НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Інтегровані інформаційні системи»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Вебзастосунок для продажу кондитерських виробів»

Виконав:
студент IV курсу, групи ІА-92
Смородінов Віталій Олександрович _____

Керівник:
Асистент кафедри ІСТ
Нестерук Андрій Олександрович _____

Рецензент:
Старший викладач кафедри
інформатики та програмної інженерії
Халус Олена Андріївна _____

Засвідчую, що у цьому проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2023 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра інформаційних систем та технологій

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма «Інтегровані інформаційні системи»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Смородінову Віталію Олександровичу

1. Тема проєкту «Вебзастосунок для продажу кондитерських виробів», керівник проєкту Нестерук Андрій Олександрович, асистент кафедри ІСТ, затверджені наказом по університету від «31» травня 2023 р. № 2101-с.

2. Термін подання студентом проєкту: 12 червня 2023 року

3. Вихідні дані до проєкту:

Каталог продуктів, кошик покупок, користувачі, обробка замовлень

4. Зміст пояснювальної записки:
1. Аналіз існуючих систем: огляд існуючих застосунків, типи застосунків, порівняльний аналіз методів створення веб-застосунків
2. Структура системи: структура системи, вибір мови програмування для реалізації,обґрунтування вибору бібліотек, вибір середовища розробки, інформаційне забезпечення
3. Реалізація системи: змістовна постановка задачі, математична постановка задачі, опис існуючих методів розв'язання, обґрунтування методу розв'язання, специфікація функцій застосунку, діаграма компонентів,

діаграма станів, діаграма послідовності, діаграма потоків даних, діаграма зв'язків
4. Технологічний розділ: керівництво користувача, випробування продукту

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо)
1. Структурна схема
2. Діаграма компонентів
3. Діаграма станів
4. Діаграма потоків даних
5. ER-діаграма

6. Дата видачі завдання 13 березня 2023 року

Календарний план

№ з/п	Назва етапів виконання проєкту	Термін виконання етапів проєкту	Примітка
1.	Затвердження теми роботи	27.04.23	
2.	Аналіз рекомендаційних систем, як компонент існуючих рішень	02.05.23 – 09.05.23	
3.	Проектування користувацького інтерфейсу	09.05.23 – 12.05.23	
4.	Проектування бекенду та баз даних	12.05.23 – 16.05.23	
5.	Розробка користувацької архітектури	16.05.23 – 19.05.23	
6.	Розробка серверної архітектури	19.05.23 – 23.05.23	
7.	Реалізація програмного забезпечення	23.05.23 – 01.06.23	
8.	Розгортання системи	01.06.23 – 02.06.23	
9.	Тестування системи	02.06.23 – 06.06.23	
10.	Оформлення текстової документації	06.06.23 – 12.06.23	

Студент Віталій СМОРОДІНОВ

Керівник Андрій НЕСТЕРУК

АНОТАЦІЯ

Ключові слова: веб-застосунок, електронна комерція, база даних, рекомендації, фільтри, навігація.

Пояснювальна записка дипломного проєкту містить 4 розділи, 9 рисунків, 9 таблиць, 1 додаток, 26 джерел.

Метою розробки є створення веб-застосунку для продажу кондитерських виробів для облегшення покупки користувачам у режимі онлайн.

Для реалізації системи було використано: мови програмування HTML, JavaScript, CSS, бібліотека ReactJS, фреймворк NodeJS та модулі для нього, реляційна база даних PostgreSQL, середовище розробки Visual Studio Code.

У дипломному проєкті було розроблено працюючий веб-застосунок для продажу кондитерських виробів.

Дана розробка може підтримуватись надалі та використовуватись у цілях електронної комерції.

ANNOTATION

Keywords: web application, e-commerce, database, recommendations, filters, navigation.

The explanatory note of the diploma project contains 4 chapters, 9 figures, 9 tables, 1 appendix, 26 sources.

The goal of the development is to create a web application for the sale of confectionery products to facilitate online shopping for users.

To implement the system, the following programming languages were used: HTML, JavaScript, CSS, the ReactJS library, the NodeJS framework and modules for it, the PostgreSQL relational database, and the Visual Studio Code development environment.

In the diploma project, a working web application for the sale of confectionery products was developed.

This development may be further supported and used for e-commerce purposes.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІА92.200БАК.004 ПЗ	Пояснювальна записка	63		
6	А3	ІА92.200БАК.004 Д1	Вебзастосунок для продажу	1		
7			кондитерських виробів.			
8			Структурна діаграма			
9	А3	ІА92.200БАК.004 Д2	Вебзастосунок для продажу	1		
10			кондитерських виробів.			
11			Діаграма компонентів			
12	А3	ІА92.200БАК.004 Д3	Вебзастосунок для продажу	1		
13			кондитерських виробів.			
14			Діаграма станів			
15	А3	ІА92.200БАК.004 Д4	Вебзастосунок для продажу	1		
16			кондитерських виробів.			
17			Діаграма потоків даних			
18	А3	ІА92.200БАК.004 Д5	Вебзастосунок для продажу	1		
19			кондитерських виробів.			
20			ER-діаграма			
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІА92.200БАК.004 ТП		
Розроб.		Смородінов В.О.			Вебзастосунок для продажу кондитерських виробів. Відомість проєкту	Літ.	Аркуш	Аркушів
Керівн.		Нестерук А.О.				Т		1
						КПІ ім. Ігоря Сікорського Група ІА-92		
Затв.								

Пояснювальна записка
до дипломного проєкту
на тему: «Вебзастосунок для продажу кондитерських виробів»

Київ – 2023 року

ЗМІСТ

ВСТУП

У сучасному світі, де електронна комерція набуває все більшої популярності, веб-додатки для продажу товарів та послуг в Інтернеті стають невід'ємною складовою успішного бізнесу. Кондитерські вироби, які завжди були дуже популярними серед широкого загалу, не є винятком. У зв'язку з цим, розробка веб-застосунку для продажу кондитерських виробів є актуальною темою для дипломної роботи. Описуючи процес розробки такого додатку, можна проаналізувати технології та інструменти, які використовуються в індустрії електронної комерції, вивчити особливості маркетингу та продажу кондитерських виробів в Інтернеті, а також дослідити різноманітні аспекти взаємодії користувача з веб-додатком. У даній дипломній роботі буде розглянуто процес розробки веб-застосунку для продажу кондитерських виробів, включаючи проектування архітектури додатку, розробку бази даних, програмування функціоналу та тестування. При розробці веб-додатку для продажу кондитерських виробів необхідно враховувати такі фактори, як зручність та легкість використання для користувача, якість фотографій товарів, детальну інформацію про кожен продукт, можливість швидкої та безпечної оплати, швидку доставку та обслуговування.

У даному дипломному проєкті буде розглянуто основні етапи розробки веб-додатку для продажу кондитерських виробів, починаючи від визначення потреб клієнтів та аналізу конкурентів, а закінчуючи тестуванням та випуском готового додатку. В процесі розробки будуть використані такі технології, як HTML, CSS, JavaScript, ReactJS та PostgreSQL для розробки фронтенду та бекенду веб-додатку.

В додаток до цього, будуть розглянуті питання безпеки та захисту даних користувачів, оскільки важливо забезпечити конфіденційність та безпеку особистої інформації клієнтів. Для цього будуть використані різноманітні

методи та технології, такі як шифрування даних та використання захищених протоколів передачі даних.

Крім того, дана дипломна робота може стати корисною для компаній, що займаються виробництвом та реалізацією кондитерських виробів, які хочуть розширити свою онлайн присутність та залучити нових клієнтів. Розробка веб-додатку для продажу кондитерських виробів може стати важливим інструментом у підвищенні конкурентоспроможності компанії та розширенні її аудиторії для зміцнення позиії на ринку та збільшення прибутків.

Мета дослідження – підвищення ефективності реалізації продукту та забезпечення легкого зручного доступу користувачів до продукту, шляхом розробки веб-додатку для продажу кондитерських виробів.

Завданням дослідження є аналіз існуючих веб-додатків, визначення їх переваг та недоліків; розробці вимог до функціональності та дизайну веб-додатку, які будуть відповідати потребам користувачів та забезпечать зручність використання; розробці архітектури веб-додатку, що включає в себе структуру бази даних, логіку бізнес-процесів; розробити веб-додаток та забезпечити його подальшу підтримку та обслуговування.

Об'єктом дослідження є процес розробки веб-додатку для продажу кондитерських виробів. Це означає, що дослідженням охоплюється весь комплекс робіт по створенню веб-додатку, включаючи аналіз потреб користувачів, проектування інтерфейсу, розробку функціональності, тестування та оптимізацію додатку для забезпечення його безперебійної та ефективної роботи. Дослідження також охоплює питання реклами та просування веб-додатку в Інтернеті для залучення нових користувачів і підвищення обігу продажів.

Предметом дослідження даної дипломної роботи є аналіз існуючих додатків та розробка власного для продажу кондитерських виробів, технічна реалізація веб-додатку, його тестування, розробка функціональних

можливостей та забезпечення безпеки платежів та захисту особистої інформації користувачів.

Дипломний проєкт складається з наступних розділів: вступ, основні розділи, висновки, список використаних джерел із 19 найменувань, 1 додаток. Графічна частина включає 4 кресленики формату А3. Загальний обсяг 66 сторінки.

Отже, ця дипломна робота стане важливим кроком у підвищенні якості веб-додатків для продажу кондитерських виробів. Вона дозволить детально розібратися із основними етапами розробки веб-додатку, використовуючи актуальні технології та методи захисту даних.

1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

Із розвитком глобальної мережі Інтернет, почали з’являтися все більше онлайн-послуг. Серед них також чимало інтернет-магазинів та застосунків. Однією із галузей у цьому полі стали кондитерські магазини та пекарні. Їх популярність легко пояснити: далеко не в кожному місті є подібні заклади.

Із подальшим розвитком веб-технологій магазини почали модернізуватись, в них з’являлися нові можливості. Попит на кондитерські вироби зараз високий, і можна відмітити декілька найпопулярніших веб-застосунків у цій галузі.

Такі сайти як: CakeShop, Beze, Dolce Confections тощо міцно закріпились у списках лідерів, постійно набираючи все нових клієнтів. Окрім них, існують ще десятки інших, менш відомих магазинів, і їх стає все більше, адже попит росте. Дійсно, кондитерська продукція – це тема перспективна.

Виходячи із вище приведених фактів можна сказати, що обрана тема на сьогодні є популярною.

1.1 Огляд існуючих веб-застосунків

Веб-застосунки для кондитерських виробів – це онлайн-ресурси, що надають широкий спектр функцій для зручного вибору, замовлення та отримання кондитерських виробів. Ці веб-сайти використовуються для залучення клієнтів та полегшення процесу покупки.

Одним з ключових елементів веб-застосунка є його база даних, яка зберігає важливу інформацію про користувачів. Ця база даних містить дані про клієнтів, включаючи їх контактну інформацію, адреси доставки, інформацію про попередні замовлення та вподобання. Це дозволяє зберігати і використовувати персоналізовані дані для кожного користувача, що полегшує процес оформлення замовлення.

Веб-застосунок також має забезпечувати зручну навігацію та пошукові можливості для клієнтів [1]. Це може включати категорії продуктів, фільтри для пошуку за типом, смаком, розміром тощо. Детальна інформація про кожен продукт, включаючи фотографії, склад і характеристики, повинна бути легко доступною. Також можуть бути надані розділи з акціями, новими продуктами або рекомендаціями для клієнтів.

Важливою функцією веб-застосунку для кондитерських виробів є можливість зробити замовлення онлайн. Користувачі повинні мати можливість додавати продукти до кошика, вибирати опції, такі як кількість, дату та час доставки або самовивозу. Після оформлення замовлення користувачі повинні мати можливість здійснити безпечну онлайн-оплату через різні платіжні системи.

Зручність і прозорість доставки є ще однією важливою функцією веб-застосунку. Користувачі повинні мати можливість вибрати зручну дату та час доставки, а також вказати адресу доставки. Крім того, система відстеження доставки може бути використана для надання користувачам інформації про статус їх замовлення та орієнтований час доставки.

Нарешті, важливою додатковою функцією може бути система відгуків та рейтингу. Користувачі повинні мати можливість залишати відгуки про придбані кондитерські вироби та оцінювати їх якість та задоволення. Це може допомогти іншим користувачам при виборі продуктів і підвищити довіру до веб-застосунку.

1.1.1 Веб-застосунок CakeShop

Один із найпопулярніших українських кондитерських застосунків – cakeshop.com.ua [3]. Цей веб-сайт використовується не тільки для покупки вже готових солодких виробів. На ньому також можна замовити сировину для

місцевих пекарень, або всілякі пристрої для кондитера, починаючи формами для випікання та закінчуючи професійними пекарськими інструментами.
На головній сторінці магазину розміщені списки із категоріями товарів, показуючи новинки або товари зі знижкою. Зручний інтерфейс дозволяє вільно переміщатись серед категорій товарів. Є можливість зміни мови. Веб-застосунок не дає змоги зареєструватись в ньому, і це створює ряд мінусів. Наприклад, магазин не зможе використовувати дані про покупки для формування списку товарів до вподоби користувача та не зможе зберігати дані про адресу доставки товару.

1.1.2 Веб-застосунок Beze

Ще один популярний український застосунок – beze.com.ua [4]. Веб-сайт схожий по функціоналу до минулого, а саме: є списки новинок на головній сторінці, присутня можливість купляти інгредієнти для власної випічки. На відміну від CakeShop, тут є можливість реєстрації користувача. Присутня категорія “Обране”, де користувач по бажанню може зберегти товари до вподоби.

1.1.3 Веб-застосунок Dolce Confections

Зарубіжний веб-сайт dolceconfections.com орієнтований на більш широкий промисловий ринок [5]. Головна сторінка сайту не пропонує вибір категорій товарів, швидше категорії подій, на які можна б було замовити товар. На відміну від минулих застосунків, Dolce спеціалізується саме на готових кондитерських виробах та не продає окремі компоненти. На веб-застосунку можна зареєструватись, але не можна змінити мову. Також списку товарів, що були б до вподоби користувачу, немає. Однак застосунок має більш інтуїтивний та краще оформлений інтерфейс. Також присутня окрема вкладка

для відгуків – це хороший підхід до дизайну, адже користувачі можуть одразу переглянути їх, склавши думку про сервіс.

1.2 Методи та системи

У веб-застосунку для кондитерських виробів можуть використовуватись різні методи та системи для поліпшення функціональності та користувацького досвіду [6].

— Система управління замовленнями: ця система дозволяє користувачам зробити замовлення на кондитерські вироби через веб-застосунок. Вона повинна мати зручний інтерфейс для вибору продуктів, внесення спеціальних вимог, вибору дати та часу доставки тощо. Також, вона може включати функції оплати і відстеження стану замовлення.

— Системи рекомендацій: ця система може рекомендувати користувачам конкретні кондитерські вироби на основі їхніх вподобань, історії покупок або трендів. Вона може використовувати методи колаборативної фільтрації або системи на основі вмісту для здійснення рекомендацій.

— Система оглядів та рейтингу: ця система дозволяє користувачам залишати відгуки та оцінки для кондитерських виробів, які вони придбали. Це може допомогти іншим користувачам при виборі продуктів і збільшити довіру до веб-застосунку.

— Система відстеження доставки: якщо веб-застосунок надає послуги доставки кондитерських виробів, система відстеження доставки може бути корисною. Вона дозволяє користувачам відстежувати місцезнаходження свого замовлення в режимі реального часу та отримувати оновлення про його статус.

— Система каталогу продуктів: веб-застосунок може мати систему каталогу, яка детально описує кожен доступний кондитерський виріб. Вона

може містити зображення, опис, інгредієнти, рекомендації щодо зберігання та іншу важливу інформацію про продукти.

— Система розрахунку вартості: якщо веб-застосунок дозволяє користувачам замовляти кондитерські вироби за різними параметрами (розмір, декорації, складові тощо), система розрахунку вартості може автоматично обчислювати вартість замовлення на основі вибраних параметрів.

1.2.1 “Content-based” система

Content-based система (також відома як система на основі вмісту) – це метод рекомендаційної системи, який використовується для підбору і пропозицій об'єктів (таких як товари, статті, фільми або музика) на основі аналізу їх вмісту.

Основна ідея полягає в тому, щоб порівнювати характеристики (вміст) об'єктів, які сподівається рекомендувати, з характеристиками об'єктів, які користувач виразив попередній інтерес. Наприклад, якщо користувача зацікавили певні фільми жанру науково-фантастичного бойовика, система на основі вмісту може рекомендувати інші фільми з подібним вмістом.

Для роботи content-based системи необхідно мати набір властивостей (характеристик) кожного об'єкта, які можуть бути виміряні або визначені. Наприклад, для фільмів це можуть бути жанр, режисер, актори, рік випуску, опис сюжету тощо. Ці властивості формують векторне представлення об'єктів.

Після того як вектори характеристик створені для кожного об'єкта, можна застосувати різні методи обчислення схожості між ними, наприклад, косинусну схожість. Цей метод вимірює кут між векторами об'єктів, де більший кут вказує на меншу схожість, а менший кут – на більшу схожість.

Коли користувач вибирає об'єкт, система на основі вмісту знаходить інші об'єкти з найбільшими значеннями схожості і рекомендує їх

користувачеві. Наприклад, якщо користувач вибрав певну статтю, система може рекомендувати інші статті з подібною тематикою або авторством.

Однією з переваг content-based системи є те, що вона може надавати рекомендації, навіть коли немає достатньої кількості даних про взаємодію користувачів. Вона також може бути особливо корисною, коли користувачі виражають індивідуальні вподобання або коли об'єкти мають докладні властивості, які можуть бути використані для порівняння.

Проте, content-based система має свої обмеження. Наприклад, вона може мати обмежену здатність до виявлення нових і неочікуваних інтересів користувача, оскільки вона базується на аналізі властивостей об'єктів. Також можуть виникати проблеми з точністю рекомендацій, якщо властивості об'єктів не відображають дійсні вподобання користувача або якщо властивості недостатньо репрезентативні.

1.2.2 Система “Collaborative Filtering”

Системи колаборативної фільтрації – це тип рекомендаційних систем, які засновані на аналізі взаємодії користувачів з об'єктами (такими як товари, фільми, музика тощо) для надання індивідуальних рекомендацій. Ці системи використовують інформацію про вподобання користувачів і спираються на спільність у смаках та поведінці, щоб прогнозувати, які об'єкти можуть зацікавити конкретного користувача.

Системи колаборативної фільтрації можуть бути двох типів: базовані на спільній фільтрації та базовані на моделі.

— Системи колаборативної фільтрації на основі спільної фільтрації: У цих системах рекомендації надаються на основі історії взаємодії користувача з об'єктами. Аналізуючи спільність у смаках та взаємодії між користувачами, система може здогадатись, що якщо два користувачі сподобалися однаковим об'єктам у минулому, то ймовірність того, що їм

сподобаються інші спільні об'єкти, є висока. Цей підхід може використовувати методи, такі як підрахунок схожості між користувачами (користувач-користувач) або між об'єктами (об'єкт-об'єкт) на основі оцінок користувачів.

— Системи колаборативної фільтрації на основі моделі: Ці системи створюють модель, яка прогнозує рекомендації на основі взаємодії користувачів та об'єктів. Вони можуть використовувати методи машинного навчання, такі як алгоритми класифікації або регресії, щоб побудувати модель, яка прогнозує вподобання користувачів для непереглянутих об'єктів. Модель може враховувати багато факторів, таких як попередні оцінки користувача, властивості об'єктів та інші контекстні дані.

Перевагами систем колаборативної фільтрації є їх здатність до виявлення складних залежностей індивідуальних смаків та виходу за межі властивостей об'єктів. Вони можуть рекомендувати об'єкти, які користувач може бути не знав або не очікував, що додає елемент сюрпризу.

Однак, системи колаборативної фільтрації також мають свої обмеження, такі як проблеми холодного старту (коли немає достатньо даних про нового користувача або об'єкт) і проблеми масштабування до великої кількості користувачів та об'єктів.

1.2.3 Системи відстеження доставки

Система відстеження доставки є важливою складовою веб-застосунку, оскільки вона надає користувачам можливість відстежувати рух свого замовлення в режимі реального часу та отримувати оновлення про його статус. Основна мета системи відстеження доставки – забезпечити клієнтам зручність, а також знизити невизначеність щодо часу прибуття їхніх замовлень.

Основні компоненти системи відстеження доставки включають:

— Генерація унікального номера відстеження: Кожне замовлення отримує унікальний ідентифікатор або номер відстеження, який пов'язує його

з системою відстеження. Це дозволяє користувачам введення цього номера на веб-сайті або у додатку для перегляду статусу доставки.

— Інтеграція з доставковими службами: Система відстеження доставки повинна бути інтегрована зі службами доставки, які використовуються для перевезення замовлення. Це дозволяє отримувати оновлення про місцезнаходження та статус замовлення безпосередньо від служби доставки.

— Оновлення статусу доставки: Система автоматично отримує оновлення про статус доставки, такі як прийняття замовлення, підготовка до доставки, виїзд кур'єра, прибуття до пункту призначення тощо. Ці оновлення можуть включати часові мітки, інформацію про місцезнаходження та будь-які інші додаткові деталі.

— Сповіщення для користувачів: Користувачі повинні мати можливість отримувати сповіщення про зміни статусу доставки. Це може включати повідомлення по електронній пошті, смс-повідомлення або повідомлення через мобільний додаток. Сповіщення допомагають користувачам бути в курсі процесу доставки та вчасно реагувати на будь-які зміни або непередбачені обставини.

— Візуалізація на карті: Деякі системи відстеження доставки можуть візуалізувати маршрут доставки на карті. Користувачі можуть переглядати рух замовлення на мапі, що дозволяє їм зрозуміти, де саме знаходиться їхнє замовлення і як швидко воно наближається до їх місця призначення.

Висновки до розділу 1

Отже, у розділі описані найбільш популярні веб-застосунки. Кожен з них має свої особливості та переваги, але всі вони успішно привертають нових клієнтів і набирають популярність на ринку. Ці веб-застосунки дозволяють

замовляти та придбавати різноманітні кондитерські вироби онлайн, роблять процес вибору та покупки простим та зручним для користувачів.

Було оглянуто методи та системи, які допомагають покращити досвід роботи із застосунком. "Content-based" система та система "Collaborative Filtering" є двома широко використовуваними підходами для персоналізації рекомендацій кондитерських виробів для користувачів. Вони аналізують інформацію про смакові уподобання та історію покупок, щоб пропонувати індивідуально підібрані пропозиції.

Крім того, оглянуто системи відстеження доставки, які дозволяють клієнтам відстежувати статус свого замовлення та знаходитися в курсі процесу доставки. Це важлива функція, яка забезпечує прозорість та довіру між покупцем та продавцем.

Огляд тем, що піднімалися у розділі, надали важливої інформації щодо концептуалізації проєкту та його структури.

2 СТРУКТУРА СИСТЕМИ

Система може бути побудована за допомогою сучасних веб-технологій. Також вона потребує розробки реляційної бази даних.

Можна виділити наступні задачі для вирішення:

— розробка бази даних PostgreSQL;

— розробка алгоритмів для функціонування веб-застосунку та їх імплементація;

— розробка клієнтської частини застосунку;

— розробка серверної частини застосунку.

2.1 Структура системи

Веб-застосунок для продажу кондитерських виробів має наступну архітектуру. Він отримує інформацію з бази даних, такі як інгредієнти, рецепти, товари тощо. Ця інформація проходить валідацію та відправляється у клієнтську частину застосунку.

Важливим компонентом системи є база даних, яка містить інформацію про наявні кондитерські вироби, замовлення та клієнтську інформацію. Також присутні окремі фронтенд та бекенд компоненти, які дозволяють зручно презентувати та управляти продуктами, замовленнями та клієнтською інформацією.

Запити від користувачів надходять до API, яке виступає в ролі ретранслятора. API приймає та пересилає запити до відповідних модулів для обробки. У веб-застосунку також присутній модуль роботи з базою даних, який забезпечує доступ до необхідної інформації для виконання запитів та збереження оновлень.

Структурну діаграму наведено у кресленику ІА92.200БАК.004 Д1.

2.2 Вибір мови програмування

Вибір мови програмування є важливим етапом при розробці веб-застосунку, оскільки це визначає технології та інструменти, які будуть використовуватись для створення застосунку. Зазвичай для створення веб-застосунків використовують такі мови як HTML, CSS та JavaScript, які працюють у стеку, доповнюючи одне одного. Також замість JavaScript нерідко використовують PHP для серверної архітектури [7].

2.2.1 Мова розмітки HTML

Мова HTML (HyperText Markup Language) - це стандартна мова розмітки для створення та представлення веб-сторінок. HTML використовується для створення структури та визначення вмісту веб-сторінок, включаючи текст, зображення, посилання, таблиці, форми тощо [8].

Переваги HTML:

— Простота вивчення: HTML має простий синтаксис, що робить його легким для вивчення навіть для початківців. Він використовує зрозумілі теги та атрибути для опису структури сторінки.

— Сумісність з браузерами: HTML є стандартом для веб-розробки і підтримується всіма сучасними браузерами. Це означає, що веб-сторінки, створені з використанням HTML, будуть відображатись коректно на різних платформах та пристроях.

— Широке співвідношення з CSS та JavaScript: HTML використовується для створення структури та міток веб-сторінок, а для стилізації та взаємодії використовуються CSS та JavaScript відповідно. Ця комбінація дозволяє розробникам створювати веб-сторінки з красивим дизайном та взаємодією.

— Підтримка мультимедіа: HTML підтримує вбудовування зображень, аудіо та відео контенту безпосередньо на веб-сторінки. Це дозволяє створювати багатомедійні веб-сторінки з різноманітним контентом.

Проте, у HTML відсутня динамічна функціональність: HTML зосереджений на структурі та вмісті сторінки. Тож для динамічності необхідно використовувати сторонні мови, наприклад, JavaScript.

2.2.2 Мова стилів CSS

CSS (Cascading Style Sheets) – це мова стилів, яка використовується для визначення зовнішнього вигляду та форматування веб-сторінок, створених за допомогою HTML або інших мов розмітки [9].

Переваги CSS:

— Розділення змісту та представлення: CSS дозволяє розділити зміст сторінки (HTML) від її вигляду та стилів. Це дозволяє змінювати вигляд сторінки без необхідності змінювати її вміст. Такий підхід полегшує редагування та підтримку веб-сайту.

— Консистентність та повторне використання: CSS дозволяє визначати стилі один раз та застосовувати їх до різних елементів на сторінці або на різних сторінках веб-сайту. Це спрощує створення консистентного вигляду та забезпечує повторне використання коду.

— Гнучкість та контроль над виглядом: CSS надає широкі можливості для впливу на вигляд елементів веб-сторінки. З його допомогою можна задавати кольори, шрифти, розташування, розміри, фонові зображення, анімацію та інші атрибути, що визначають зовнішній вигляд елементів.

CSS є потужним інструментом для стилізації веб-сторінок та забезпечення їх вигляду та немає аналогів. Він дозволяє розділити зміст та представлення, надає гнучкість та контроль над виглядом, а також сприяє підтримці консистентного дизайну.

2.2.3 Мова програмування JavaScript

JavaScript – це високорівнева, інтерпретована мова програмування, що використовується для надання динаміки та інтерактивності веб-сторінкам [10]. Найчастіше використовується для створення сценаріїв вебсторінок, що надає можливість на боці клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд вебсторінки. JavaScript класифікують як прототипну (підмножина об'єктно-орієнтованої), скриптову мову програмування з динамічною типізацією. Окрім прототипної, JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну) і деякі відповідні архітектурні властивості, зокрема: динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування, функції як об'єкти першого класу.

Переваги JavaScript:

— Клієнтська та серверна сторона: JavaScript може бути використаний як на клієнтській стороні (у браузері), так і на серверній стороні (з використанням платформи, такої як Node.js). Це дозволяє розробникам створювати повноцінні веб-додатки, які взаємодіють з сервером та користувачем.

— Інтерактивність: JavaScript дозволяє додавати інтерактивні ефекти, валідацію форм, анімацію, динамічні зміни на сторінці та взаємодію з користувачем без необхідності перезавантаження сторінки.

— Розширення функціональності веб-сторінок: JavaScript дозволяє підключати сторонні бібліотеки та фреймворки, які розширюють можливості мови та спрощують розробку веб-додатків.

— Багата екосистема: JavaScript має велику спільноту розробників та широкий вибір інструментів, бібліотек, фреймворків та розширень, які допомагають збільшити продуктивність та розширити можливості розробки.

Недоліки JavaScript:

— Залежність від клієнта: JavaScript виконується на боці клієнта, тому він залежить від підтримки браузером та доступності JavaScript у пристроях користувачів. Різні браузери можуть мати різні реалізації та підтримку стандартів, що може вимагати додаткової роботи для забезпечення сумісності.

— Безпека: Виконання JavaScript на боці клієнта відкриває можливість для атак, таких як впровадження зловмисного коду або перехоплення даних користувача. Однак, застосування безпечних практик розробки та заходів безпеки можуть зменшити ризики.

— Виконання на клієнтському пристрої: Оскільки JavaScript виконується на клієнтському пристрої, це може призвести до збільшення навантаження на процесор та пам'ять пристрою, особливо для важких або складних додатків.

JavaScript використовується для створення динамічного веб-контенту, який може реагувати на дії користувача без перезавантаження сторінки. Це включає валідацію форм, анімацію, динамічне оновлення даних та багато іншого. Також JavaScript дозволяє взаємодіяти зі сторонніми API, такими як соціальні мережі, картографічні сервіси, сервіси платежів тощо. Це дозволяє отримувати та надсилати дані з зовнішніх джерел та інтегрувати їх у веб-додаток. JavaScript використовується для розробки веб-додатків, а застосування фреймворків, таких як React, Angular або Vue.js, дозволяє прискорити розробку та покращити організацію коду.

2.2.4 Мова програмування PHP

PHP – це скриптова мова програмування, яка широко використовується для розробки веб-додатків та веб-сайтів [11].

Переваги PHP:

— Широке використання: PHP є однією з найпопулярніших мов програмування для розробки веб-додатків. Він має велику спільноту розробників, багато розширень та фреймворків, що сприяє швидкій розробці та розширенню функціональності проектів.

— Простота використання: PHP має простий та зрозумілий синтаксис, що дозволяє розробникам швидко освоювати мову та виконувати завдання з мінімальними зусиллями.

— Інтеграція з веб-серверами: PHP є серверною мовою, яка працює безпосередньо на стороні сервера. Він може інтегруватися з різними веб-серверами (такими як Apache, Nginx) та базами даних (такими як MySQL), що забезпечує потужні можливості веб-розробки.

Попри його переваги, існує чимало недоліків:

— Незручність при роботі з асинхронними операціями: PHP за замовчуванням не має сильної підтримки для асинхронного програмування, що може призвести до проблем з продуктивністю та масштабованістю додатків, особливо в великих проектах.

— Застарілість: Деякі аспекти PHP вважаються застарілими, зокрема старі версії мови мають деякі недоліки та проблеми безпеки.

— Масштабованість: В деяких випадках PHP може мати обмежену масштабованість для великих проектів з великою кількістю користувачів, оскільки він виконується на рівні сервера та вимагає більше ресурсів.

Через ці мінуси було прийнято рішення використовувати замість PHP середовище виконання Node.js для JavaScript.

2.2.5 Середовище виконання Node.js

Node.js – це кросплатформне серверне середовище з відкритим кодом [12]. Воно дозволяє розробникам використовувати JavaScript для написання інструментів командного рядка та сценаріїв на стороні сервера. Можливість запуску коду JavaScript на сервері часто використовується для створення динамічного вмісту веб-сторінки перед тим, як сторінка надсилається у веб-браузер користувача. Отже, Node.js представляє парадигму «JavaScript скрізь», об'єднуючи розробку веб-додатків навколо однієї мови програмування, на відміну від використання різних мов для програмування на стороні сервера чи клієнта.

Переваги Node.js:

— Висока продуктивність: Node.js використовує асинхронну та подієву модель програмування, що дозволяє обробляти багато запитів одночасно без блокування потоку виконання. Це призводить до високої продуктивності та масштабованості додатків.

— Використання JavaScript: За допомогою Node.js розробники можуть використовувати одну мову - JavaScript, яка виконується як на сервері, так і на клієнті. Це спрощує розробку та спільне використання коду між серверною та клієнтською частиною додатка.

— Велика екосистема: Node.js має велику кількість модулів та бібліотек, доступних через пакетний менеджер npm. Це дозволяє розробникам швидко використовувати готові рішення для різних задач, таких як маршрутизація, робота з базами даних, робота з мережевими протоколами та багато іншого.

Недоліком можна вважати синхронність – Node.js базується на однопотоковому моделі виконання, що може стати проблемою при обробці великих обчислювальних завдань або блокуючих операцій. Проте, ця

проблема вирішується використанням асинхронних операцій та функцій, що вбудовані у мову.

2.2.6 Порівняльна характеристика мов програмування

У таблиці 2.1 наведено порівняльну характеристику мов програмування застосунку:

Таблиця 2.1 – порівняльна характеристика мов програмування

Назва	**Використання**	**Застосування**	**Порівняння**
HTML	Фронтенд	Розмітка веб-сторінки	Не має аналогу
CSS	Фронтенд	Стилізація веб-сторінки	Не має аналогу
JavaScript	Фронтенд/Бекенд (у вигляді Node.js)	Надання динаміки та інтерактивності веб-сторінкам	Не має аналогу
PHP	Бекенд	Написання серверних скриптів	Застарілий, на відміну від Node.js
Node.js	Бекенд	Використання JavaScript на серверній частині	Сучасний та скрізь використовується

Отже, для клієнтської частини було обрано технологічний стек HTML/CSS/JavaScript, як такий, що не має аналогів. Для серверної частини був обраний Node.js як сучасний.

2.3 Вибір фреймворків

У процесі розробки застосунку можуть бути використані декілька фреймворків для полегшення виконання поставленої задачі. Фреймворки веб-розробки – це набір інструментів, бібліотек та стандартів, які допомагають розробникам будувати веб-додатки та веб-сайти швидше та ефективніше [13]. Фреймворки зазвичай надають структуру та організацію проекту, що допомагає розробникам тримати код організованим, легким для розуміння та підтримки. Фреймворки надають готові компоненти, модулі та функціональність, які можна використовувати для швидкої розробки. Це зменшує необхідність писати код з нуля та прискорює процес розробки. Також вони зазвичай встановлюють стандарти та найкращі практики, що допомагає розробникам писати якісний та стабільний код. Вони сприяють використанню кращих практик безпеки, оптимізації та сумісності.

2.3.1 Фреймворк React.js

React.js – це один з найпопулярніших фреймворків JavaScript для розробки інтерфейсів користувача [14]. React.js базується на компонентній архітектурі, що дозволяє розбити інтерфейс на незалежні компоненти. Компоненти можна повторно використовувати та збирати разом, що спрощує розробку та підтримку коду. React.js використовує віртуальний DOM (Document Object Model), що дозволяє ефективно оновлювати лише необхідні частини сторінки під час зміни даних. Це забезпечує швидку та ефективну роботу інтерфейсу. React.js дозволяє розробникам створювати ефективні та високоякісні інтерфейси користувача, що легко масштабувати та підтримувати. Він чудово підходить для розробки односторінкових додатків, веб-сайтів та мобільних додатків.

2.3.2 Фреймворк Angular

Angular – це ще один популярний фреймворк для розробки веб-додатків. Angular використовує мову програмування TypeScript, яка є суперсетом JavaScript [15]. TypeScript додає сильну типізацію, об'єктно-орієнтовану парадигму та інші функції, що допомагають писати більш безпечний та організований код. Angular побудований на компонентній архітектурі, де кожен елемент інтерфейсу представляється окремим компонентом. Це спрощує організацію коду та робить його більш повторно використовуваним та легко змінюваним. Angular є повним фреймворком, що надає не лише інструменти для роботи з інтерфейсом користувача, але й для керування станом додатку, маршрутизації, валідації форм, роботи з HTTP запитами та багатьох інших аспектів розробки. Angular використовує декларативний синтаксис, який дозволяє описувати структуру інтерфейсу та поведінку компонентів за допомогою шаблонів та директив. Це робить код більш зрозумілим та підтримуваним.

Незважаючи на те, що Angular та React є схожими фреймворками для розробки веб-додатків, вони мають деякі відмінності та недоліки. Наприклад, Angular є досить складним фреймворком, особливо для новачків. Він має значну кількість концепцій, правил та архітектурних рішень, що вимагають певного часу та зусиль для їх освоєння. Також Angular накладає певні обмеження та шаблони на розробку додатків. Це може призвести до меншої гнучкості та варіативності в порівнянні з React, де розробник має більше свободи у виборі підходів та структури проекту. Порівняно з React, тестування Angular додатків може бути складнішим, особливо через залежність від фреймворку та побудову додатку навколо Angular-специфічних концепцій.

Через надмірну складність було вирішено відкинути Angular.

2.3.3 Фреймворк Vue.js

Vue.js – це прогресивний фреймворк для розробки користувацького інтерфейсу, який зосереджується на простоті використання та гнучкості [16]. Один з основних плюсів Vue.js – це його простота використання. Він має дружній та зрозумілий синтаксис, що дозволяє швидко освоювати його навички навіть новачкам. Vue.js пропонує підхід, який легко інтегрується в існуючі проекти. Vue.js надає більшу гнучкість у порівнянні з React. Він дозволяє використовувати компоненти як в шаблонах, так і в скриптах, що дозволяє розробникам використовувати підхід, який найкраще підходить для їх потреб. Vue.js має вбудовану систему реактивності, що дозволяє автоматично оновлювати компоненти при зміні даних. Це спрощує управління станом додатку та автоматизує оновлення інтерфейсу користувача.

Хоча екосистема Vue.js швидко зростає, вона все ще не настільки розширена, як у React. Це означає, що може бути складніше знайти плагіни, інструменти та бібліотеки, які підходять для конкретних потреб. Тому Vue.js було вирішено не використовувати.

2.3.4 Фреймворк Express.js

Express.js – це легкий та гнучкий веб-фреймворк для створення серверної частини додатків на основі Node.js [17]. Express відомий своїм мінімалістичним підходом. Він надає лише основні функції та структуру, не нав'язуючи жорстких правил або обмежень. Це дозволяє розробникам більшу свободу в розробці та архітектурі своїх додатків. Він має простий та лаконічний синтаксис, що робить його легким у використанні. Він пропонує інтуїтивно зрозумілі методи та функції для роботи з HTTP-запитами, шляхами та параметрами. Це дозволяє розробникам швидко створювати серверні додатки. Middleware є ключовою особливістю Express. Він дозволяє додавати

функціональність до запитів та відповідей перед тим, як вони досягають фінального обробника запиту.

2.3.5 Порівняльна характеристика фреймворків

У таблиці 2.2 наведено порівняння фреймворків, описаних вище:

Таблиця 2.2 – Порівняння фреймворків

Назва	**Застосування**	**Складність**	**Розмір**	**Реактивність**
React.js	Розмітка веб-сторінки	Легко вивчити базовий синтаксис	Невеликий розмір	Віртуальний DOM
Angular	Стилізація веб-сторінки	Має високий поріг входження	Великий розмір	Двосторонній зв'язок
Vue.js	Надання динаміки та інтерактивності веб-сторінкам	Легко вивчити і розпочати розробку	Компактний розмір	Двосторонній зв'язок
Express.js	Спрощення створення серверної частини	Легкий синтаксис	Невеликий розмір	–

Для серверної частини було обрано Express.js як найновіший серед серверних фреймворків. Для клієнтської частини було обрано React.js через легкість синтаксису та більш широкий функціонал, ніж у Vue.js

2.4 Вибір бібліотек JavaScript

Бібліотеки представляють собою збірку попередньо написаних функцій, класів, компонентів або модулів, які можна використовувати у своєму програмному коді. Вони дозволяють розширити функціональність програми, прискорити розробку, забезпечити готові рішення для поширених задач та полегшити підтримку коду. Бібліотеки надають готові функції та компоненти, які можна використовувати в різних проектах [18].

— Fingerprint.js – бібліотека для отримання “відбитка пальця” браузера або пристрою, що може використовуватись для унікальної ідентифікації користувача або пристрою. Ця бібліотека використовується для створення веб-аналітики, захисту від шахрайства або для розпізнавання користувачів на основі їх відбитка пальця.

— Bluebird – бібліотека реалізації обіцянок (Promises) для JavaScript, яка надає розширені можливості для керування асинхронним кодом. Вона дозволяє зручно працювати з асинхронним кодом, включаючи послідовність асинхронних запитів, обробку помилок та інші операції з промісами.

— i18n – бібліотека для локалізації веб-додатків, яка дозволяє розгортати веб-додатки на різних мовах. i18n надає механізми для заміни текстових ресурсів, форматування дати та часу, чисел та інших елементів інтерфейсу, що залежать від мови та культури.

— jQuery – це легковага бібліотека JavaScript, яка спрощує взаємодію з HTML-документами, обробку подій, анімацію та інші завдання. jQuery дозволяє зручно вибирати та маніпулювати HTML-елементами, виконувати

асинхронні запити до сервера, створювати анімаційні ефекти та спрощувати роботу з подіями на стороні клієнта.

2.5 Вибір бази даних

База даних – структурована колекція даних, організованих та збережених з урахуванням специфічних правил та форматів [19]. Вона служить для ефективного зберігання, керування та доступу до великого обсягу інформації.

У веб-застосунку база даних може бути використана для збереження різноманітної інформації, яка потрібна для правильної роботи та взаємодії. Основними типами баз даних, які можуть бути використані, є реляційні бази даних та нереляційні.

Реляційна база даних (Relational Database) – це тип бази даних, який організований у вигляді таблиць зі зв'язками між ними. Кожна таблиця має стовпці та рядки, де стовпці представляють атрибути, а рядки – записи. Реляційні бази даних використовують мову SQL (Structured Query Language) для здійснення операцій з даними.

Нереляційна база даних (NoSQL Database) – тип бази даних, що використовує іншу модель зберігання даних, не обмежену таблицями та зв'язками. Вони можуть використовувати документи, ключ-значення, стовпці, графи та інші структури для зберігання даних. Нереляційні бази даних часто використовуються для обробки великого обсягу даних або для специфічних вимог проекту.

2.5.1 Oracle

Oracle – це одна з провідних реляційних баз даних на ринку, розроблена компанією Oracle Corporation. Вона має багатий функціонал та широкі можливості для зберігання та обробки даних.

Oracle відома своєю високою надійністю і стійкістю. Вона має механізми резервного копіювання, відновлення даних та відмовостійкості, що дозволяє забезпечити постійну доступність даних; має оптимізований двигун запитів, який забезпечує швидку обробку запитів і виконання операцій з великими обсягами даних; надає багатий функціонал, такий як транзакційна підтримка, засоби безпеки даних, аналітичні можливості, робота з географічними даними та інші.

Незважаючи на її переваги, Oracle відноситься до комерційного програмного забезпечення, тому вона є дорогою для використання. Також Oracle має досить складну архітектуру і вимагає глибоких знань для його налагодження та адміністрування. Використання Oracle може потребувати навчання та досвіду для досягнення оптимальних результатів.

2.5.2 MySQL

MySQL – ще одна популярна реляційна база даних, також розроблена компанією Oracle Corporation. Вона використовує мову запитів SQL для зберігання, організації та управління даними.

MySQL є відкритою базою даних, що означає, що ви можете безкоштовно використовувати, змінювати та розповсюджувати її. Це дозволяє широке співтовариство розробників спільно працювати над її вдосконаленням. MySQL має простий та інтуїтивно зрозумілий інтерфейс, що спрощує створення та керування базами даних. Вона також має багатий набір інструментів для розробників, які спрощують роботу з базою даних. Також

вона має механізми забезпечення надійності, такі як резервне копіювання, відновлення даних та механізми відмовостійкості, що дозволяє забезпечити стабільну роботу системи та захист даних.

Однак, у порівнянні з деякими комерційними базами даних, MySQL може мати обмежені функціональні можливості. Деякі продвинуті функції, такі як реплікація даних або управління транзакціями, можуть бути менш розвинуті у MySQL. Також, у порівнянні із іншими рішеннями, MySQL може не мати деяких функцій, які можуть бути важливими для деяких проєктів.

2.5.3 MongoDB

MongoDB – це документо-орієнтована нереляційна база даних, яка забезпечує гнучку та масштабовану систему збереження та управління даними.

MongoDB використовує гнучку структуру даних, відому як документи BSON (Binary JSON). Це дозволяє зберігати дані без строго заданої схеми, що робить його особливо корисним для проектів зі змінними або невизначеними схемами даних. MongoDB пропонує швидкий доступ до даних завдяки своїй архітектурі, яка використовує індекси та кешування, щоб забезпечити ефективну роботу з даними. Він також підтримує паралельні запити та розподілені операції, що покращує продуктивність при обробці великих обсягів даних.

Однак, MongoDB вимагає значних обсягів оперативної пам'яті для оптимальної роботи. База даних може вимагати більше дискового простору порівняно з традиційними реляційними базами даних, особливо при використанні реплікації та резервного копіювання даних, також вона має обмежену підтримку транзакцій, зокрема на рівні кількох документів. Це може бути проблемою для деяких додатків, які потребують суворої консистентності даних.

2.5.4 PostgreSQL

PostgreSQL – це потужна реляційна база даних з відкритим вихідним кодом, яка пропонує розширені можливості для збереження та управління даними.

PostgreSQL надає багато розширених можливостей, включаючи підтримку складних SQL-запитів, транзакцій, керування конкурентністю, підтримку географічних даних та багато іншого. Він також підтримує різні типи даних, включаючи JSON, XML та геометричні типи. База даних має вбудовану підтримку транзакцій та механізми відновлення, що дозволяють забезпечити надійність та цілісність даних. Вона також підтримує механізми реплікації, які дозволяють створювати резервні копії та забезпечувати високу доступність бази даних. Postgres надає розширені засоби безпеки, включаючи автентифікацію, авторизацію та шифрування даних. БД має вбудовану підтримку ролей користувачів, контроль доступу до об'єктів бази даних та можливість налаштування аудиту дій користувачів.

Однак, PostgreSQL може бути складним у налаштуванні та керуванні, особливо для новачків. Він вимагає розуміння основних принципів реляційних баз даних та знань SQL-запитів. Також PostgreSQL вимагає певних обсягів пам'яті та обчислювальних ресурсів для оптимальної роботи. Це може бути фактором, особливо при роботі з великими обсягами даних.

Для проєкту було обрано реляційну базу даних PostgreSQL як оптимальну.

2.5.5 Порівняльна характеристика баз даних

У таблиці 2.3 наведено порівняння баз даних, описаних вище.

Таблиця 2.3 – Порівняння баз даних

Назва	**Тип**	**Масштабованість**	**Транзакційна цілісність**
Oracle	Реляційна	Висока	Так
MySQL	Реляційна	Середня	Так
MongoDB	Нереляційна	Висока	Ні
PostgreSQL	Реляційна	Висока	Так

Оскільки всі бази даних приблизно однакові у технічних характеристиках, було обрано PostgreSQL як повністю безкоштовну базу даних.

2.6 Розподілене сховище Redis

Redis (Remote Dictionary Server) – це відкрита, високопродуктивна система управління даними, яка працює у операційній пам'яті [20]. Він дозволяє зберігати, кешувати та обробляти дані в реальному часі. Redis підтримує різноманітні типи даних, такі як рядки, хеші, списки, набори та сортовані множини, що робить його вельми гнучким для використання в різних сценаріях.

Redis надає кілька основних функцій, які роблять його цінним інструментом для розробників:

— Швидкість: Redis виконує операції в пам'яті, що забезпечує дуже швидкий доступ до даних. Це робить його особливо корисним для сценаріїв, де вимагається низька латентність, наприклад, кешування часто використовуваних даних;

— Кешування: Redis може використовуватись як кеш для зберігання часто використовуваних даних у пам'яті. Це дозволяє значно зменшити навантаження на основну базу даних та покращити швидкість обробки запитів.

— Атомарні операції: Redis забезпечує можливість виконувати атомарні операції над даними, що робить його корисним для сценаріїв, де потрібно гарантувати консистентність даних.

Redis можна використовувати у багатьох сферах, включаючи розробку веб-додатків, аналітику в реальному часі, кешування даних, сесійне зберігання, роботу з чергами повідомлень та багато іншого.

2.7 Вибір середовища розробки

Вибір середовища розробки для веб-застосунку є важливим етапом у розробці програмного забезпечення. Одними з найпопулярніших середовищ розробки для програмування веб-застосунків є Visual Studio Code, Sublime Text та WebStorm.

2.7.1 Visual Studio Code

Visual Studio Code (VS Code) є безкоштовним та відкритим середовищем розробки, розробленим компанією Microsoft [21]. VS Code підтримує Windows, macOS та Linux, що дозволяє розробникам працювати на будь-якій операційній системі за їхнім вибором. Велика кількість розширень та плагінів доступна для VS Code, що дозволяє налаштувати середовище розробки під свої потреби та використовувати різноманітні інструменти. VS Code має інтуїтивно зрозумілий та добре організований інтерфейс користувача, який сприяє зручній навігації та редагуванню коду.

2.7.2 Sublime Text

Sublime Text є популярним комерційним середовищем розробки, відомим своєю швидкістю та легковагістю [22]. Sublime Text працює дуже

швидко, завдяки чому розробникам не потрібно чекати на завантаження та відкриття файлів. Він має багато корисних функцій, таких як швидке перетворення тексту, автодоповнення, підсвічування синтаксису, розширення та багато інших.

2.7.3 WebStorm

WebStorm є інтегрованою розробкою середовищем (IDE) для веб-розробки, розробленим компанією JetBrains [23]. WebStorm надає потужний редактор коду з автодоповненням, підсвічуванням синтаксису, перевіркою помилок та багатьма іншими функціями, що полегшують роботу з кодом. WebStorm надає потужні інструменти для рефакторингу коду, що дозволяє швидко та безпечно змінювати структуру та функціонал програми. Проте, WebStorm вимагає значних обсягів пам'яті та процесорної потужності, особливо при роботі з великими проектами, що може обмежувати його використання на менш потужних комп'ютерах.

2.7.4 Порівняльна характеристика середовищ розробки

У таблиці 2.4 наведено порівняння середовищ розробки, описаних вище.

Таблиця 2.4 – Порівняння середовищ розробки

Назва	**Вартість**	**Розширюваність**	**Швидкість та продуктивність**
Visual Studio Code	Безкоштовний	Велика кількість розширень	Швидкий та ефективний
Sublime Text	Комерційна, пробний період	Велика кількість плагінів	Швидкий та ефективний
Web Storm	Комерційна, пробний період	Вбудовані функції	Менш швидкий, висока продуктивність

Було обрано Visual Studio Code як безкоштовний потужний та ефективний редактор із великою кількістю розширень.

Висновки до розділу 2

Структура системи, яку ми розробляємо, відповідає обраним мовам та інструментам, таким як HTML, CSS, JS, Node.js, фреймворк React та Express, і середовищу розробки VS Code. Основною мовою для створення веб-застосунку є HTML, яка використовується для створення структури та розмітки веб-сторінок. CSS відповідає за стилізацію та вигляд веб-сторінок, а JavaScript - за динамічність та інтерактивність.

Node.js виступає як серверна технологія, яка дозволяє виконувати JavaScript на стороні сервера. Використання Node.js дозволяє створювати швидкі та масштабовані веб-застосунки. Фреймворк React є потужним інструментом для розробки інтерфейсів користувача, який пропонує компонентний підхід до побудови UI.

Фреймворк Express надає зручний і простий спосіб розробки серверних додатків на Node.js. Він допомагає визначити маршрутизацію, обробку запитів

та роботу з базою даних. Використання Express дозволяє зменшити час розробки та забезпечити ефективну обробку запитів.

База даних PostgreSQL пропонує розширені можливості для збереження та управління даними. База даних швидка та надійна і підійде до застосунку.

Середовище розробки VS Code є потужним та розширюваним інструментом для написання коду. Воно підтримує багато мов програмування, має вбудовану підтримку Git, автодоповнення та розширення, що полегшує розробку веб-застосунків.

Враховуючи обрані мови та інструменти, структура системи буде базуватися на клієнт-серверній архітектурі, де HTML, CSS та JavaScript використовуватимуться на стороні клієнта, а Node.js та Express будуть відповідальні за обробку запитів на стороні сервера. Фреймворк React надасть зручні засоби для побудови користувацького інтерфейсу.

В цілому, обрані мови, фреймворки та середовище розробки створюють потужний набір інструментів для розробки веб-застосунків, забезпечуючи ефективність, швидкість та зручність розробки.

3 РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Змістовна постановка задачі

Задача веб-застосунку – полегшити життя користувачу, створивши онлайн-платформу для продажу кондитерських виробів. Головна мета проєкту – надати зручний та привабливий спосіб покупки та замовлення кондитерських виробів для користувачів. Функціональні вимоги:

— Реєстрація користувача: Користувачі зможуть створювати облікові записи за допомогою електронної пошти та пароля, забезпечення можливості авторизації користувачів на сайті;

— Каталог продуктів: розмістити інтерактивний каталог кондитерських виробів з фотографіями, назвами та описами. Користувачі зможуть переглядати продукти та шукати їх за категоріями, ціною, назвою тощо.

— Корзина покупок: додати можливість додавання продуктів до корзини покупок. Користувачі зможуть змінювати кількість продуктів у корзині, видаляти їх або оновлювати інформацію про продукти;

— Оформлення замовлення: розробити процес оформлення замовлення, включаючи введення адреси доставки, обрання способу оплати та підтвердження замовлення.

3.2 Математична постановка задачі

Математична постановка задачі допомагає формалізувати вимоги та процеси, які потрібно реалізувати в системі, та забезпечує базу для подальшої розробки та оптимізації веб-застосунку.

— Формалізація вхідних даних: визначити множини даних, необхідних для функціонування системи, такі як список продуктів, їх вартість,

категорії тощо, описати параметри користувачів, такі як ім'я, електронна пошта, адреса доставки;

— Алгоритми обробки замовлень: розробити алгоритми для обробки замовлень, включаючи перевірку наявності товару на складі, розрахунок загальної вартості замовлення, обробку платежів та формування звіту про замовлення.

3.3 Існуючі методи та алгоритми

В цілому, у розробці веб-застосунків немає великої кількості готових методів та алгоритмів для вирішення поставленого завдання. Проте деякі все ж існують та націлені на оптимізацію роботи застосунку.

3.3.1 Алгоритми пошуку

Алгоритми пошуку є комп'ютерними алгоритмами, які призначені для пошуку певних даних або розв'язання задачі шляхом обходу, перевірки і аналізу різних елементів даних. Основна мета алгоритмів пошуку - знайти необхідну інформацію або знайти розв'язок задачі, перебираючи можливі варіанти або використовуючи інші ефективні стратегії. Вони можуть оперувати різними типами даних, такими як текст, числа, графи або інші структури даних. Деякі популярні алгоритми пошуку включають:

— Лінійний пошук: Цей алгоритм просто перебирає елементи один за одним у послідовності і порівнює їх зі шуканим значенням. Якщо збіг знаходиться, повертається позиція елемента. Лінійний пошук простий у реалізації, але має часову складність O(n), де n - кількість елементів у послідовності.

— Бінарний пошук: Цей алгоритм припускає, що послідовність впорядкована, і шукає значення шляхом ділення її навпіл і порівняння

середнього елемента зі шуканим значенням. Якщо середній елемент співпадає зі шуканим, пошук завершується. В іншому випадку алгоритм продовжує пошук в лівій або правій половині послідовності. Бінарний пошук має часову складність O(log n), що робить його ефективним для впорядкованих даних.

3.3.2 Алгоритми сортування

Сортування є важливим аспектом обробки даних у застосунках. Вибір конкретного алгоритму залежить від типу даних, обсягу даних та вимог до продуктивності.

— Сортування злиттям: Цей алгоритм розділяє послідовність на дві половини, сортує їх окремо, а потім злитає впорядковані підпослідовності у впорядковану послідовність. Його часова складність становить O(n log n).

— Швидке сортування: Цей алгоритм розбиває послідовність на менші підпослідовності відносно опорного елемента, сортує їх окремо і об'єднує впорядковану послідовність. Його часова складність також становить O(n log n).

— Сортування вставками: Цей алгоритм проходить по послідовності зліва направо, вставляючи кожен елемент у вже впорядковану частину послідовності. Його часова складність становить $O(n^2)$, але він ефективний для малих наборів даних.

3.3.3 Алгоритми оптимізації

Оптимізація алгоритмів є важливою задачею у веб-розробці, яка ставить за мету покращити продуктивність та ефективність виконання програмного коду.

— Мемоізація: техніка, яка полягає в збереженні результатів обчислень для пізніших викликів з тими ж вхідними даними. Це дозволяє

уникнути повторного обчислення результатів та значно покращує продуктивність. Мемоізація широко використовується у функціональних мовах програмування, де часто використовуються рекурсивні функції.

— “Розділення та панування”: Цей алгоритмічний підхід полягає в розбитті складного завдання на менші підзавдання, які вирішуються окремо, а потім комбінуються для отримання кінцевого результату. Це дозволяє зменшити складність задачі та покращити час виконання. Прикладом є сортування злиттям (Merge Sort), де масив рекурсивно розбивається на менші підмасиви, які потім зливаються відсортованими.

— Оптимізація пам'яті: Крім оптимізації часу виконання, також важливо ефективно використовувати пам'ять. Це може включати уникнення непотрібних копіювань даних, використання мінімальної кількості змінних та оптимізацію використання кешу.

3.3.4 Алгоритми рекомендацій

Алгоритми рекомендацій є важливою складовою багатьох веб-застосунків, зокрема платформ електронної комерції, стрімінгових сервісів, соціальних мереж і багатьох інших. Метою цих алгоритмів є аналіз даних про користувачів і надання рекомендацій щодо продуктів, контенту або послуг, які можуть бути їм цікавими. Вибір конкретного алгоритму рекомендацій залежить від багатьох факторів, таких як тип даних, обсяг даних, ступінь персоналізації, доступні ресурси та інші.

— Фільтрування за змістом (“Content-based filtering”): цей алгоритм рекомендує користувачеві елементи, які мають схожий зміст до тих, які йому сподобалися раніше. Він аналізує властивості або характеристики елементів, такі як ключові слова, категорії або теги, і порівнює їх з вподобаннями користувача. Наприклад, якщо користувачу сподобалися певні фільми жахів, система може рекомендувати йому інші фільми з жанру жахів.

— Фільтрування на основі співпадіння користувачів (“Collaborative filtering”): алгоритм рекомендує користувачеві елементи на основі спільних вподобань з іншими користувачами. Він аналізує дані про вподобання користувачів та встановлює зв'язки між ними. Наприклад, якщо два користувачі мають схожі вподобання щодо фільмів, система може рекомендувати одному користувачеві фільми, які сподобалися іншому користувачеві з подібними вподобаннями.

3.3.5 Алгоритми валідації та перевірки даних

Алгоритми валідації та перевірки даних є важливим етапом розробки веб-застосунків, оскільки вони допомагають забезпечити правильність та цілісність вхідних даних. Вони виконують перевірку на відповідність заданим правилам та форматам, на наявність потенційних помилок або некоректних значень, а також допомагають уникнути помилок, забезпечити безпеку та зручну роботу з даними.

— Перевірка на наявність: Цей алгоритм перевіряє, чи є вхідні дані присутніми і не порожніми. Він перевіряє, чи заповнені обов'язкові поля форми або чи передані необхідні параметри.

— Перевірка на тип даних: Цей алгоритм перевіряє, чи відповідають вхідні дані заданому типу даних. Наприклад, він може перевіряти, чи є введене значення числом, рядком або датою.

— Перевірка на формат: Цей алгоритм перевіряє, чи відповідають вхідні дані заданому формату. Наприклад, він може перевіряти, чи введений електронний адрес має правильну структуру або чи номер телефону має заданий шаблон.

— Перевірка на довжину: Цей алгоритм перевіряє, чи відповідає довжина введених даних заданим обмеженням. Наприклад, він може

перевіряти, чи не перевищує введений текст максимальну допустиму кількість символів.

— Перевірка на унікальність: Цей алгоритм перевіряє, чи є вхідні дані унікальними та не дублюються в базі даних. Він може перевіряти, чи не зареєстрований користувач з таким самим ім'ям або електронною адресою.

— Перевірка на безпеку: Цей алгоритм перевіряє вхідні дані на наявність потенційно небезпечних або шкідливих символів, які можуть призвести до атак на безпеку. Він виконує фільтрацію та екранування вхідних даних для запобігання вразливостям, таким як “ін’єкції” SQL-запитів або скриптового коду.

3.4 Обґрунтування обраних методів розв’язання

Серед алгоритмів, які було вирішено використовувати, є лінійний пошук для роботи зі списками. Лінійний пошук простий у реалізації та має лінійну складність, що робить його ефективним для невеликих обсягів даних або одиночних пошукових операцій.

Швидке сортування – поширений алгоритм у застосунках такого типу і є влаштованим у мову програмування JavaScript. Алгоритм швидкий та ефективний для великих обсягів даних, що робить його ідеальним для невеликого проєкту.

Використання мемоізації у проєкті дозволяє зберігати результати попередніх обчислень та уникати повторних обчислень або запитів. Це покращить продуктивність та швидкість виконання, зменшить навантаження на базу даних та забезпечить швидку відповідь на запити. Найчастіше використовується у зверненнях до бази даних, адже постійно шукати користувачів для кожного запиту довго та неефективно.

Алгоритм “Розділення та панування” дозволяє краще структурувати проєкт, розбивши його на невеликі блоки методів та функцій. Цей підхід

дозволяє збільшити ефективність обробки великих обсягів даних та забезпечити більшу масштабованість.

Було обрано алгоритм фільтрування за змістом, оскільки буде необхідно проводити аналіз та класифікацію даних на основі їх змісту. Цей алгоритми дозволяє визначати та виділяти ключові елементи або виконувати категоризацію даних залежно від їх вмісту.

3.5 Функції застосунку

Нижче наведено таблицю 3.1 з описом функцій застосунку.

Таблиця 3.1 – Специфікації функцій застосунку

№	Функція	Опис
1	createSession	Створює нову сесію для користувача
2	addSubscription	Додає підписку на електронну пошту користувача
3	contactForm	Додає контактну форму користувача
4	addDeliveryAddress	Додає адресу доставки користувача
5	addContactData	Додає новий контакт користувача
6	addToCart	Додає товар до кошику користувача
7	removeFromCart	Видаляє товар з кошику користувача
8	getAllCartItems	Повертає список всіх товарів у корзині користувача
9	proceedOrder	Виконує замовлення товарів у корзині користувача
10	getAll	Повертає список доступних міст/продуктів/інгредієнтів
11	getOne	Повертає місто/продукт/інгредієнт по параметру
12	search	Шукає доступне місто/продукт/інгредієнт по параметру
13	getStockOfProduct	Повертає доступні продукти у пекарні
14	decreaseQuantity	Зменшує кількість товарів у пекарні після замовлень
15	getIngredientsForOneProduct	Повертає список інгредієнтів для обраного продукту

№	Функція	Опис
16	getAllByType	Повертає продукт по заданому типу
17	getRandomFour	Обирає 4 випадкові продукти з БД
18	errorHandler	Відповідає за відладку та повернення помилок
19	register	Реєструє користувача
20	validateEmailAddress	Валідує користувача, якщо той підтвердив пошту
21	login	Авторизує користувача по логіну та паролю
22	changePassword	Змінює пароль користувача
23	changeEmail	Змінює пошту користувача
24	changeTelNum	Змінює телефон користувача
25	addDeliveryAddress	Додає адресу доставки користувача
26	getAllDeliveryAddresses	Повертає всі адреси доставки користувача
27	getTheLatestDeliveryAddress	Повертає останню адресу доставки користувача
28	changeDeliveryAddress	Змінює адресу доставки користувача
29	deleteDeliveryAddress	Видаляє адресу доставки користувача
30	refreshSession	Оновлює сесію користувача
31	logout	Виходить з акаунту користувача
32	addToCart	Додає товар до кошику користувача
33	removeFromCart	Видаляє товар з кошику користувача
34	getAllCartItems	Повертає весь товар з корзини користувача
35	getAllOrders	Повертає всі замовлення користувача
36	getAllNonDeliveredOrders	Повертає недоставлені замовлення користувача
37	searchOrders	Шукає замовлення по id
38	generateJWT	Генерує JSON Web Token
39	verifyToken	Верифікує токен
40	wipeAllUserRefreshSessions	Видаляє з БД всі активні сесії користувача
41	verifySessionRefreshRequest	Підтверджує запит про оновлення сесії
42	generateAccessToken	Генерує токен доступу до сесії
43	generateRefreshToken	Генерує токен оновлення сесії
44	generateClientsAccessToken	Генерує токен доступу клієнта
45	verifyClientsToken	Верифікує токен клієнта

№	Функція	Опис
46	generateVerificationCode	Генерує код підтвердження для користувача
47	sendMail	Відправляє лист користувачу
48	isEmpty	Перевірка на пустий об'єкт
49	rateLimiterMiddleware	Захист від Brute Force
50	defineName	Визначає обрану мову сторінки
51	start	Запускає сервер

3.6 Діаграма компонентів

Діаграми компонентів є графічними представленнями компонентів системи та їх взаємозв'язків [24]. Вони надають візуальну ілюстрацію архітектури системи та допомагають зрозуміти, як компоненти взаємодіють між собою. Основною метою діаграм компонентів є виділення основних функціональних блоків системи та показ взаємозв'язків між ними.

Діаграму компонентів наведено у кресленику ІА92.200БАК.004 Д2.

3.7 Діаграма станів

Діаграма станів (State diagram) – це графічний інструмент моделювання, який використовується для візуалізації та опису різних станів, переходів між станами та подій, що сприяють управлінню поведінкою об'єкту або системи [25].

Діаграма станів дозволяє моделювати поведінку об'єкту, аналізувати його різні стани та деталізувати переходи між ними. Вона корисна для визначення логіки інтеракції між об'єктами або компонентами системи, а також для управління становими змінами відповідно до вхідних подій.

Діаграму станів наведено у кресленику ІА92.200БАК.004 Д3.

3.8 Діаграма потоків даних

Діаграма потоків даних (Data Flow Diagram, DFD) – це графічний інструмент, який використовується для моделювання та представлення потоку даних в системі. Вона відображає, як дані переходять через різні процеси, зберігаються в різних сховищах та обробляються різними компонентами системи [26].

Основною метою діаграми потоків даних є візуалізація та аналіз потоку даних в системі, що допомагає розуміти, як дані обробляються, як вони переміщаються через різні етапи та взаємодіють з різними компонентами системи. Діаграма потоків даних дозволяє визначити основні процеси, вхідні та вихідні дані, а також залежності між компонентами системи.

Діаграма потоків даних, ілюструє те, як дані обробляються системою на вхід та вихід, також вказує на те, звідки дані надходять, де і як зберігаються та обробляються.

Діаграму потоків даних наведено у кресленику ІА92.200БАК.004 Д4.

3.9 ER-діаграма

ER-діаграма (сутнісно-реляційна діаграма) – це графічний інструмент моделювання, який використовується для візуалізації структури бази даних і взаємозв'язків між різними сутностями (об'єктами) у цій базі даних. Вона названа так на честь двох основних компонентів: "сутність" – це об'єкт або елемент, який ми хочемо зберігати в базі даних, і "реляція" – це зв'язок або взаємозв'язок між сутностями.

ER-діаграми зазвичай використовуються в контексті проектування баз даних і допомагають при розробці і моделюванні бази даних з точки зору сутностей, атрибутів та взаємозв'язків між ними. Вони є потужним засобом

комунікації між розробниками програмного забезпечення, аналітиками та іншими учасниками процесу розробки системи.

Основна мета ER-діаграми – відображення важливих сутностей, атрибутів та взаємозв'язків між ними. Вона дозволяє чітко визначити структуру бази даних, відношення між таблицями, первинні та зовнішні ключі, атрибути сутностей та їх типи даних. Це допомагає зрозуміти взаємозв'язки між об'єктами та розробити ефективну схему бази даних, що відповідає вимогам системи.

ER-діаграму наведено у кресленику ІА92.200БАК.004 Д5.

Висновки до розділу 3

У даному розділі було виконано широкий спектр робіт щодо створення та розробки системи.

Була описана структура системи, включаючи компоненти, їх взаємозв'язки та взаємодію між ними. З використанням React та Express, система була розбита на незалежні компоненти, що дозволяє полегшити розробку та підтримку системи. NodeJS та Express були використані для побудови серверної частини системи, що дозволяє обробляти запити від клієнтів та забезпечувати відповідні дані.

Для ефективної роботи з фронтендом були використані HTML, CSS та JS. HTML був використаний для структурування веб-сторінок, CSS – для оформлення та стилізації, а JS – для реалізації інтерактивності та динамічних елементів на сторінках.

У процесі реалізації системи були використані різноманітні бібліотеки та інструменти. Вони були обрані з урахуванням їх потужних можливостей, надійності та підтримки, що сприяє ефективній та продуктивній розробці системи. Завдяки огляду сучасних методів та технологій, веб-застосунок був значно наближений до розробки. Результатом дослідження матеріалів стала

концепція функціональної, ефективної та зручної системи, яка готова забезпечити користувачів потрібними функціями та відповідати їх вимогам.

Одним із ключових виборів при розробці системи була використання бази даних PostgreSQL, що забезпечує надійне зберігання та обробку великих обсягів інформації. Також для забезпечення високої продуктивності веб-застосунку була використана бібліотека React.js, яка дозволяє ефективно керувати станом інтерфейсу та забезпечує швидке оновлення даних на стороні клієнта.

4 ТЕХНОЛОГІЧНИЙ РОЗДІЛ

4.1 Керівництво користувача

Для роботи із веб-застосунком, необхідно відкрити будь-яке середовище вихідного коду, наприклад, Visual Studio Code. Після цього відкрити командний термінал та встановити залежні модулі за допомогою команди “npm install”. Наступним кроком необхідно побудувати застосунок, виконавши команду “npm run build”. Тепер можна локально запустити фронт-енд частину застосунку за допомогою команди “npm start”. Далі, необхідно перейти за посиланням http://localhost:3000/, вставивши його у адресний рядок браузера.

Після цього, у вікні браузеру повинна відобразитись головна сторінка (рис. 4.1):

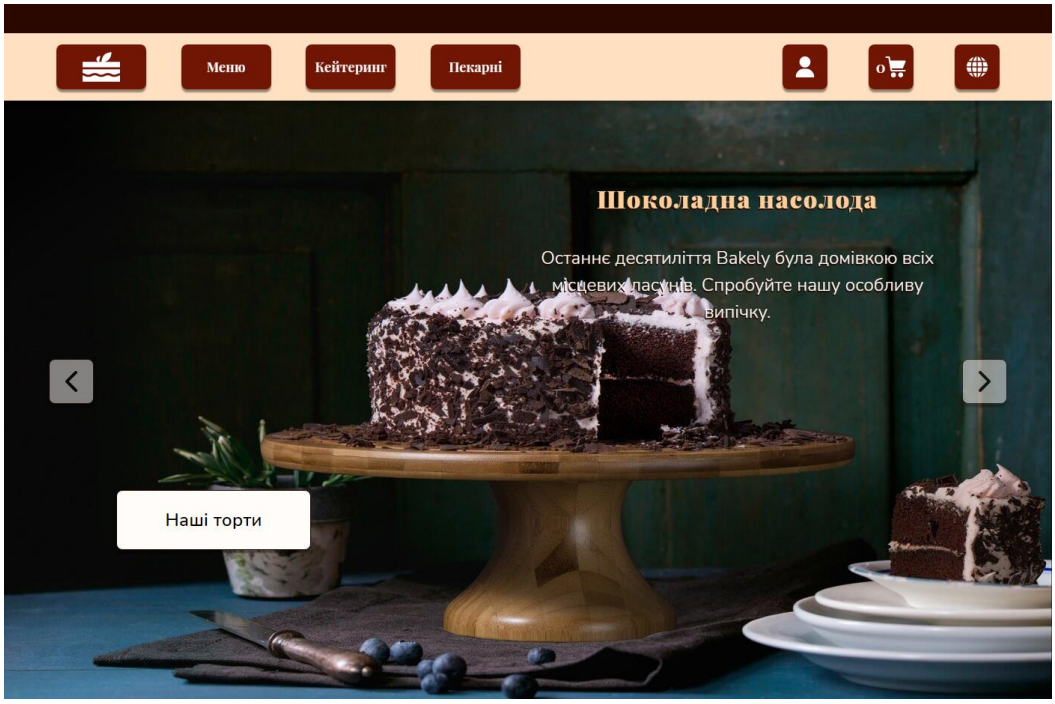

Рисунок 4.1 – Головна сторінка застосунку

У верхній частині сторінки можна бачити навігаційну панель. За допомогою неї виконується навігація сайтом – меню, кейтеринг, доступні пекарні, особистий кабінет, зміна мови, корзина.

Прогорнувши головну сторінку застосунку нижче, можна побачити іншу картинку із описом історії компанії. Навігаційна панель зверху не зникає, а залишається на екрані (рис 4.2):

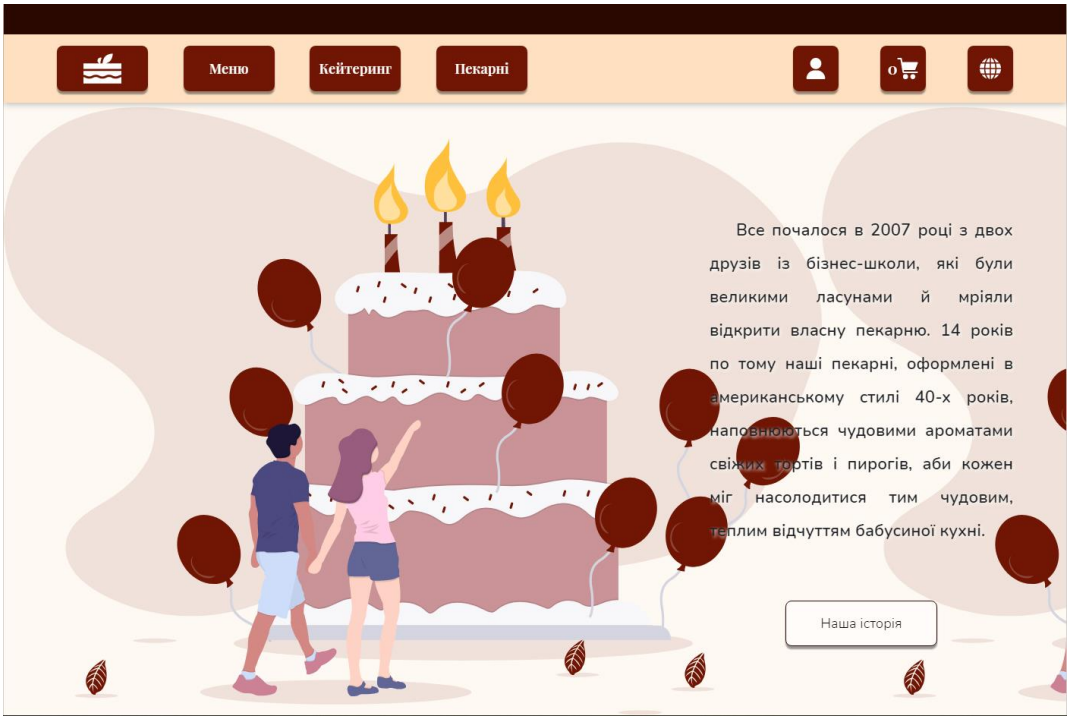

Рисунок 4.2 – Головна сторінка застосунку, прогорнута нижче

Сторінку можна прогорнути до кінця вниз, де розташований “footer” – нижня панель із посиланнями на корисні ресурси, наприклад, соціальні мережі (рис 4.3):

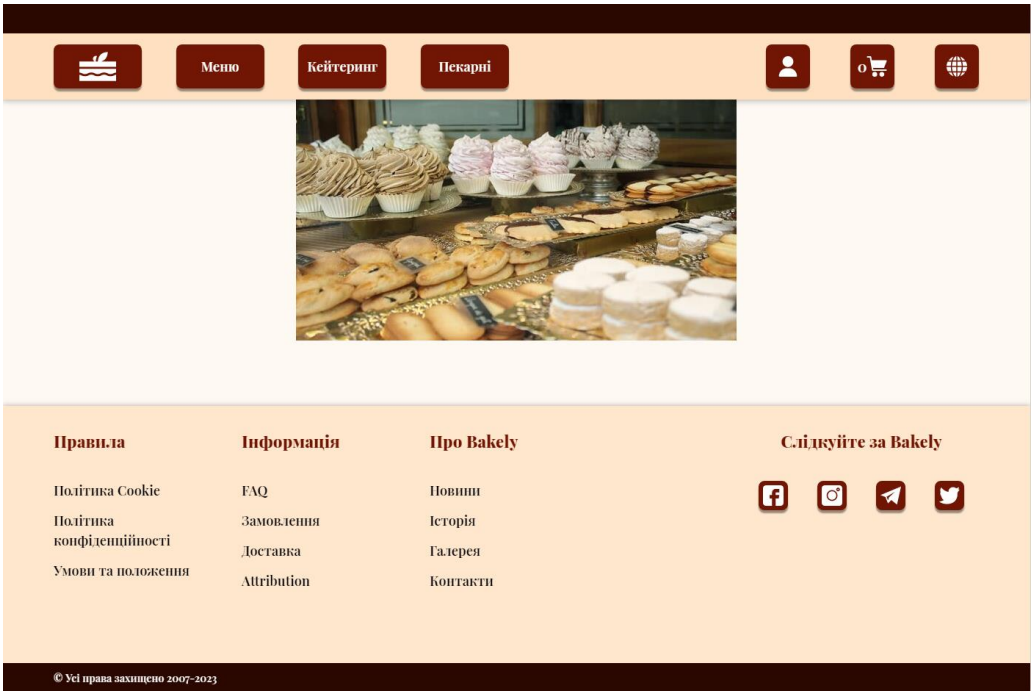

Рисунок 4.3 – Головна сторінка застосунку внизу

Після натискання кнопки “FAQ”, наприклад, користувач буде перенаправлений на сторінку із поширеними питаннями (рис. 4.4):

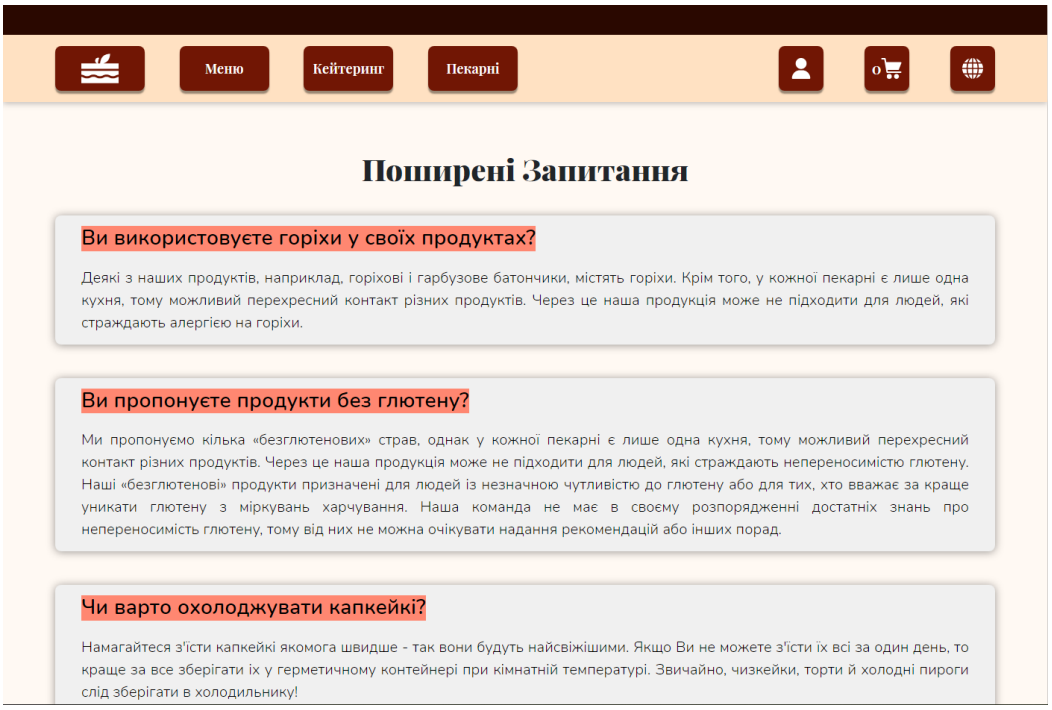

Рисунок 4.4 – Поширені запитання

Натискання кнопки у вигляді торту поверне до головної сторінки. Натискання кнопки “Меню” відобразить доступні товари (рис. 4.5):

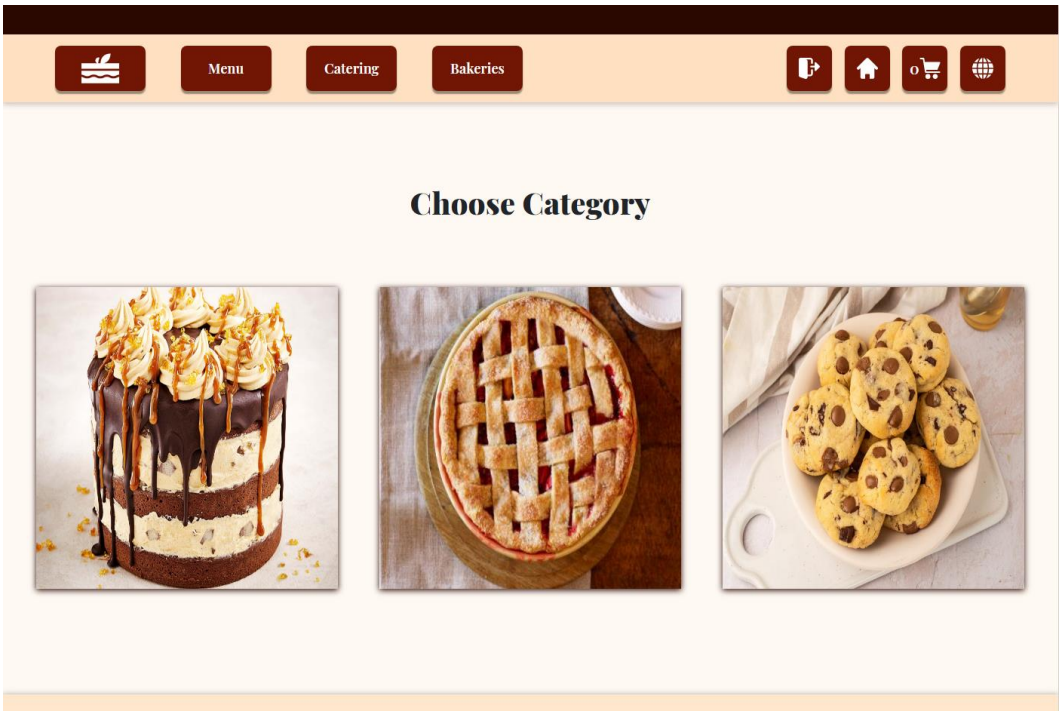

Рисунок 4.5 – Меню

Кнопка “Кейтеринг” відображає доступні кейтеринги – способи підприємництва, пов'язані із наданням послуг харчування на виїзді в таких місцях, як готель, паб, весілля тощо (рис. 4.6):

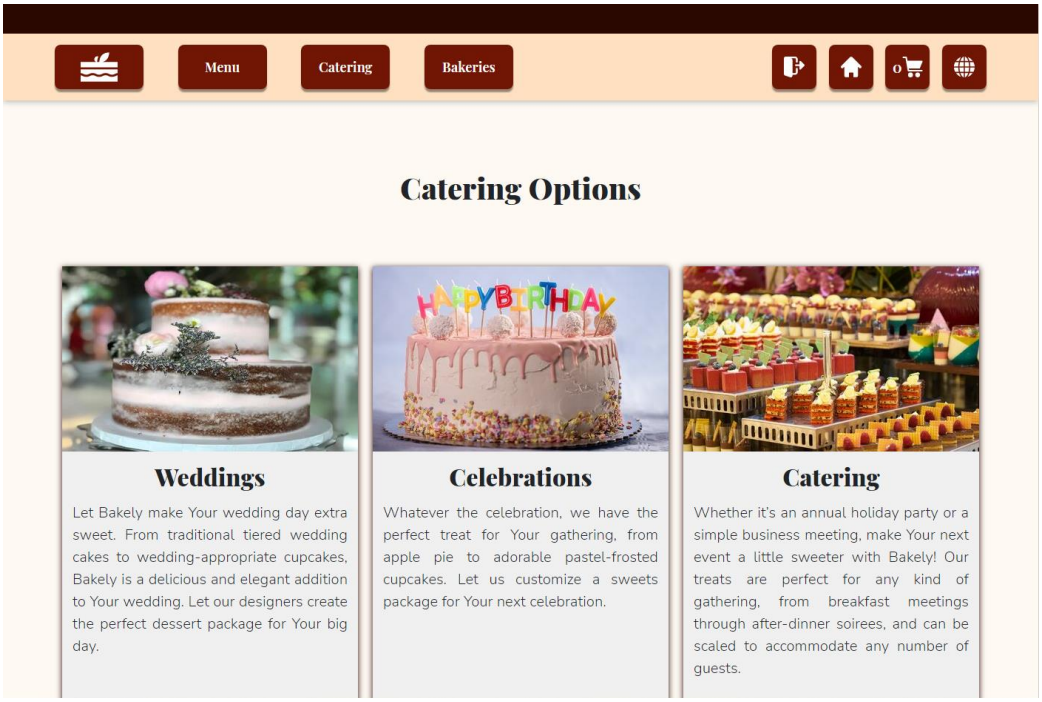

Рисунок 4.6 – Кейтеринг

Кнопка “Пекарні” відображає мапу, на якій можна подивитись доступні пекарні та де вони розташовані (рис. 4.7)

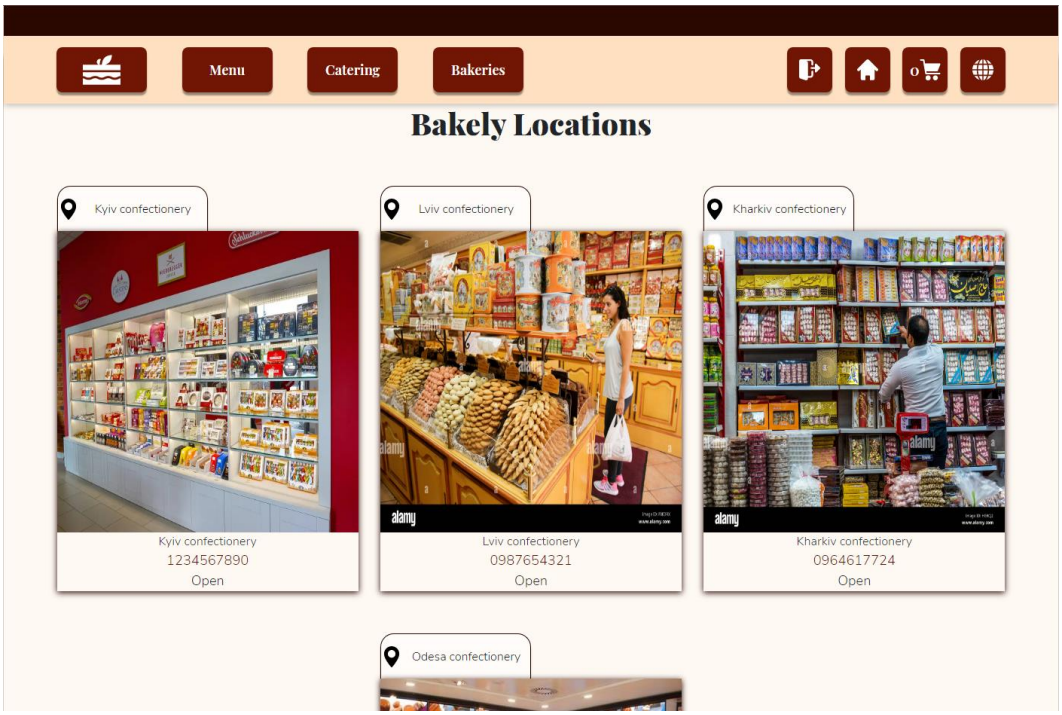

Рисунок 4.7 – Доступні пекарні

Кнопка із людиною відображає вікно входу/реєстрації до особистого кабінету (рис. 4.8):

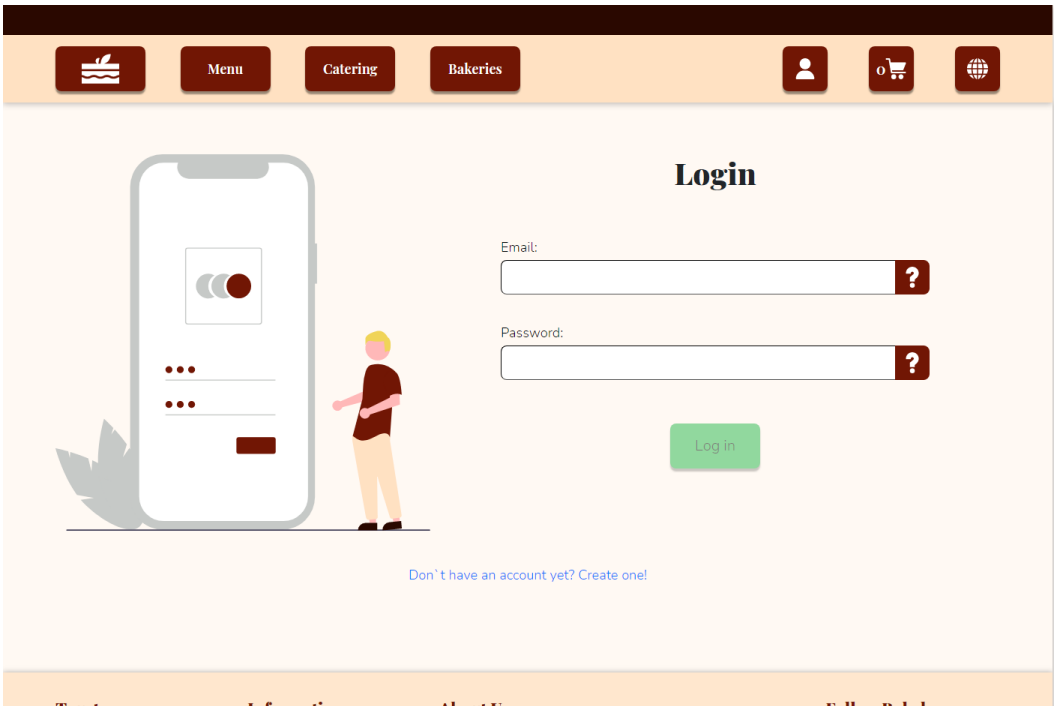

Рисунок 4.8 – Вікно входу/реєстрації

Кнопка із зображенням кошику відкриває кошик товарів користувача (рис. 4.9):

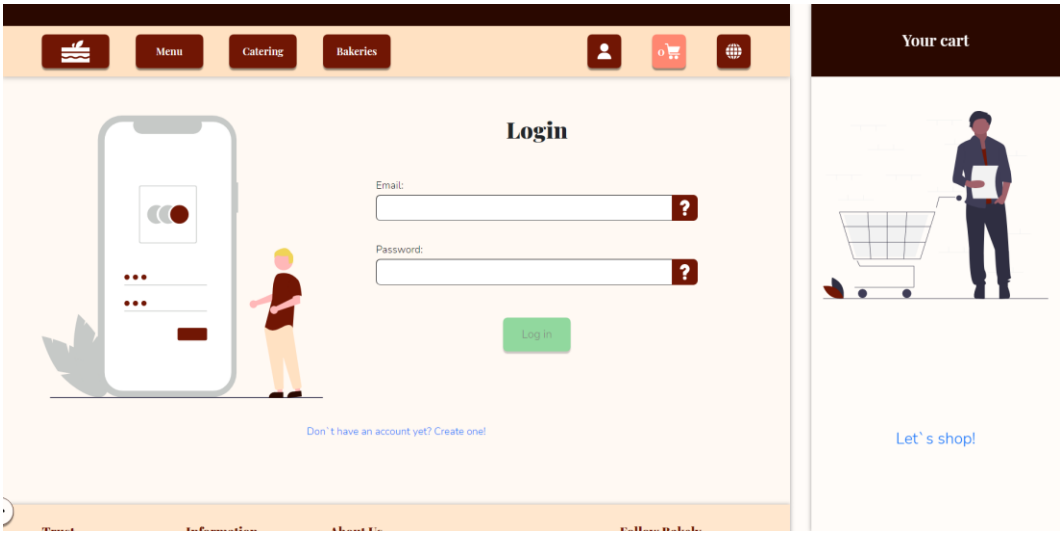

Рисунок 4.9 – Кошик товарів

4.2 Випробування програмного продукту

Випробування програмного продукту має на меті виявлення помилок, проблем та недоліків у функціональності та поведінці застосунку перед його розгортанням у середовище розробки. Це важливий крок для забезпечення якості та надійності продукту перед його використанням користувачами.

Загальний підхід до випробування програмного продукту полягає в тому, щоб покрити якомога більше сценаріїв використання, перевірити правильність роботи функцій та функціональності, виявити та виправити помилки перед розгортанням застосунку для кінцевих користувачів.

Під час випробування веб-застосунку на React, можна використовувати різні підходи та методи, такі як ручне тестування, автоматизоване тестування, тестування одиниць та функціональне тестування.

— Ручне тестування передбачає виконання тестових сценаріїв вручну з метою перевірки роботи окремих функцій та функціональності застосунку. Цей підхід дозволяє тестувати застосунок з реальної точки зору користувача та виявляти потенційні проблеми.

— Автоматизоване тестування використовує спеціальні інструменти та фреймворки для автоматизації тестових сценаріїв та перевірки очікуваного поведінки застосунку. Це дозволяє зменшити ручну працю та прискорити процес випробування.

— Тестування одиниць передбачає тестування окремих компонентів та функцій застосунку для перевірки їх правильності та працездатності. Цей підхід дозволяє виявляти помилки на ранніх етапах розробки та забезпечує легку підтримку та розширення кодової бази.

— Функціональне тестування зосереджується на перевірці функціональності та взаємодії різних компонентів та модулів веб-застосунку. Це допомагає виявити проблеми, пов'язані зі зміною стану, некоректною взаємодією та іншими аспектами функціональності.

У цьому розділі буде виконане ручне тестування як найпростіше з вище запропонованих.

В ході випробувань, проводилась перевірка всього функціоналу застосунку, результати записано до таблиць 4.1. – 4.4.

Таблиця 4.1 – Перевірка роботи користувацького інтерфейсу

Мета випробування	Перевірити функціонування користувацького інтерфейсу застосунку
Початковий стан	Відкрита головна сторінка застосунку
Випробування	Натиснути кнопки переходу до різних сторінок, зміни мови
Очікуваний результат	Перехід на сторінки веб-застосунку, коректна зміна мови застосунку
Стан моделі після випробування	Перехід на сторінки веб-застосунку, коректна зміна мови застосунку

Отже, з таблиці 4.1 можемо зробити висновок, що користувацький інтерфейс працює без помилок.

Таблиця 4.2 – Перевірка функції реєстрації

Мета випробування	Перевірити функціонал реєстрації на веб-сторінці
Початковий стан	Відкрита сторінка реєстрації

Випробування	Введення користувацьких даних, опційне підтвердження акаунту через лист на електронній пошті
Очікуваний результат	Успішна реєстрація та валідація акаунту через електронну пошту
Стан моделі після випробування	Успішна реєстрація та валідація акаунту через електронну пошту

Отже, з таблиці 4.2 можемо зробити висновок, що функціонал реєстрації працює коректно.

Таблиця 4.3 – Перевірка працездатності кошику товарів

Мета випробування	Перевірити працездатність кошику
Початковий стан	Відкрита сторінка магазину товарів
Випробування	Додавання товарів до корзини; їх видалення
Очікуваний результат	Товари успішно додаються та видаляються з корзини
Стан моделі після випробування	Товари успішно додаються та видаляються з корзини

Отже, з таблиці 4.3 можна зробити висновок, що кошик товарів працює правильно.

Таблиця 4.4 – Перевірка купівлі товару

Мета випробування	Перевірка роботи купівлі товару
Початковий стан	Відкритий кошик
Випробування	Перевірка системи купівлі товару

Очікуваний результат	Коректна робота застосунку щодо адреси користувача, суми грошей за товар у кошику
Стан моделі після випробування	Коректна робота застосунку щодо адреси користувача, суми грошей за товар у кошику

Отже, з таблиці 4.4 можна зробити висновок, що система купівлі товарів працює коректно.

Висновки до розділу 4

В даному розділі було наведено інструкцію для початку роботи та подальшого користування розроблюваною системою, детально описано кожну функцію та її виконання. Вся інструкція проілюстрована, для більшої зручності користувача. Також було проведено тестування застосунку. Весь інтерфейс, його кнопки та функціонал було перевірено на справність. Система готова до експлуатації та розширення функціоналу.

ВИСНОВКИ

В пояснювальній записці було розглянуто найпопулярніші веб-застосунки для продажу кондитерських виробів, їх типи та алгоритми, що в них застосовуються. Було досліджено різні методи реалізації застосунку, його проектування та розгортання.

В першому розділі було розглянуто веб-застосунки, різні методи та системи їх реалізації, було проведено порівняльний аналіз. Це дозволило визначити найпопулярніші та найефективніші підходи.

В другому розділі було розглянуто способи реалізації, обґрунтовано вибір мови, фреймворків та бібліотек, середовища розробки.

В третьому розділі було постановлено задачі розробки, розглянуто методи та алгоритми роботи з веб-застосунками, розроблено діаграми майбутньої системи.

В технологічному розділі наведено інструкцію користувача та проведено ряд тестувань. Було перевірено елементи веб-інтерфейсу.

Розробка, виконана в даному дипломному проєкті, є повноцінним працюючим веб-застосунком. У ній є клієнтський інтерфейс, серверна частина, під’єднана база даних.

Перевагою розроблюваної системи є її гнучка архітектура, що в подальшому дозволятиме розширювати систему без суттєвих змін у існуючих модулях. Крім того, система відрізняється швидкодією та привітним інтерфейсом, що сприяє комфортному користуванню. Загалом, розроблений веб-застосунок є повноцінним та ефективним інструментом для реалізації кондитерських виробів.

Цей проєкт демонструє високу якість розробки, враховуючи різноманітні аспекти, від архітектури до інтерфейсу користувача. Розроблений застосунок відповідає сучасним вимогам та забезпечує зручну та ефективну роботу.

ПЕРЕЛІК ПОСИЛАНЬ

1. A Step-by-Step Guide to Designing a Web Application [Електронний ресурс] – https://ugem.design/blog/web-app-design-process
2. Jon Yablonski Laws of UX: Using Psychology to Design Better Products & Services. O'Reilly Media, 2020 – 150 ст.
3. Інтернет-магазин Cakeshop [Електронний ресурс] – Режим доступу до ресурсу: https://cakeshop.com.ua/
4. Інтернет-магазин Beze [Електронний ресурс] – Режим доступу до ресурсу: https://beze.com.ua/
5. Інтернет-магазин Dolce Confections [Електронний ресурс] – Режим доступу до ресурсу: https://www.dolceconfections.com/
6. Стаття “Рекомендательные системы сегодня – необходимость для бизнеса” [Електронний ресурс] – Режим доступу до ресурсу: https://blog.heyml.com/рекомендательные-системы-сегодня-необходимость-для-бизнеса-41c1c9109988
7. 5 Most Popular Programming Languages for Web App Development [Електронний ресурс] – Режим доступу до ресурсу: https://blog.devgenius.io/5-most-popular-programming-languages-for-web-app-development-fa5912e2a23b
8. John Duckett: HTML & CSS Design and Build Web Sites. John Wiley & Sons, 2011 – 490 ст.
9. David DuRocher HTML and CSS QuickStart Guide: The Simplified Beginners Guide to Developing a Strong Coding Foundation, Building Responsive Websites, and Mastering Web Design. ClydeBank Media LLC, 2021 – 352 ст.
10. Marijn Haverbeke Eloquent JavaScript 3rd edition. No Starch Press, 2018 – 472 ст.

11. What is PHP? [Електронний ресурс] – Режим доступу до ресурсу: https://www.php.net/manual/en/intro-whatis.php
12. Node.js official web site [Електронний ресурс] – Режим доступу до ресурсу: https://nodejs.org/en
13. 10 Best JavaScript Frameworks to Use in 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://hackr.io/blog/best-javascript-frameworks
14. ReactJS official web site [Електронний ресурс] – Режим доступу до ресурсу: https://react.dev/
15. Angular official web site [Електронний ресурс] – Режим доступу до ресурсу: https://angular.io/
16. VueJS official web site [Електронний ресурс] – Режим доступу до ресурсу: https://vuejs.org/
17. ExpressJS official web site [Електронний ресурс] – Режим доступу до ресурсу: http://expressjs.com/
18. The 40 Best JavaScript Libraries and Frameworks for 2023 [Електронний ресурс] – Режим доступу до ресурсу: https://kinsta.com/blog/javascript-libraries/
19. What Is a Database? [Електронний ресурс] – Режим доступу до ресурсу: https://www.oracle.com/database/what-is-database/
20. Redis [Електронний ресурс] – Режим доступу до ресурсу: https://aws.amazon.com/redis/
21. Visual Studio Code official web-site [Електронний ресурс] – Режим доступу до ресурсу: https://code.visualstudio.com/
22. Sublime Text official web-site [Електронний ресурс] – Режим доступу до ресурсу: https://www.sublimetext.com/
23. Web Storm official web-site [Електронний ресурс] – Режим доступу до ресурсу: https://www.jetbrains.com/webstorm/

24. What is Component Diagram? [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/
25. All You Need to Know about State Diagrams [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/about-state-diagrams/
26. What is a Data Flow Diagram [Електронний ресурс] – Режим доступу до ресурсу: https://www.lucidchart.com/pages/data-flow-diagram

ДОДАТОК А

Посилання на репозиторій проєкту з програмним кодом: https://github.com/Suspensium/bakely