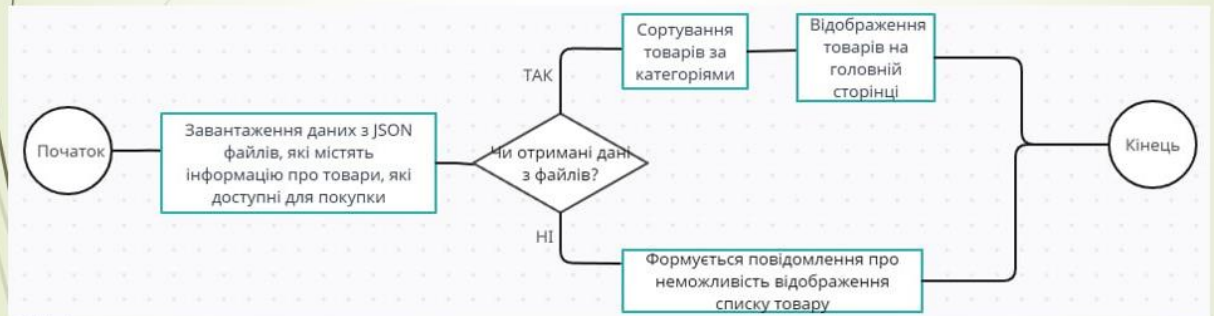


Рисунок В.8 – Слайд №8

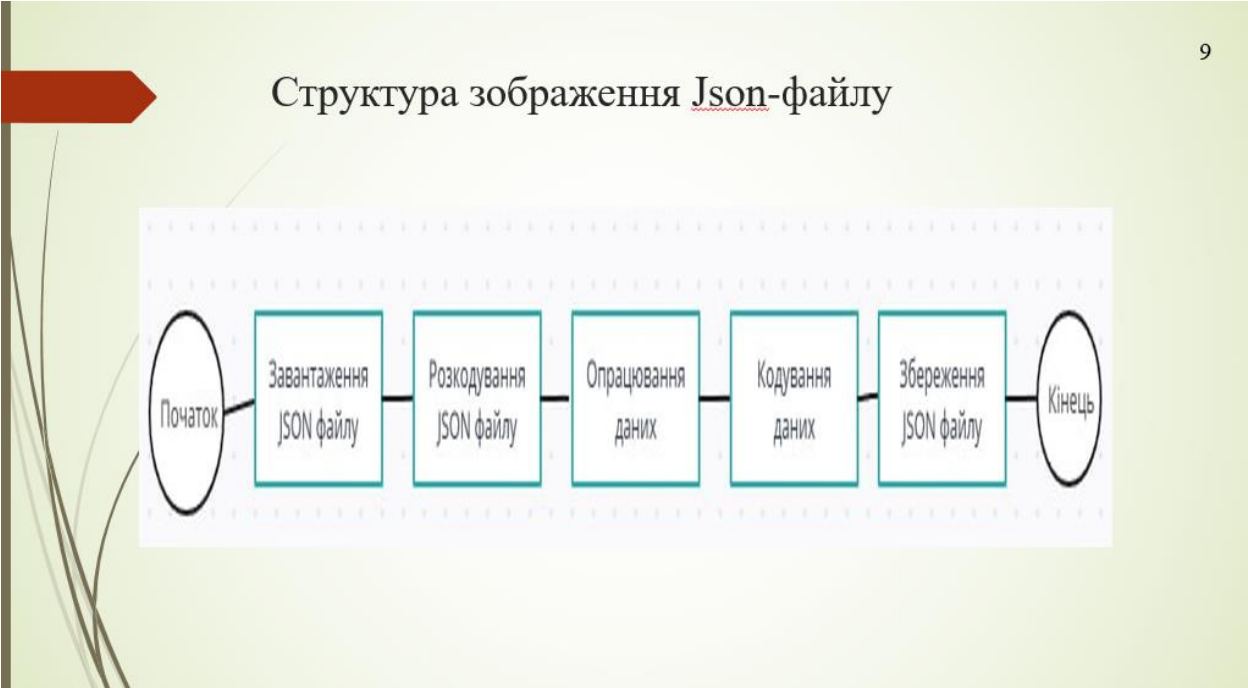


Рисунок В.9 – Слайд №9

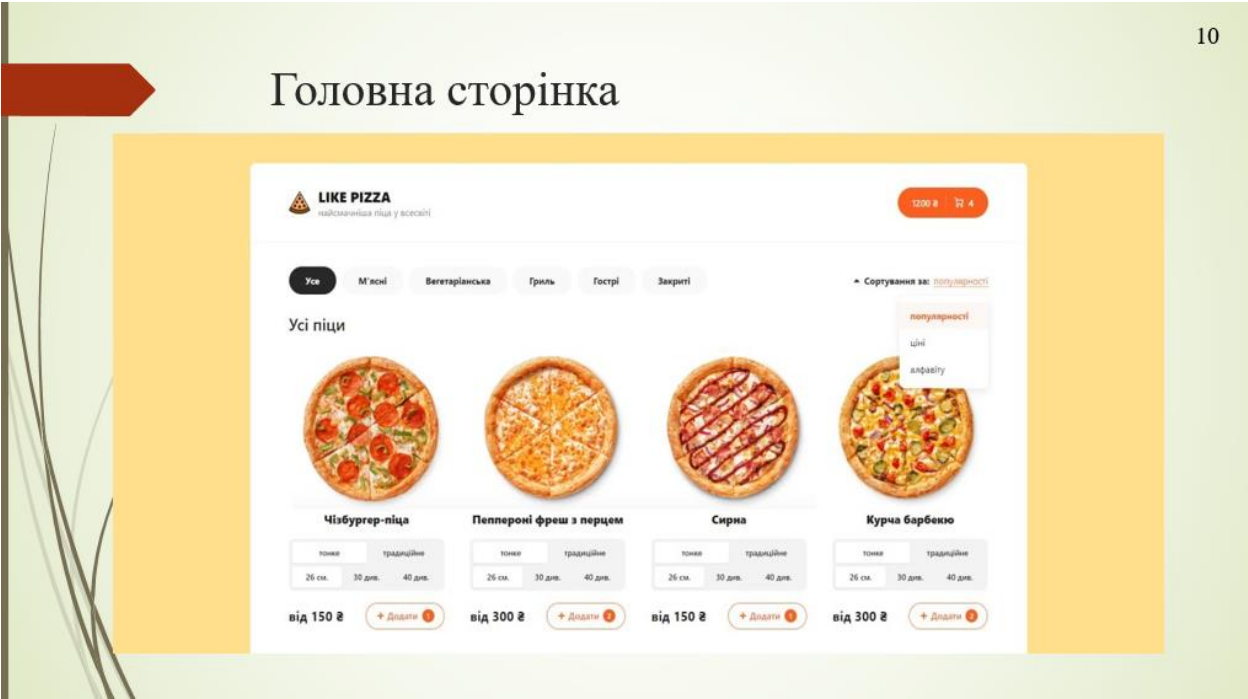


Рисунок В.10 – Слайд №10

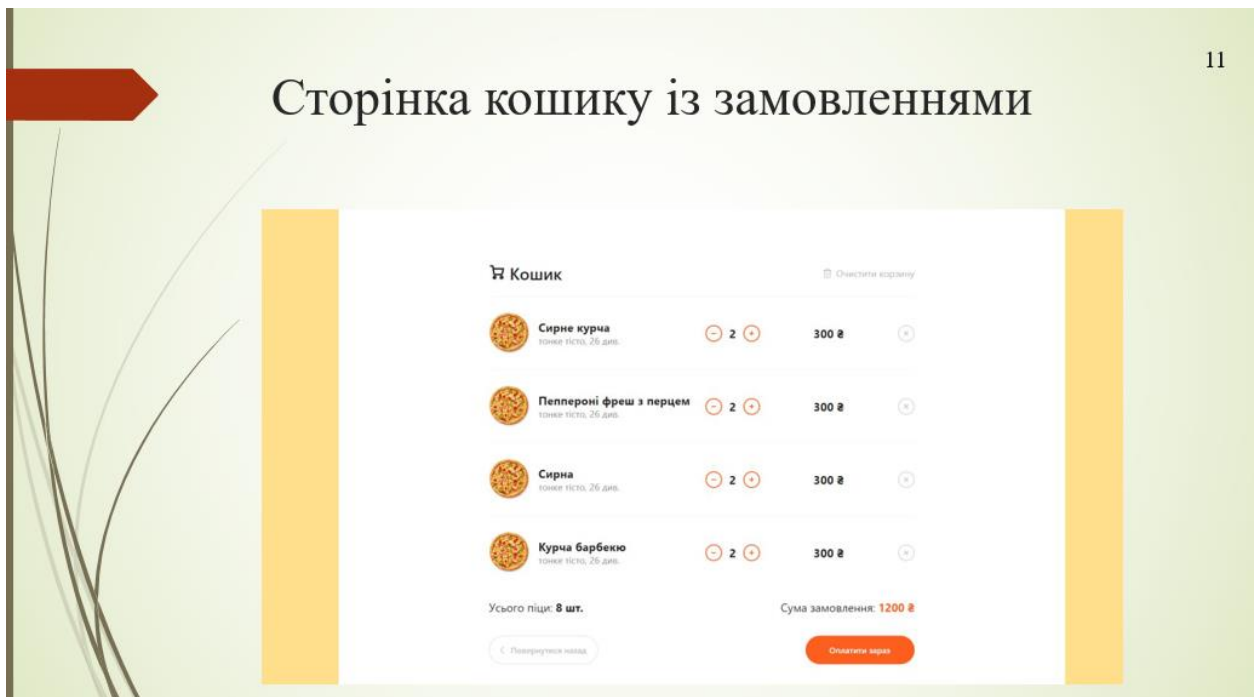


Рисунок В.11 – Слайд №11

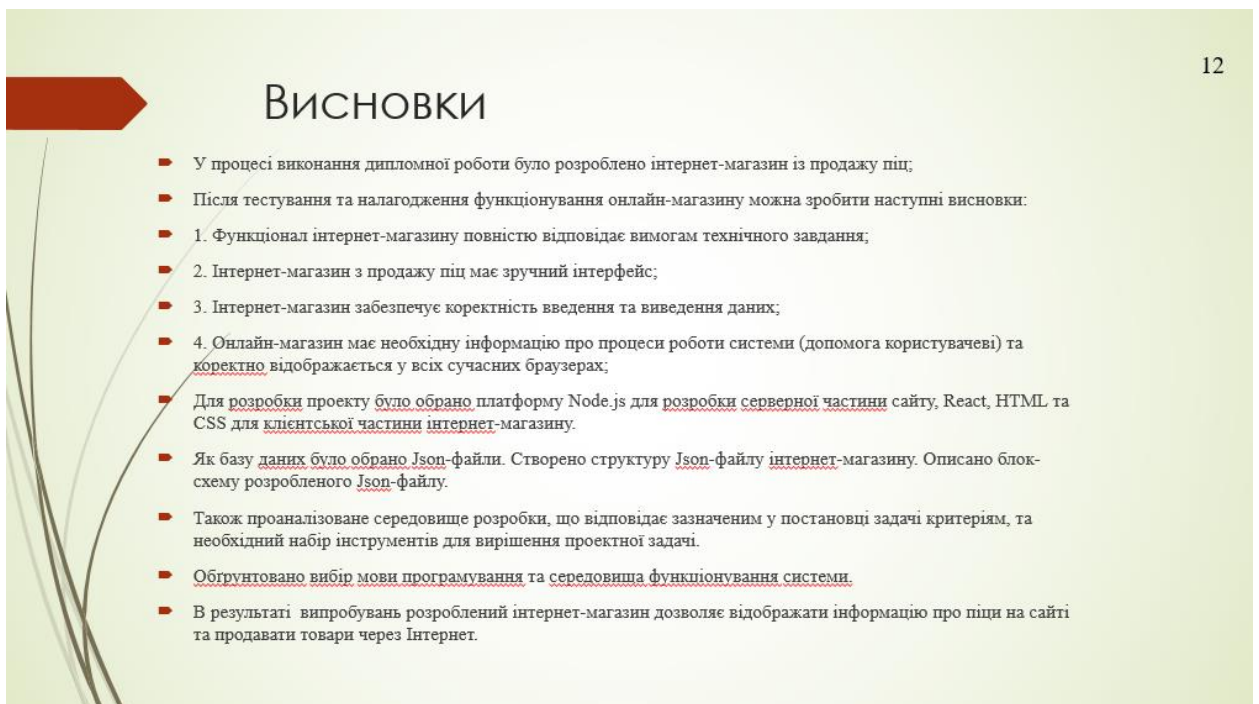


Рисунок В.12 – Слайд №12



Рисунок В.13 – Слайд №13