Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для продажу виготовлених на замовлення кондитерських виробів. Front-end
(тема)

Виконав:
студент 4 курсу, групи ПЗПІ–20–6

Кривушин Ю. О.
(прізвище, ініціали)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо–професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Кравець Н. С.
(посада, прізвище, ініціали)

Допускається до захисту
Зав. кафедри _____ З.В.Дудар
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо–професійна
Освітня програма Програмна інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2024 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Кривушину Юрію Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмна система для продажу виготовлених на замовлення кондитерських виробів. Front-end
Затверджена наказом по університету від 20.05. 2024р. № 471 Ст
2. Термін подання студентом роботи до екзаменаційної комісії 10.06.2024
3. Вихідні дані до роботи Розробити веб застосунок для замовлення кондитерських виробів, який надасть можливість замовляти як і готові продукти, так і створювати власні, мова програмування Type Script за допомогою бібліотеки React.
4. Перелік питань, що потрібно опрацювати в роботі
Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проектування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки, додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	08.04.2024	виконано
2	Створення специфікації ПЗ	10.04.2024	виконано
3	Проектування ПЗ	12.04.2024	виконано
4	Розробка ПЗ	14.05.2024	виконано
5	Тестування ПЗ	25.05.2024	виконано
6	Оформлення пояснювальної записки	01.06.2024	виконано
7	Підготовка презентації та доповіді	05.06.2024	виконано
8	Попередній захист	07.06.2024	виконано
9	Нормоконтроль, рецензування	10.06.2024	виконано
10	Здача роботи у електронний архів	15.06.2024	виконано
11	Допуск до захисту у зав. кафедри	17.06.2024	виконано

Дата видачі завдання 8 квітня 2024р.

Студент (ка)_____ Кривушин Ю. О.
(підпис)

Керівник роботи _____ доц. кафедри ПІ Кравець Н. С.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ / ABSTRACT

Пояснювальна записка до кваліфікаційної роботи бакалавра, 79 стор., 48 рис., 1 табл., 15 джерел.

ПРОГРАМНА СИСТЕМА ДЛЯ ПРОДАЖУ КОНДИТЕРСЬКИХ ВИРОБІВ ВИГОТОВЛЕНИХ НА ЗМОВЛЕННЯ, FRONTEND, REACT, TYPESCRIPT, VISUAL STUDIO CODE

Об’єкт розробки – програмна система для продажу кондитерських виробів виготовлених на замовлення FrontEnd.

Мета розробки – створення веб–застосунку, який дозволить користувачам замовляти кондитерські вироби та отримувати їх за замовленням у точно визначений час. Також повинна матись можливість створювати власні торти.

Метод рішення – середовище розробки Visual Studio Code, мови програмування TypeScript на фреймворк React, також будуть використані допоміжні технології, такі як ReduxToolkit для тимчасового зберігання даних, Scss препроцесор для Css стилів.

У результаті розробки створено веб–застосунок, який дозволяє користувачам замовляти кондитерські вироби, створювати власні за шаблоном, переглядати їх в каталозі, додавати в кошик та робити замовлення.

PROGRAMMING SYSTEM FOR SELLING CUSTOM–MADE CONFECTIONERY PRODUCTS, FRONTEND, REACT, TYPESCRIPT, VISUAL STUDIO CODE

The object of development is a software system for selling confectionery products made to order FrontEnd.

The purpose of the development is to create a web application that will allow users to order confectionery and receive it on order at a precisely defined time. It should also be possible to create your own cakes.

The solution method is the Visual Studio Code development environment, TypeScript programming language on the React framework, and auxiliary technologies such as ReduxToolkit for temporary data storage, Scss preprocessor for Css styles will also be used.

As a result of the development, a web application was created that allows users to order confectionery products, create their own using a template, view them in the catalog, add them to the cart, and place an order.

Я, Кривушин Юрій Олександрович, студент гр. ПЗПІ–20–6, здобувач вищої освіти на першому (бакалаврському) рівні кафедри «Програмна інженерія», заявляю: моя кваліфікаційна робота на тему «Програмна система для продажу кондитерських виробів виготовлених на замовлення. FrontEnd», що буде представлена до екзаменаційної комісії для публічного захисту, виконана самостійно, в ній не містяться елементи плагіату і вона може бути опублікована в електронному архіві відкритого доступу ElAr KhNURE. Усі запозичення з друкованих та електронних джерел мають відповідні посилання.

Я ознайомлений з діючим положенням «Про протидію академічному плагіату в ХНУРЕ», згідно з яким виявлення плагіату є підставою для відмови до допуску кваліфікаційної роботи до захисту та застосування дисциплінарних заходів.

ЗМІСТ

Вступ 7
1 Аналіз предметної галузі 8
1.1 Аналіз предметної галузі 8
1.2 Виявлення проблем та актуалізація рішень 12
1.3 Постановка задачі 13
1.3.1 Цільова аудиторія 14
2 Формування вимог до програмної системи 16
3 Архітектура та проєктуваня програмного забезпечення 19
3.1 UML проєктування ПЗ 19
3.2 Проєктування архітектури ПЗ 21
3.3 Проєктування структури зберігання даних 24
3.3 Розробка UI/UX інтерфейсу 25
4 Опис прийнятих програмних рішень 34
5 Тестування програмного забезпечення 48
5.1 Тестування застосунку 48
Висновки 51
Перелік джерел посилання 52

ВСТУП

Темою цієї кваліфікаційної роботи є програмна система для продажу кондитерських виробів, виготовлених на замовлення. Основне завдання цієї системи – надати користувачам можливість замовляти кондитерські вироби та отримувати їх за замовленням через зручний веб–інтерфейс з можливістю кастомізації.

Кінцевий програмний засіб повинне складати серйозну конкуренцію наявним конкурентам, бути зручним, враховуючи всі помилки та успіхі інших схожих продуктів.

Актуальність даної теми обумовлена рядом факторів. По–перше, в сучасному світі все більше людей віддають перевагу онлайн–покупкам, включаючи продукти харчування. Це зумовлено зручністю, швидкістю та можливістю вибору з широкого асортименту. Програмна система для продажу кондитерських виробів, виготовлених на замовлення, відповідає цим потребам, надаючи користувачам зручний інтерфейс для вибору та замовлення продуктів.

По–друге, враховуючи специфіку кондитерського бізнесу, де велику роль відіграє індивідуальний підхід до кожного клієнта, можливість кастомізації продуктів є важливим конкурентним перевагою. Наша система надає користувачам можливість створювати власні продукти, що відповідає цій потребі.

Таким чином, розробка програмної системи для продажу кондитерських виробів, виготовлених на замовлення, є актуальною та важливою задачею, яка відповідає сучасним тенденціям розвитку технологій та потребам ринку.

Метою роботи є розробка веб–застосунку, який дозволить користувачам замовляти кондитерські вироби, переглядати їх в каталозі, створювати власні продукти, додавати в кошик та робити замовлення. Для розробки продукту використовувалися технології React та TypeScript, Scss, Redux Toolkit а також середовище розробки Visual Studio Code.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Аналіз предметної галузі

За довгий час галузь по виготовленню кондитерських виробів на замовлення розвивалася тому враховуючи ще не дуже велику цільову аудиторію, досить важко складати конкуренцію в цій галузі, але також все буде залежати від міста в якому буде стартувати наше виробництво, адже в деяких містах ця конкуренція буде слабшою.

Давайте оцінимо галузь та її потреби з нашого боку, тобто інтерфейсу. Головною метою є забезпечення зручності. Інтерфейс повинен бути інтуїтивно зрозумілим і легким у використанні. Користувачі повинні мати змогу легко знайти те, що вони шукають, і замовити продукцію без зайвих кліків. Це означає, що меню навігації, каталог продуктів, кошик для покупок та процес оформлення замовлення повинні бути чітко організовані і легко доступні. Потрібно завчасно вивчити конкурентів[1], щоб підкреслити важливі моменти(див. рис. 1.1).

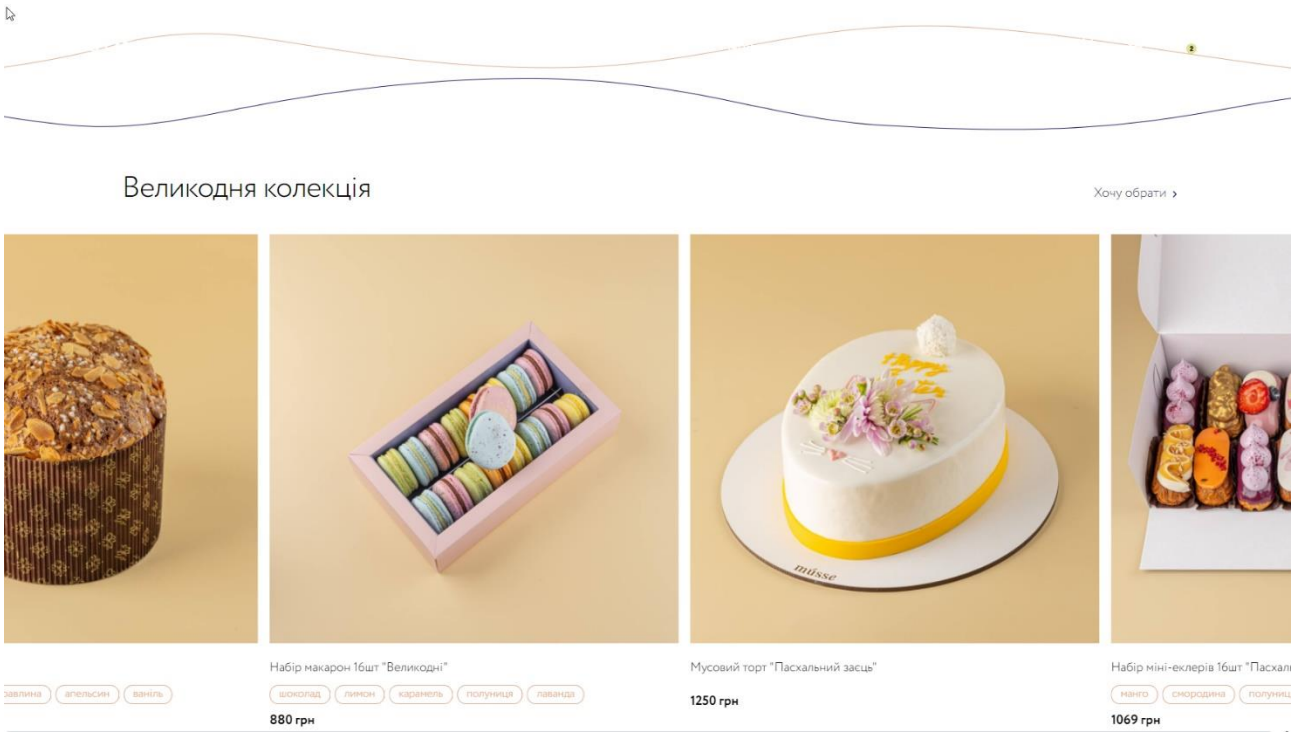

Рисунок 1.1 – Сайт кондитерських виробів musse (за даними [1])

Другою важливою ціллю є створення привабливого дизайну. Інтерфейс повинен бути візуально привабливим, щоб привернути увагу користувачів і створити позитивне враження про бренд. Це може включати в себе яскраві зображення продуктів, приємні колірні схеми та естетично приємні шрифти.

У цілому сайт musse(див. рис. 1.1) з точки зору привабливості дизайну є дуже добрим прикладом, він не є типовим для цієї галузі, тому запам’ятовується, але це несе свої ризики в зручності користування. Розглянемо ще одного конкурента[2](див. рис. 1.2).

Переваги musse:

– цікавий інтерфейс що запам’ятовується;
– користувач чітко розумію, що робить кожен елемент інтефейсу;
– хороша навігація по сайту;
– фото товару узгоджене з кольоровою гамою інтерфейсу;

Недоліки musse:

– схожості кольорів різних елементів інтферфейсу, із–за цього погано видно тексту чи кнопок у деяких місьцях сайту;
– перегруженість інтефейсу;
– майже відсутність фільтрів для товару;
– відсутність локалізації;

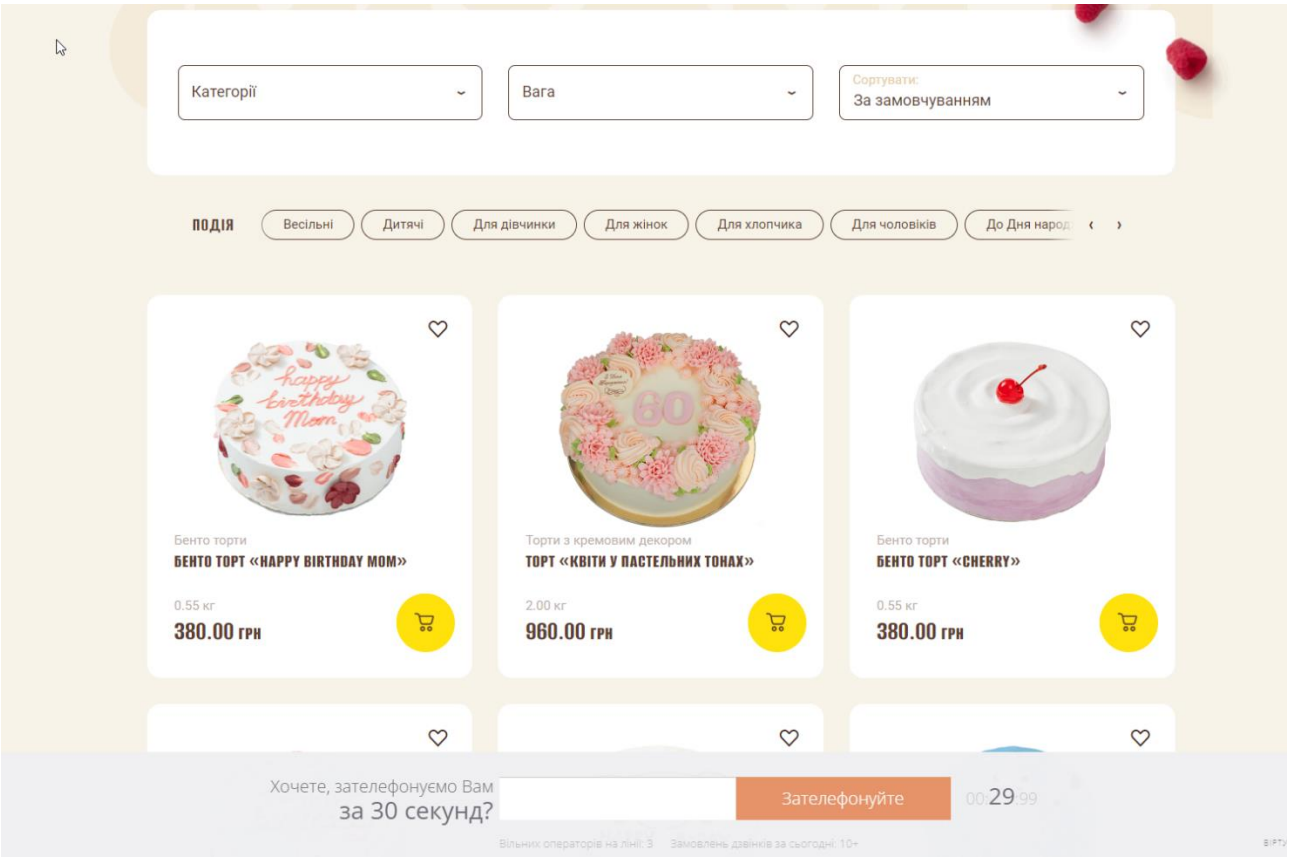

Рисунок 1.2 – Сайт кондитерських виробів Vatsak (за даними [2])

Вже на цьому прикладі конкурента ми бачимо більшу легкість інтерфейсу, але в цілому він не відрізняється від маси подібних сайтів та слабко запам’ятовується.

Все це разом сприяє створенню позитивного користувацького досвіду, що може привести до збільшення продажів і лояльності клієнтів. Завдяки цьому користувачі будуть повертатися до вашого магазину знову і знову.

Переваги Vatsak:

– простий та зрозумілий інтерфейс;

– наявність локалізації;
– хороша система фільтрів товару;

Недоліки Vatsak:

– не сучасний інтерфейс;
– однотонний вибір кольорів, із–за цього товар погано виділяється;
– при прокрутці нижче елементи плавно з’являються, це мінус оскільки тільки ускладнює використання сайту;

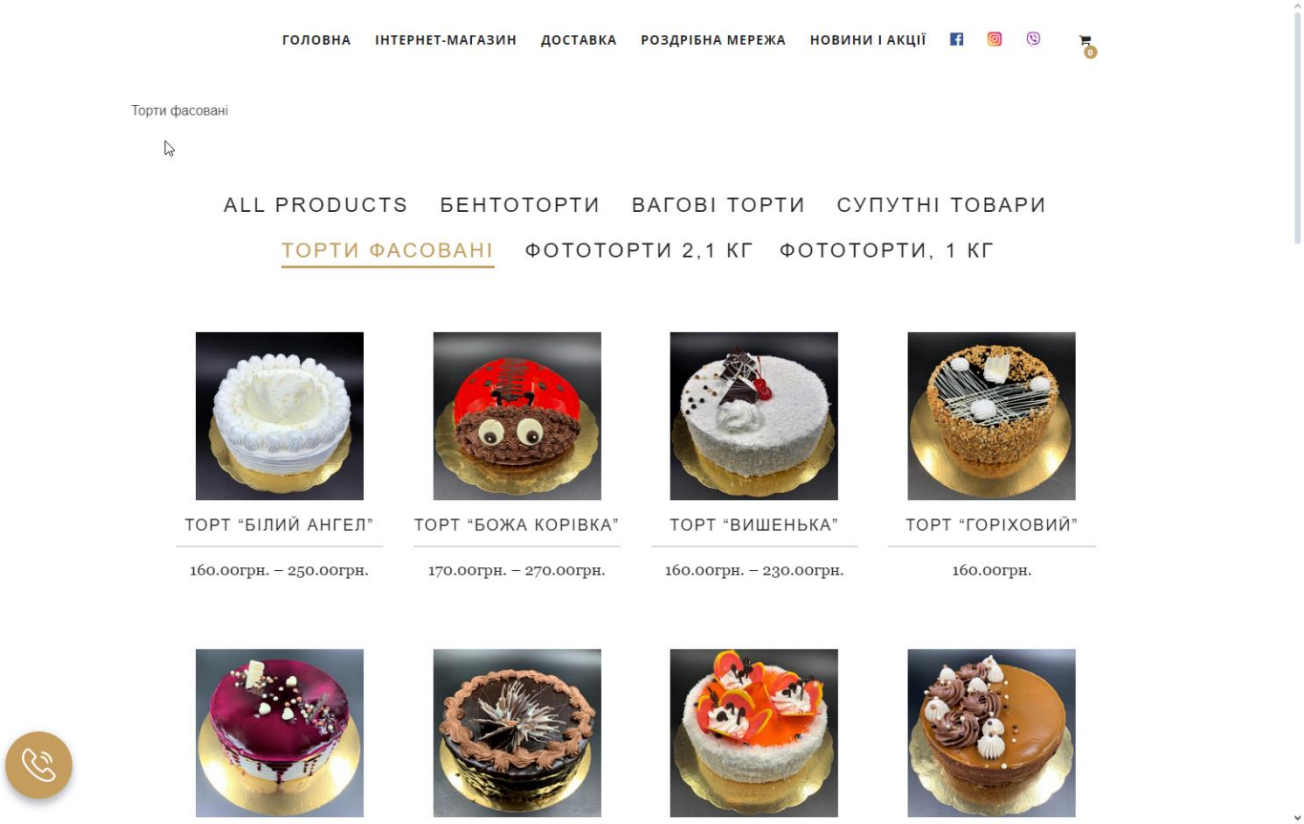

Рисунок 1.3 – Сайт кондитерських виробів «карамелька» (за даними [3])

Інтерфейс сайту на рис. 1.3 має чистий дизайн, з добре структурованою навігацією та акцентом на продукції. Основна частина сторінки містить сітку зображень тортів з назвами та цінами, що полегшує перегляд і вибір продукції.

Однак, є кілька проблем, які можуть погіршити користувацький досвід. Фільтри для категорій виглядають як звичайний текст і можуть бути неочевидними

для користувачів, тоді як додавання більш виразних кнопок могло б покращити їх помітність. Кожен товар має лише назву, ціну і зображення, що може бути недостатньо – було б корисно додати короткий опис або позначки про наявність і спеціальні пропозиції. Відсутність візуального виділення активних елементів категорій або фільтрів може призводити до плутанини, а невизначені діапазони цін, наприклад, 160.00 грн. – 250.00 грн. не пояснюють, що впливає на зміну ціни.

Загалом, дизайн сайту добре підходить для привертання уваги до продукції та забезпечує базову навігацію, але потребує деяких удосконалень для покращення користувацького досвіду.

Переваги «карамелька»:

– чистий сайт, без зайвих елементів;

– добре структурована навігація;

Недоліки «карамелька»:

– обмежена інформація про товари;

– відсутність візуального виділення активних елементів;

– не зрозумілий інтерфейс, такий як діапазони цін в інтерфейсі;

1.2 Виявлення проблем та актуалізація рішень

Однією з основних проблем, які ми можемо виявити в цій галузі, є висока конкуренція. Це означає, що нам потрібно знайти способи вирізнитися серед інших. Можливо, це може бути через унікальний дизайн продукту, винятковий сервіс або інноваційні технології.

Ще одна проблема – це змінність вимог та очікувань споживачів. Щоб залишатися актуальними, нам потрібно постійно слідкувати за трендами ринку та відгуками клієнтів.

Щодо актуалізації рішень, ми можемо розглянути впровадження гнучких методологій розробки, щоб швидко адаптуватися до змін. Також ми можемо

розглянути можливість створення сильного бренду, який буде асоціюватися з якістю та надійністю.

Проблемою буде забезпечити користувачам можливість користуватися вашою системою з будь-кого пристрою.

Отже головна проблема, яку треба вирішити, це баланс між впізнаваностю між конкурентами[1] та зручністю[2], потрібно враховувати людей різних вікових категорій.

1.3 Постановка задачі

Для вирішення проблеми, що присутні на ринку, та задля отримання аудиторії потрібно розробити застосунок, що буде враховувати вище зазначені проблеми. Потрібно створити застосунок з максимально функціональним інтерфейсом, що буде привабливий та запам’ятовуватись, але не бути перевантажений деталями.

Застосунок повинен бути простим у використанні, але при цьому має включати всі необхідні функції.

В про цесі виконання кваліфікаційної роботи треба виконати наступні завдання:

- розробка інтуїтивно зрозумілого інтерфейсу, який відповідає очікуванням користувачів;
- створення унікального дизайну, який відрізняє нас від конкурентів;
- впровадження системи зворотного зв’язку для постійного вдосконалення продукту;
- забезпечення високої якості продукту та сервісу для забезпечення лояльності клієнтів;
- навігація по застосунку з будь якої його частини;
- створення дизайну власного торту;

- зручне відображення списку товарів кожної категорії;
- можливість за шаблоном замовляти торти;
- реалізація кошику, оформлення замовлення та його оплату;
- систему реєстрації та входу в аккаунт;
- зміни особистих даних користувача для доставки замовлень;
- перегляд замовлень користувача;
- зміна мови застосунку;
- сторінку, де можна дізнатись відповідь на популярні питання;
- рекомендація товарів;
- секція найчастіших запитань;
- пошук товарів;

Перелік програмних засобів для реалізації програмного інтерфейсу:

- Addobe Photoshop
- Visual Studio Code
- Figma

Узагальнюючи – мета створити застосунок з багатофункціональним інтерфейсом, який буде привабливим, легко запам’ятовується, але не перевантажений деталями. Застосунок має бути простим у використанні, але при цьому включати всі необхідні функції, встановленні вище.

1.3.1 Цільова аудиторія

Знання цільової аудиторії є критично важливим для успіху магазину кондитерських виробів. Воно дає глибше розуміння потреби та бажання клієнтів, що в свою чергу допомагає створювати продукти, які вони хочуть купувати. Крім того, знання цільової аудиторії допомагає розробляти ефективні стратегії маркетингу та комунікації, які відповідають їхнім інтересам та стилю життя. Також, це допомагає побудувати сильніші відносини з клієнтами, оскільки ми можемо краще відповідати на їхні потреби та очікування.

Основною цільовою аудиторією магазину будуть люди:

- любителі солодкого;
- клієнти, які організовують події;
- фуд–ловери та гурмани, що шукають цікаві смаки;
- ділові люди;
- активні користувачі соціальних мереж, які можуть прийти через гарні фото кондитерських виробів;
- різної професії;

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

Нижче зробимо концепт–документ, що буде описувати програмну систему для продажу кондитерських виробів, виготовлених на замовлення. Система надає користувачам можливість замовляти кондитерські вироби та отримувати їх за замовленням через зручний веб–інтерфейс з можливістю кастомізації.

Існує 3 ролі користувачів, користувачі без аккаунту, з аккаунтом та адміни. Люди які не ввійшли в аккаунт можуть переглядати товари, додавати їх у кошик. Користувачі, що увійшли в аккаунт, можуть робити те саме, що й люди без нього, але також можуть оформлювати замовлення, переглядати власні замовлення, робити торт з власним дизайном. Адміни можуть робити все, що і інші, і займатись менеджерськими справами, такі як додавання, редагування та видалення товару, присвоювати звичайним користувачам роль адміну.

Метою роботи є розробка веб–застосунку, який дозволить користувачам замовляти кондитерські вироби, переглядати їх в каталозі, створювати власні продукти, додавати в кошик та робити замовлення. Для розробки продукту використовувалися технології React та TypeScript, Scss, Redux Toolkit а також середовище розробки Visual Studio Code.

Використовуючи React, ми можемо створити динамічні веб–інтерфейси для покращення користувацького досвіду[10]. TypeScript допомагає нам виявляти помилки на ранніх етапах розробки завдяки безпеці типів[1]. Scss полегшує розробку та підтримку стилів, роблячи CSS більш організованим та читабельним. Redux Toolkit спрощує управління станом додатку, що полегшує розробку та підтримку коду. Нарешті, Visual Studio Code надає потужні інструменти для розробки, що покращують продуктивність та якість коду. Всі ці технології разом дозволяють нам створити високоякісний, масштабований веб–додаток для вашого магазину кондитерських виробів.

Система повинна мати наступні функціональні вимоги:

– автентифікація та реєстрація: Користувачі повинні мати можливість зареєструватися та увійти в систему;

- перегляд каталогу: Користувачі повинні мати можливість переглядати каталог доступних кондитерських виробів;
- замовлення товарів: Користувачі повинні мати можливість замовляти кондитерські вироби;
- оплата: Користувачі повинні мати можливість оплатити свої замовлення;
- профіль користувача: Користувачі повинні мати можливість переглядати та редагувати свій профіль;
- конструктор тортів: Користувачі повинні мати можливість створювати власні торти, вибираючи різні параметри, такі як розмір, вага, склад тощо;

Система повинна відповідати наступним нефункціональним вимогам:

- чистий та зрозумілий UI/UX дизайн: Інтерфейс повинен бути інтуїтивно зрозумілим і легким у використанні;
- швидка обробка дій користувача: Система повинна реагувати на дії користувача (натискання кнопок, запити, зміни сторінок) протягом не більше 0.5 секунд;
- повне завантаження будь-якої сторінки повинно відбуватись не більше ніж за 2 секунди при швидкості інтернету не менше 10 Мбіт/с;
- захист інформації користувача: Система повинна захищати особисту інформацію користувачів;
- підтримка різних браузерів: Система повинна підтримувати останні дві версії наступних браузерів: Google Chrome, Mozilla Firefox, Safari, Microsoft Edge;
- локалізація на 2 мови, англійську і українську;
- перемикання мов: Користувач повинен мати можливість змінити мову інтерфейсу не більше ніж за 3 кліки;

Архітектура системи буде складатися з клієнтської частини (FrontEnd), розробленої на React, TypeScript, Scss, Redux Toolkit, та серверної частини (BackEnd), деталі якої будуть визначені під час подальшої розробки.

Мова TypeScript – це інструмент, який забезпечує більшу надійність та читабельність коду при розробці програмних продуктів. Однією з його ключових

переваг є можливість статичної типізації. Визначення типів даних для змінних, параметрів функцій і об'єктів допомагає виявляти помилки на етапі компіляції, що полегшує процес розробки та забезпечує більшу надійність програми.

Крім того, TypeScript покращує читабельність коду[13]. Явна декларація типів дозволяє розробникам краще розуміти функціонал програми та полегшує спільну роботу над проектом. Завдяки цьому, розробники можуть швидше зрозуміти, як працює код, і швидше вносити необхідні зміни[12].

У випадку розробки застосунків на React, використання TypeScript є особливо важливим. Оскільки React дозволяє створювати складні інтерфейси користувача, типізація допомагає уникнути помилок при взаємодії з компонентами та передачі даних між ними. Це покращує якість коду та допомагає уникнути потенційних помилок, що можуть виникнути під час розробки великих та складних проектів на React[10].

План розробки буде включати в себе різні етапи, такі як проектування, розробка, тестування та впровадження. Кожен етап буде мати визначені дати початку та завершення.

Потенційні ризики можуть включати затримки у розробці, проблеми з якістю продукту, недостатні ресурси тощо. Для кожного ризику буде розроблена стратегія подолання, що може включати додаткове планування, алокацію ресурсів, тестування тощо.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 UML проєктування ПЗ

Перед початком розробки програмної систем для продажу кондитерських виробів виготовлених на замовлення були визначені основні функції зі сторони клієнту. Після детального аналізу була створена Use–case діаграма для покупця[14](див. рис. 3.1).

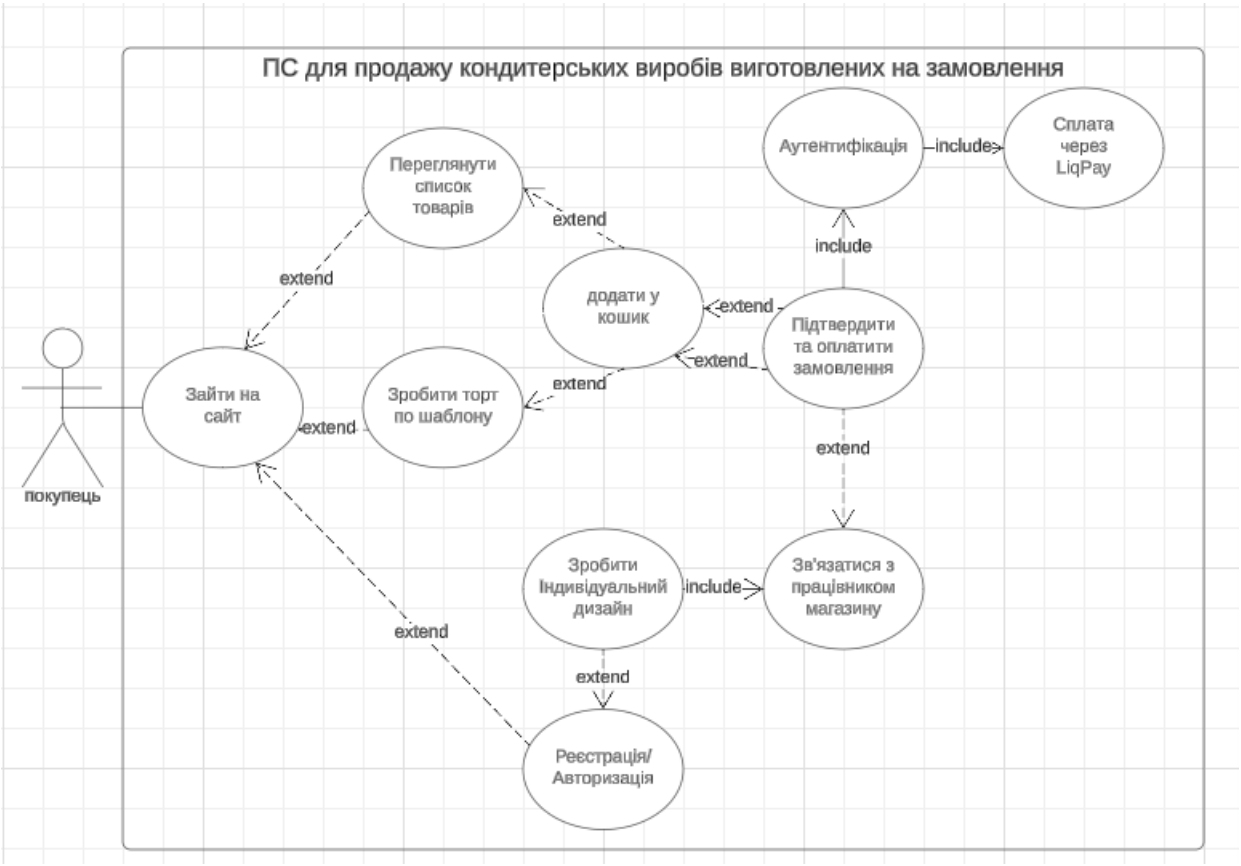

Рисунок 3.1 – Use–case діаграма застосунку для клієнту(рисунок виконаний самостійно)

Клієнт нашого застосунку повинен мати легкий доступ до будь якої частини сайту, мати можливість аутентифікації, використання кошику та створювання власних шаблонів торту. Він повинен мати можливість зареєструватись та увійти в аккаунт, змінювати власну інформацію, переглядати

та замовляти торти ти інші кодитерські вироби, замовляти торти за власним дизайном.

Також була створена Use–case діаграма для адміністратора сайту (див. рис. 3.2).

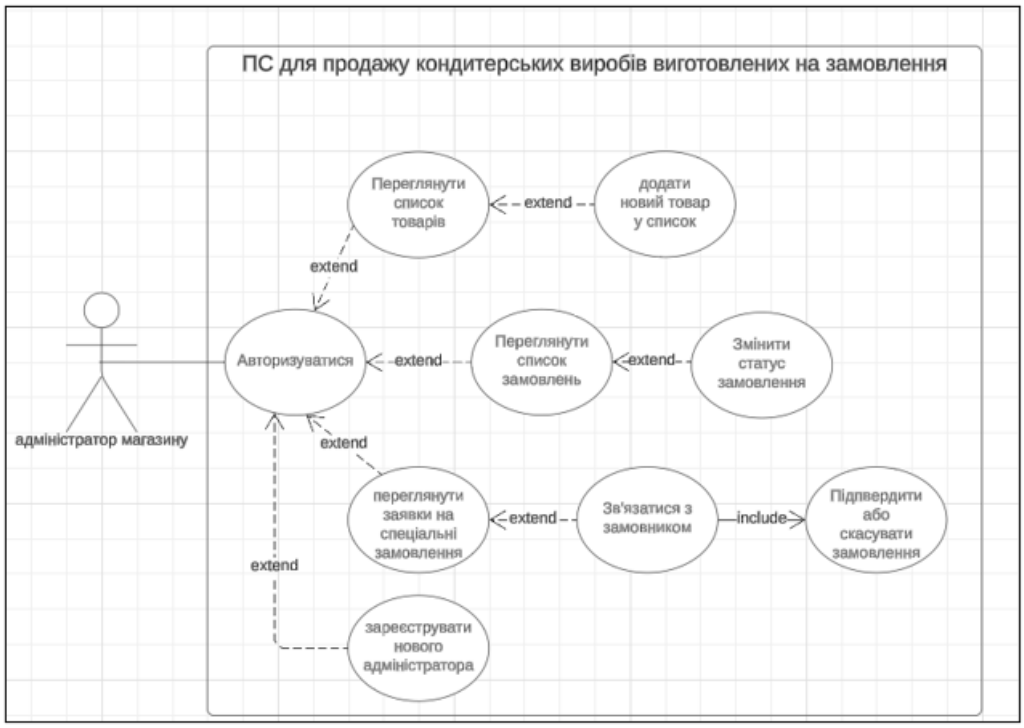

Рисунок 3.2 – Use–case діаграма застосунку для адміністратора(рисунок виконаний самостійно)

Адміністратор має функціонал авторизації, та різні взаємодії з товаром, перегляд і оформлення замовлень, та додавання інших адміністраторів. Він має всі функції звичайного користувача, і додатково може адмініструвати сайт.

3.2 Проєктування архітектури ПЗ

Загалом у застосунку буде використовуватись клієнт–серверна архітектура(див. рис. 3.3). Вона являє собою модель взаємодії, де клієнтські пристрої запитують та отримують ресурси або послуги від серверів. Сервери обробляють ці запити, виконують необхідні операції і надсилають результати назад клієнтам. Ця архітектура є основою для більшості сучасних мережевих додатків та сервісів, таких як веб-сайти, електронна пошта та бази даних, забезпечуючи ефективний розподіл ресурсів і централізоване управління даними[9].

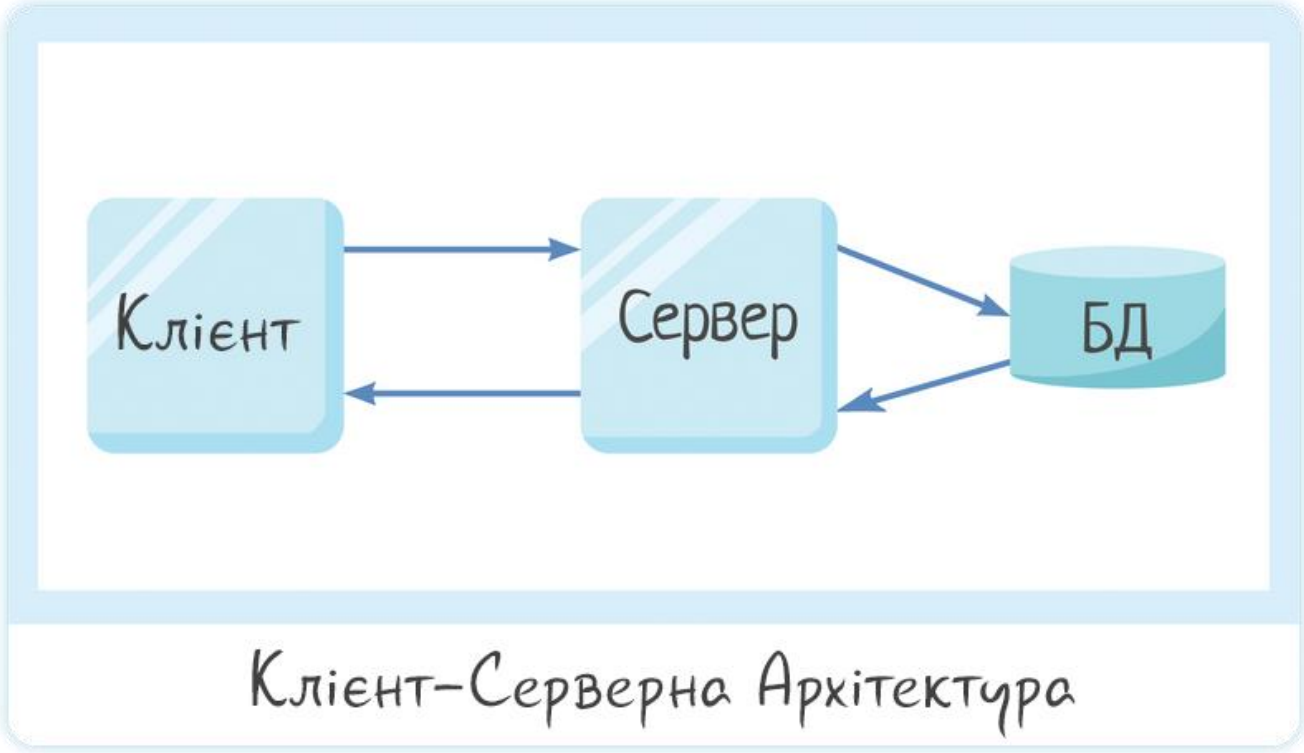

Рисунок 3.3 – Клієнт-серверна архітектура (за даними [8])

React дає можливість досить просто та зручно будувати інтерфейс з блоків, які ми робимо, в нього легко інтегруються інші наші бібліотеки, ось як буде виглядати структура файлів клієнтської частини(див. рис. 3.4).

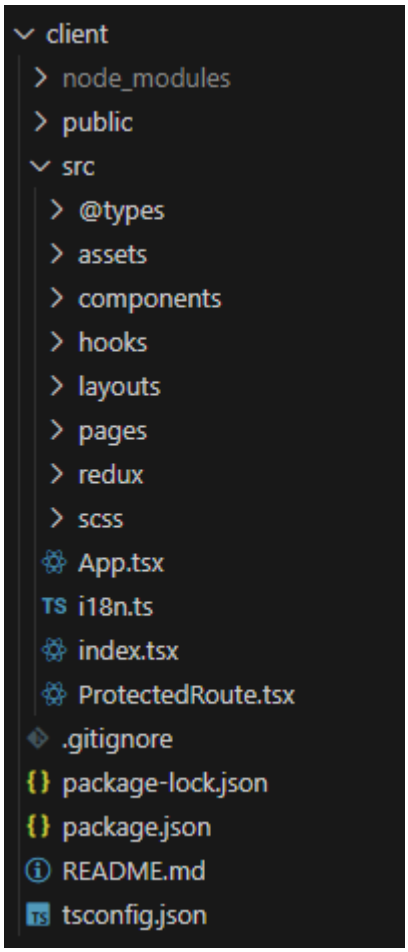

Рисунок 3.4 – Архітектура FrontEnd частини (рисунок виконаний самостійно)

Використовується архітектурний шаблон, що використовується для front–end це «Modular Architecture», він пропонує структуру, де код організований у окремі, незалежні модулі, що відповідають за конкретні функції або частини додатку[9]. Це забезпечує високу масштабованість, оскільки нові модулі можна додавати без значних змін до існуючого коду. Також поліпшується підтримка та рефакторинг коду, оскільки кожен модуль має чітко визначені межі та залежності, що полегшує виявлення і виправлення помилок.

Обрано цей шаблон через його здатність спростити розробку великих і складних проектів. Модульний підхід сприяє зручності командної роботи, дозволяючи різним розробникам одночасно працювати над окремими модулями без конфліктів. Це також покращує повторне використання коду, оскільки модулі можуть бути легко інтегровані або замінені. В результаті, додаток стає більш

гнучким, адаптивним до змін та простішим у підтримці, що є ключовими факторами для успішної довготривалої розробки.

На рис. 3.4 ми можемо бачити, що ми максимально розділяємо всі частина. В assets знаходиться всі елементи графіки, що нам може знадобитись, components це наші найдрібніші частинки з яких складаються сторінки, hooks має всі хуки що можуть знадобитись в будь якій частині застосунку, layouts містить всі макети для різних частин сайту, pages має всі вікна нашого додатку, redux містить логіку нашої тимчасової бази даних та scss має всі стилі до кожного елементу.

У нашому застосунку для FrontEnd частини буде використовуватись патер проєктування Redux на мові програмування TypeScript[6]. Redux – це відкритий JavaScript–бібліотека для управління станом додатку. Він найчастіше використовується разом з React для побудови користувацьких інтерфейсів. Redux є надзвичайно потужним і гнучким інструментом.

В основі Redux лежить ідея, що весь стан додатку зберігається в одному великому об’єкті, який називається store. Це робить систему прозорою і простою для розуміння, оскільки ви завжди знаєте, де знаходяться ваші дані і як вони змінюються в часі.

Коли в додатку відбувається якась подія (наприклад, користувач клікає кнопку), відправляється дія. Дія – це простий об’єкт, який описує, що щойно сталося. Він містить тип дії (який описує, що сталося) і деякі дані (які описують деталі події).

Ці дії обробляються функціями, які називаються reducers. Reducer – це чиста функція, яка приймає поточний стан і дію, а потім повертає новий стан. Важливо зауважити, що reducer ніколи не змінює поточний стан; він завжди повертає новий об’єкт.

Цей патерн дозволяє вам писати додатки, які сильно відрізняються від традиційних шаблонів MVC. Ви можете легко відстежувати, як стан вашого додатку змінюється в часі, і ви можете впевнено знати, що ваш додаток завжди буде працювати так, як ви очікуєте.

3.3 Проєктування структури зберігання даних

У застосунку була обрана СКБД Mongo, яка дозволяє легко масштабувати проєкт та вносити зміни(див. рис. 3.5).

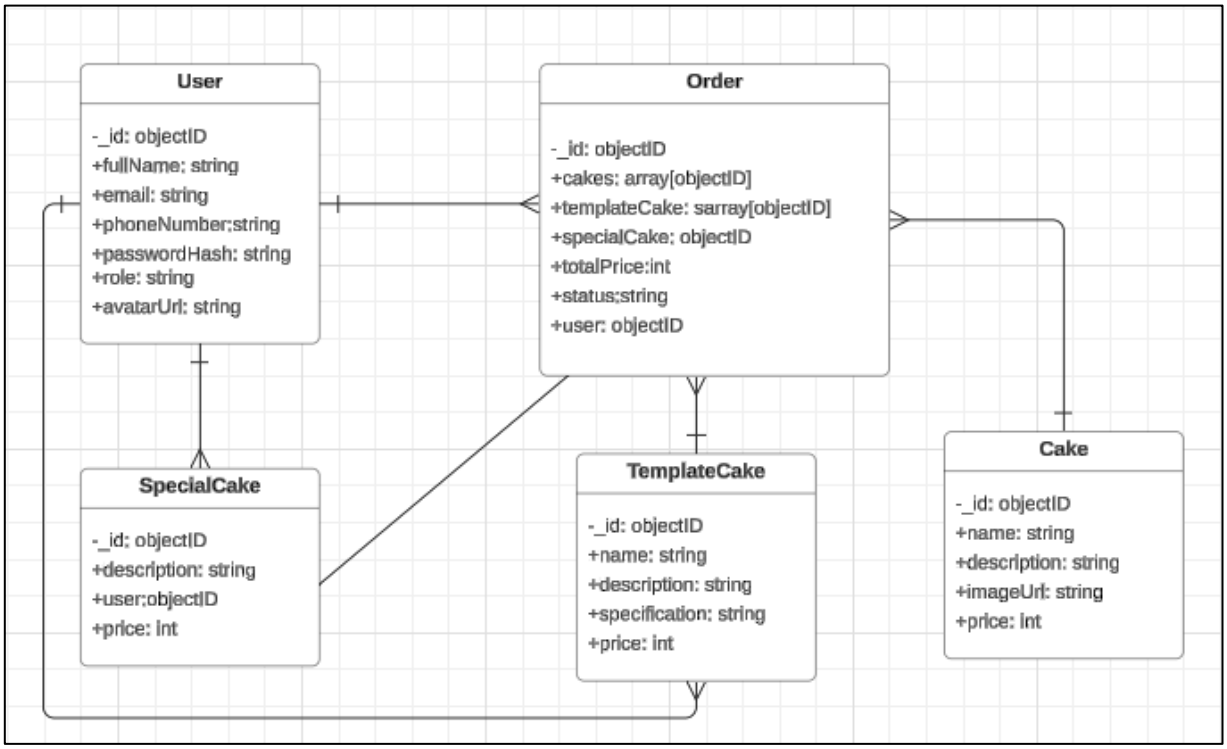

Рисунок 3.5 – ER діаграма бази даних(рисунок виконаний самостійно)

Дані зберігаються в 5 сутностях: користувач, торт, торт по шаблону, спеціальний торт, замовлення. Замовлення містить у собі список замовлених тортів, їх загальну ціну та ID користувача. Спеціальні торти оформлюються по одному на замовлення. Звичайні торти та торти створені по шаблону зберігаються у замовлені через масиви їхніх ID.

На FrontEnd використано ReduxToolkit, він дає можливість дізнатись з серверної частини данні, що цікавлять, та записати до Redux, що зменшить навантаження на сервер, адже можна буде локально дізнаватись дані, що цікавлять[6]. Сам Redux це основний файл управління сховищем, у нас це store.ts та самі сховища, тобто слайси (див. рис. 3.6).

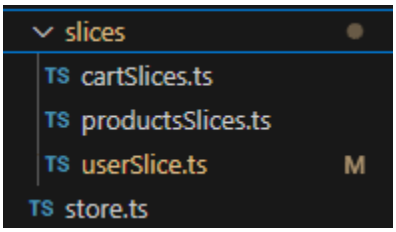

Рисунок 3.6 – приклад вигляду структури ReduxToolkit

3.4 Розробка UI/UX інтерфейсу

Для онлайн покупок кондитерських виробів була обрана кольорова палітра, яка створює теплу, привабливу та затишну атмосферу, що сприяє викликанню апетиту та позитивних емоцій у відвідувачів. Основний білий фон забезпечує чистоту та легкість дизайну, дозволяючи іншим кольорам та зображенням виглядати яскравіше та контрастніше. Світлі нейтральні відтінки, такі як світло–сірий та вторинний сірий, підтримують загальну гармонію та збалансованість, не перевантажуючи візуально користувача[11].

Принципи кольорового кола включають комплементарні, аналогічні та тріадні схеми кольорів. Комплементарні кольори знаходяться напроти один одного на кольоровому колі і створюють високий контраст. Аналогічні кольори розташовані поруч і забезпечують гармонію. Тріадні кольори рівномірно розташовані на колі і дають динамічний, але збалансований ефект. Кольори за цими принципами було створено за допомгою Adobe Color[7] (див. рис. 3.7).

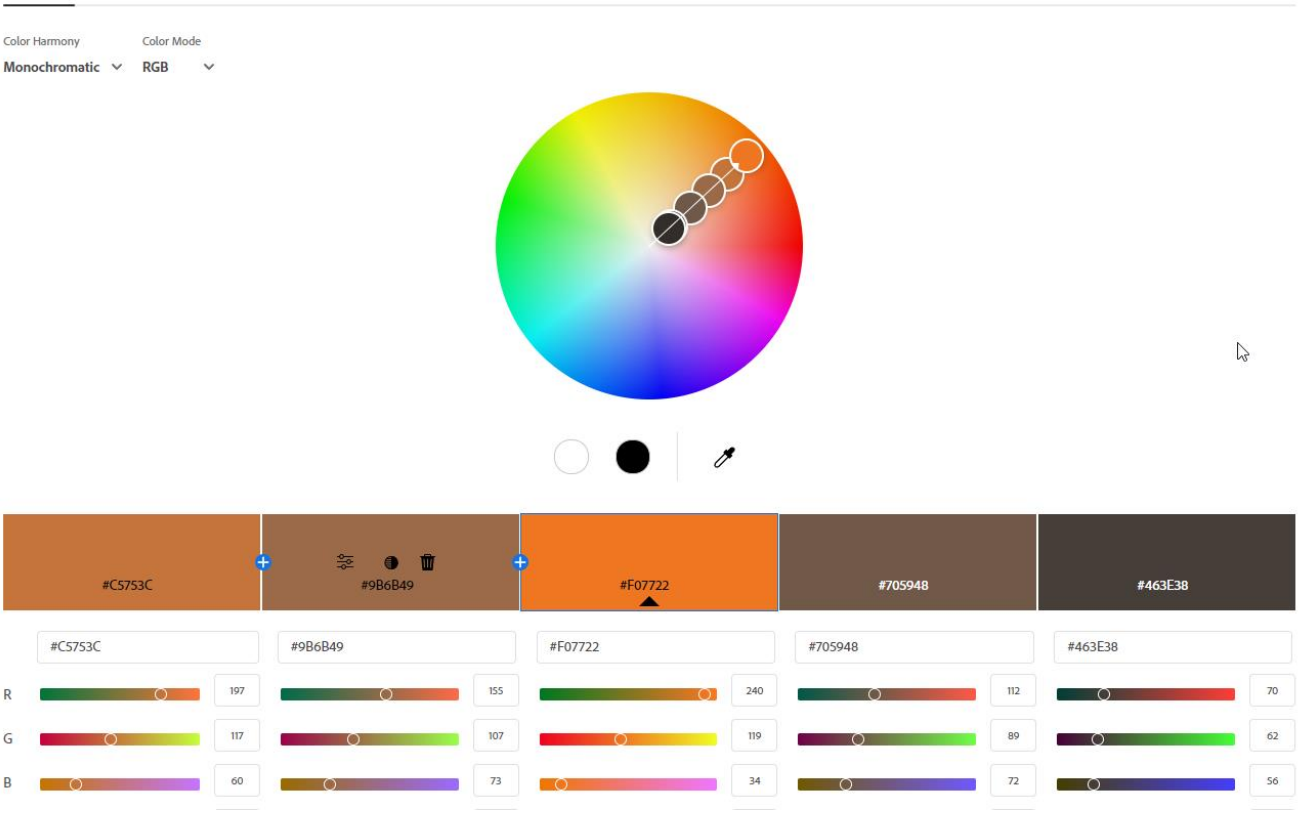

Рисунок 3.7 – приклад кольорового кола для вибору(за даними [7])

Щоб краще зрозуміти структуру сайту, створимо скетчі прикладу того, як виглядатиме наш застосунок[5]. Почнемо з головної сторінки, де схематично розмістимо основні блоки з яких буде складатись наш інтерфейс, де буде короткий опис блоків(див. рис. 3.8).

Рисунок 3.8 – Скетч головної сторінки (рисунок виконаний самостійно)

Розглянемо скетч, що буде універсальною базою для списку будь яких товарів (див. рис. 4.9).

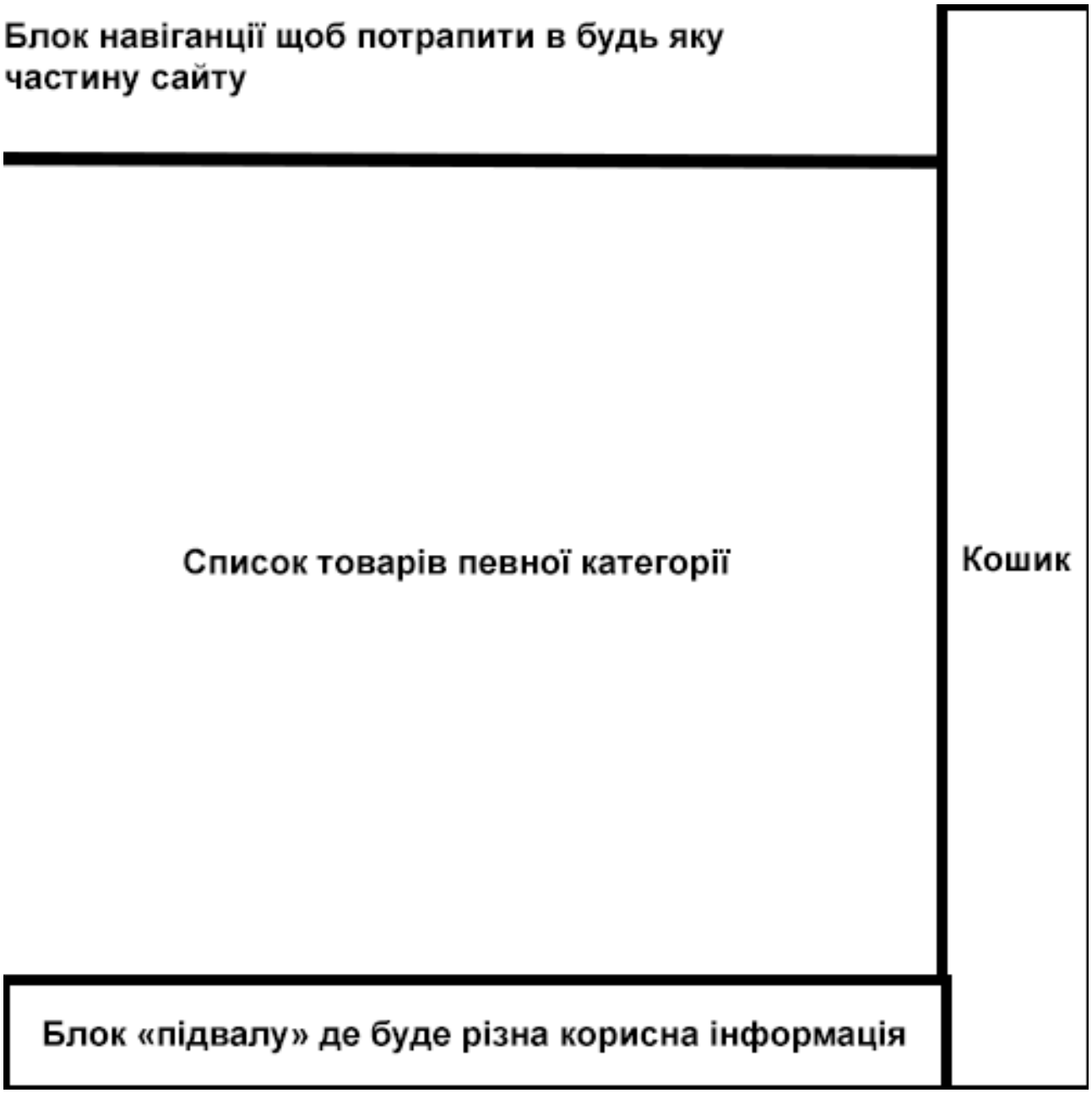

Рисунок 3.9 – Універсальна база для списку товарів (рисунок виконаний самостійно)

Розглянемо скетч з прикладом налаштування акаунту, де буде меню з виборами різних блоків, що можуть бути корисні клієнту (див. рис. 3.10).

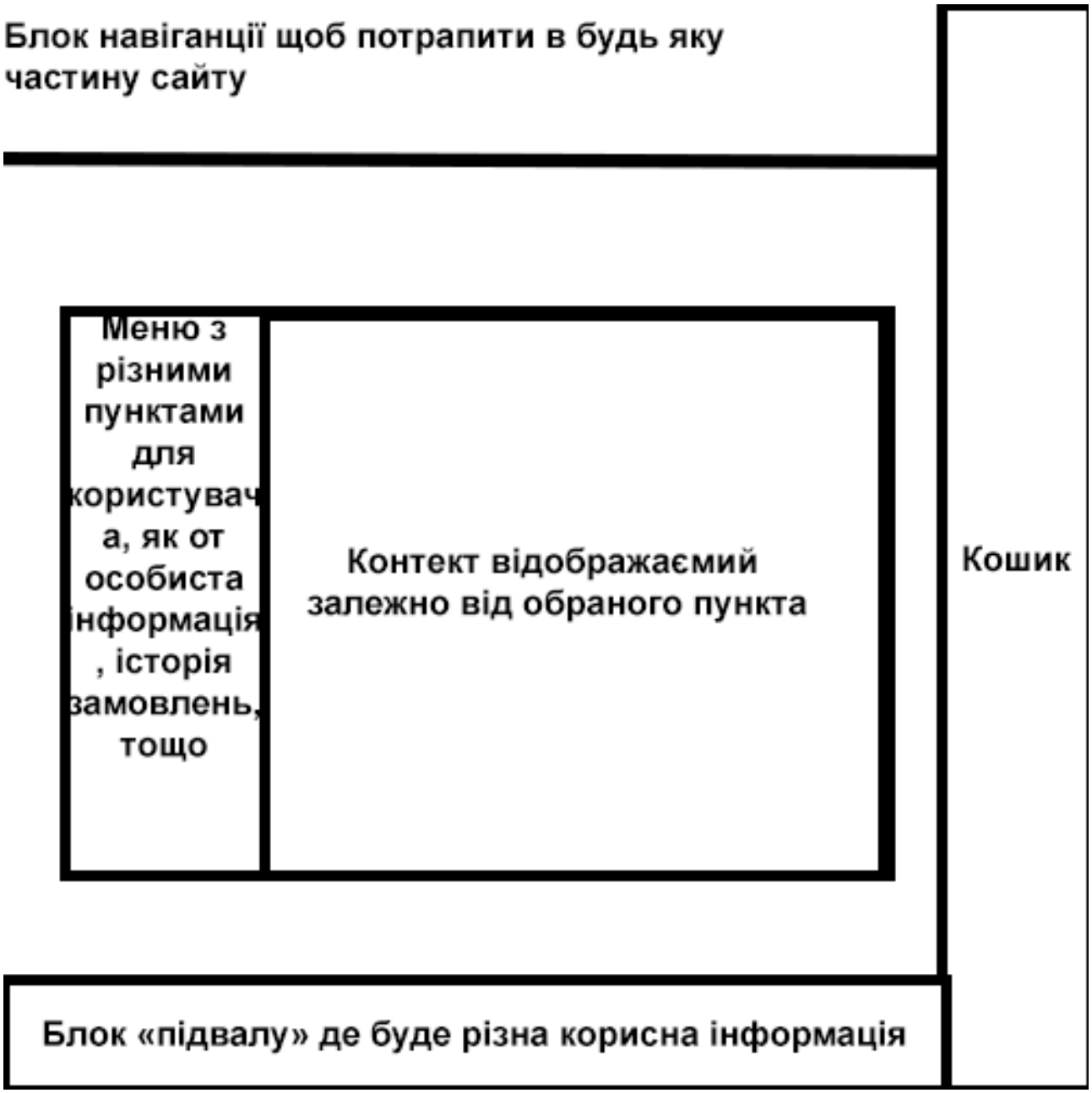

Рисунок 3.10 – Приклад наповнення аккаунту (рисунок виконаний самостійно)

Також розглянемо приклад сторінки оформлення замовлення(див. рис. 3.11).

Рисунок 3.11 – Скетч оформлення замовлення (рисунок виконаний самостійно)

Також важливим елементом нашого застосунку є конструктор тортів, тож зробимо скетч з цією сторінкою (див. рис. 3.12).

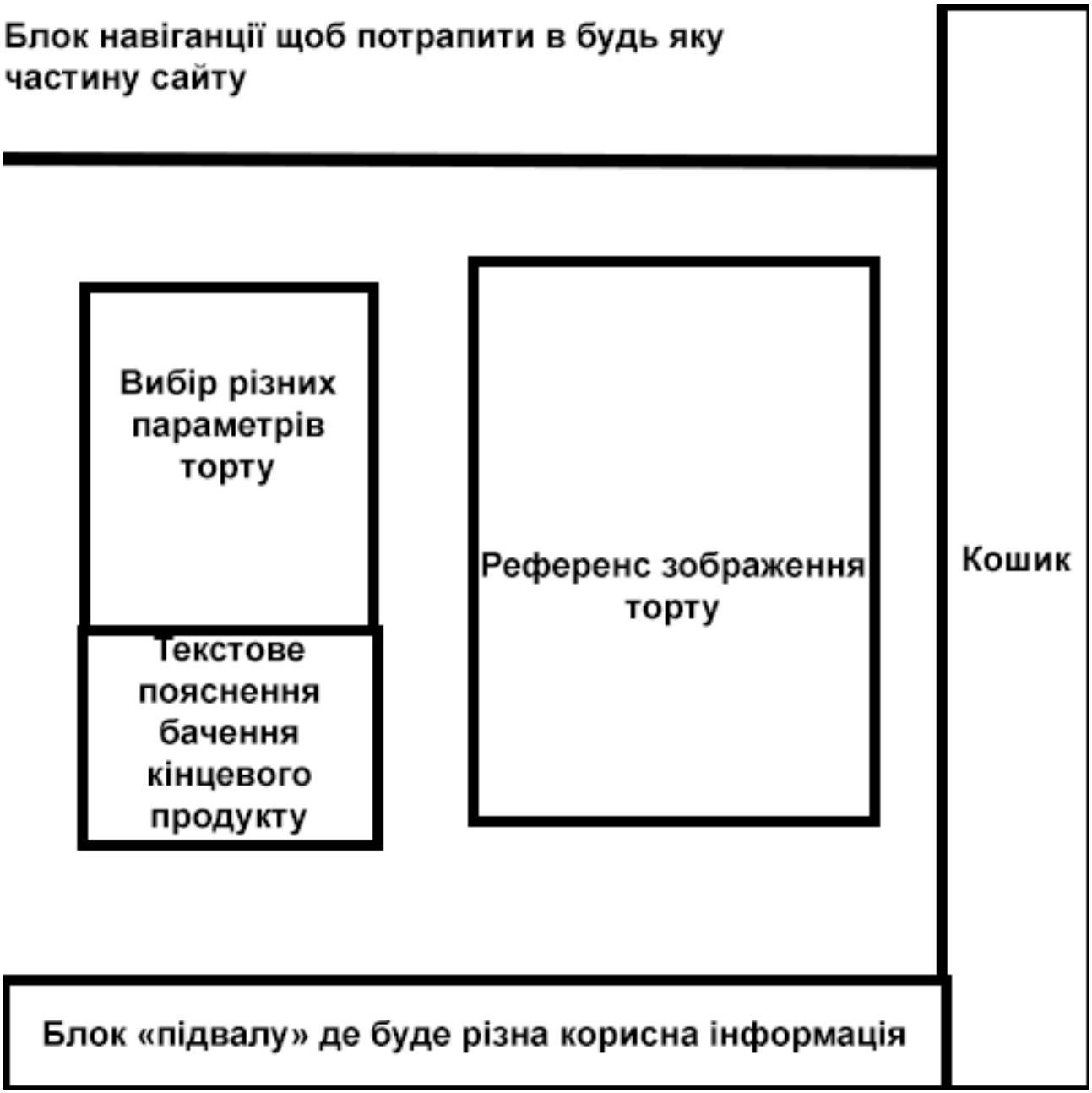

Рисунок 3.12 – Скетч конструктору торту (рисунок виконаний самостійно)

За цими скетчами можна легко зрозуміти базові елементи інтерфейсу, та приблизний вигляд сторінок, що облегшить подальше проектування інтерфейсу.

Щоб краще зрозуміти кількість сторінок та головних елементів, які будуть містити ці сторінки створимо діаграму екранів[4] (див. рис. 3.13).

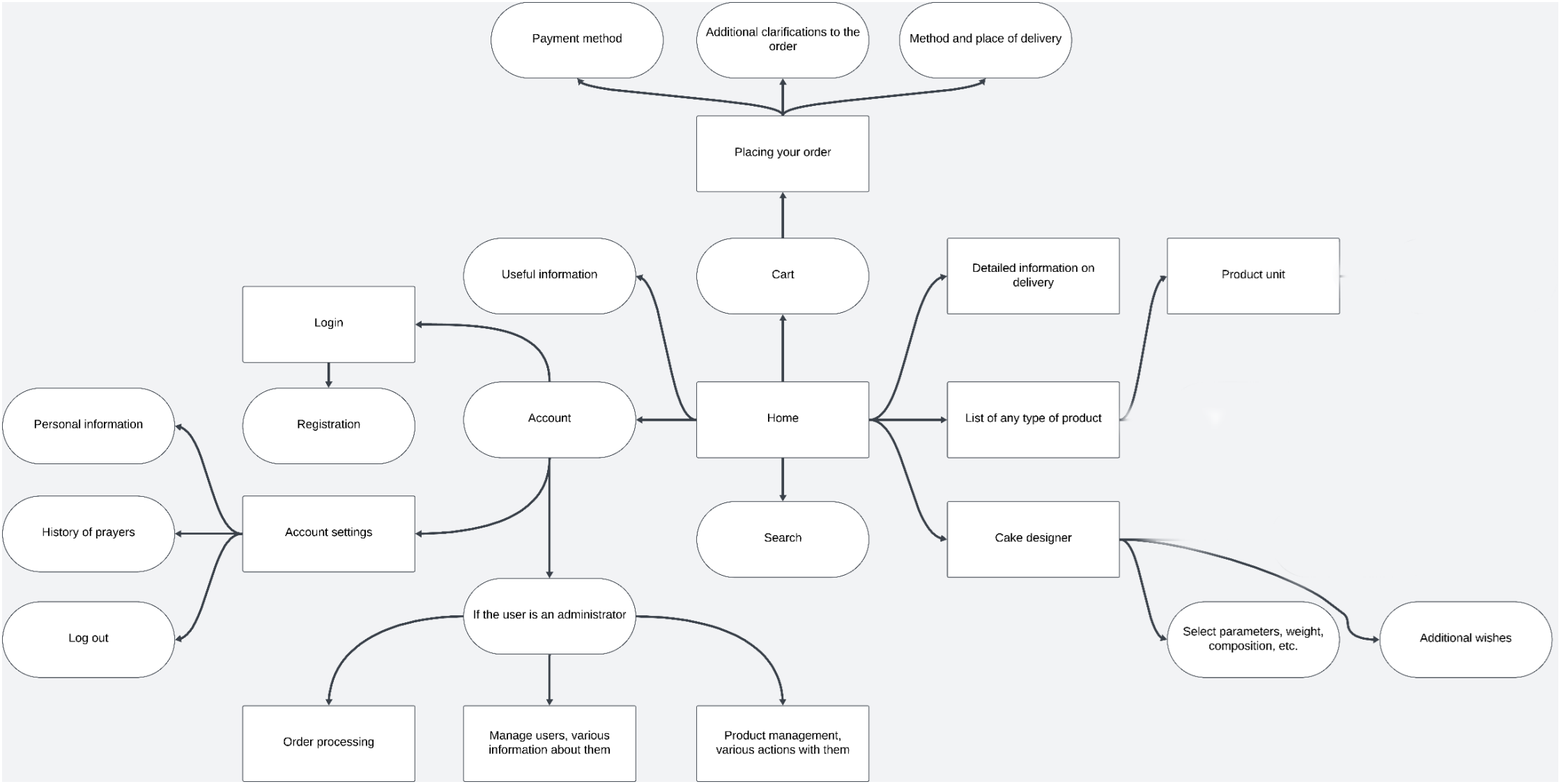

Рисунок 3.13 – Діаграма екранів (рисунок виконаний самостійно)

Діаграма екранів[4] – це візуальне зображення, що описує структуру та поведінку користувацького інтерфейсу програмного додатка або веб–сайту. Вона містить набір екранів, які відображаються користувачу під час використання системи, а також зв'язки між ними.

На діаграмі (див. рис. 3.13), відображені власне самі екрани, у прямокутних фігурах, та основні елементи що містять ці екрани, у закруглених фігурах[14].

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

На основі раніше розроблених діаграм та скетчів починаєм розробку та верстку інтерфейсу(див. рис. 4.1).

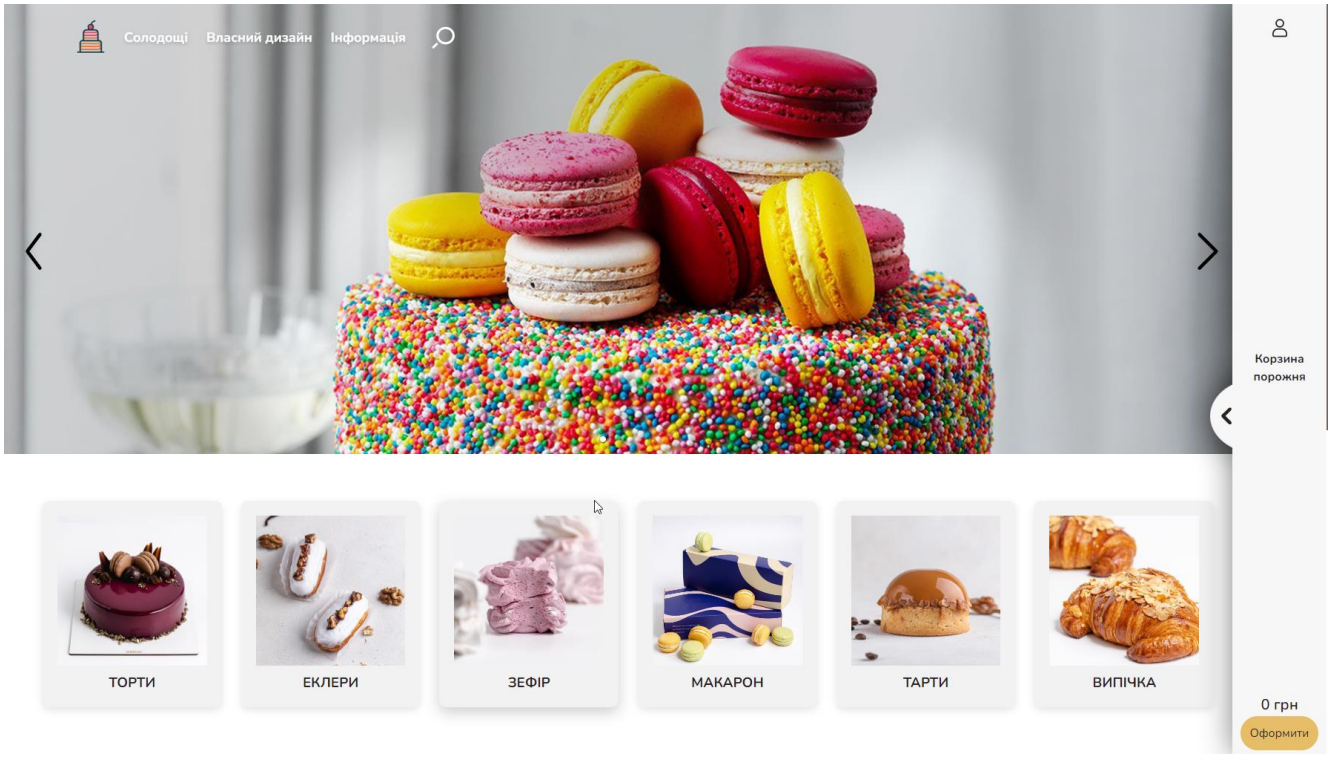

Рисунок 4.1 – Перша частина головної сторінки сайту(рисунок виконаний самостійно)

На головній сторінці сайту ми відразу бачимо карусель в які демонструються різні товари, зверху блок header в якому присутня вся навігація, також нижче демонстративно з рисунками показані різні категорії випічки. Також справа ми бачимо корзину до якої додаються товари, для перегляду більш детальної інформації можна розгорнути корзину.

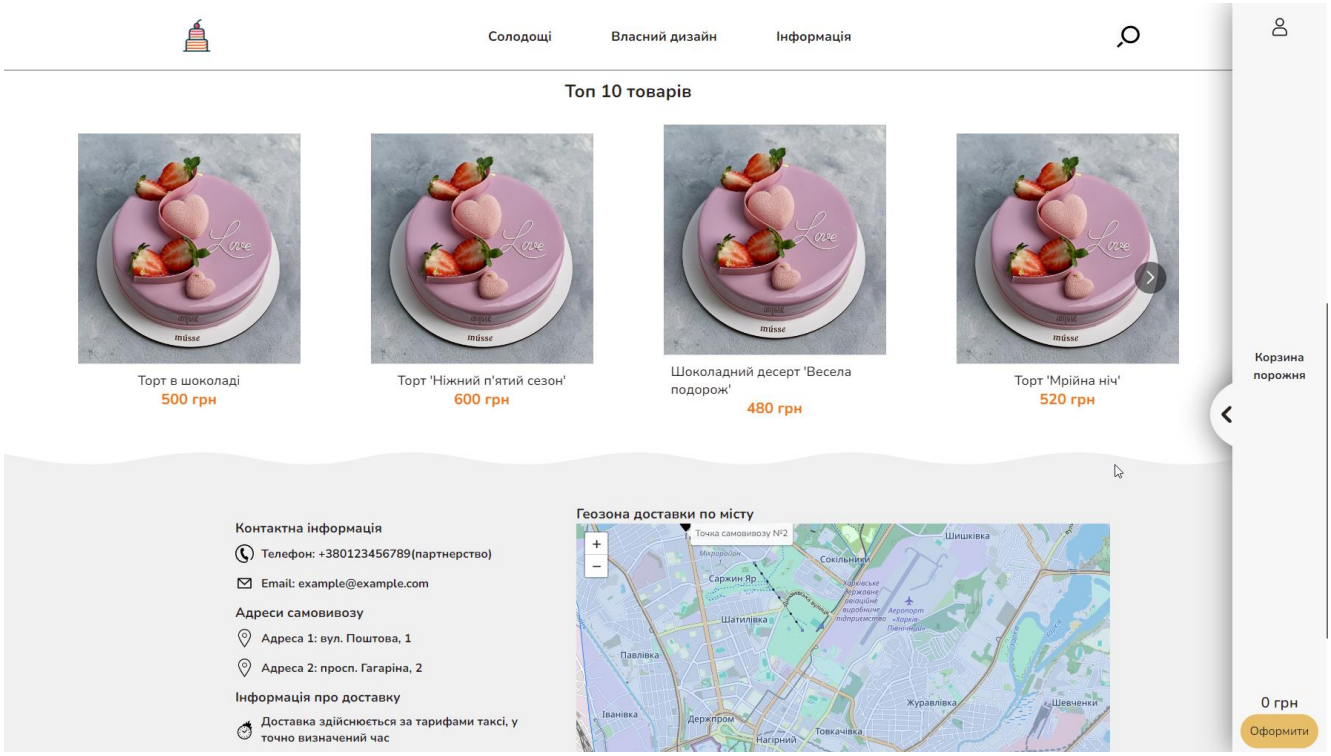

Рисунок 4.2 – Друга частина головної сторінки(рисунок виконаний самостійно)

На рис. 4.2 ми бачимо, що опускаючись до низу сайту, для зручності навігації з’являється header який зручно читати на будь якому фоні. Нижче знаходиться вибір топ товарів, та блок footer на якому коротка корисна інформація, та геозона доставки, та пункти самовивозу.

На прикладі каруселі топ товарів продемонструю код, та зручність модульної архітектури, не потрібно демонструвати код всієї сторіки, можна використовувати окремі компоненти.

```
const ProductCarousel = () => {
  const dispatch = useDispatch<AppDispatch>();
  const products = useSelector((state: RootState) =>
state.Products.products);
  useEffect(() => {
    dispatch(fetchTopProducts());
  }, [dispatch]);
  return (
    <div className="carousel-wrapper">
      <h2 className="carousel-title">Топ 10 товарів</h2>
```

```
      <Carousel responsive={responsive} containerClass="carousel-
container">
        {products.map((product) => (
          <div className="carousel-item" key={product._id}>
            <ProductItem
              photo={product.src}
              price={product.price}
              name={product.title}
              id={product._id}
            />
          </div>
        ))}
      </Carousel>
    </div>
  );
};
```

У наведеному коді використовуються TypeScript для типізації та написання логіки компонента ProductCarousel, який створюється за допомогою бібліотеки React. Для керування станом додатка використовується Redux, а з його інтеграцією з React допомагає бібліотека react–redux, яка надає хуки useSelector для доступу до стану і useDispatch для диспетчеризації асинхронної дії fetchTopProducts. Бібліотека react–multi–carousel використовується для створення каруселі, яка відображає продукти, отримані з глобального стану, а стилізація компоненту виконується за допомогою SCSS.

Перевага TypeScript у тому, що він надає статичну типізацію, яка значно покращує безпеку і якість коду[13]. Завдяки типам розробники можуть виявляти помилки на етапі написання коду, а не під час виконання, що знижує кількість багів і підвищує стабільність додатку. Наприклад, типізація RootState і AppDispatch у прикладі дозволяє забезпечити правильність структури стану та методів диспетчеризації, що знижує ймовірність помилок типу та неправильного використання даних. Крім того, TypeScript покращує інтеграцію з IDE, надаючи

автодоповнення, перевірку типів у реальному часі та навігацію по коду, що значно спрощує процес розробки і робить його більш ефективним[12].

Переваги React забезпечують компонентний підхід до побудови інтерфейсів, що сприяє створенню модульного і повторно використовуваного коду. Кожен компонент виконує окрему функцію, що полегшує його тестування, налагодження та підтримку. У наведеному прикладі компонент ProductCarousel відповідає за відображення каруселі продуктів, а ProductItem за рендеринг окремого продукту, що дозволяє легко змінювати або покращувати кожен з них незалежно один від одного. React також оптимізує оновлення DOM завдяки віртуальному DOM, що робить рендеринг швидшим і ефективнішим, особливо при роботі з великими наборами даних. Це підвищує продуктивність додатку і забезпечує кращий користувацький досвід.

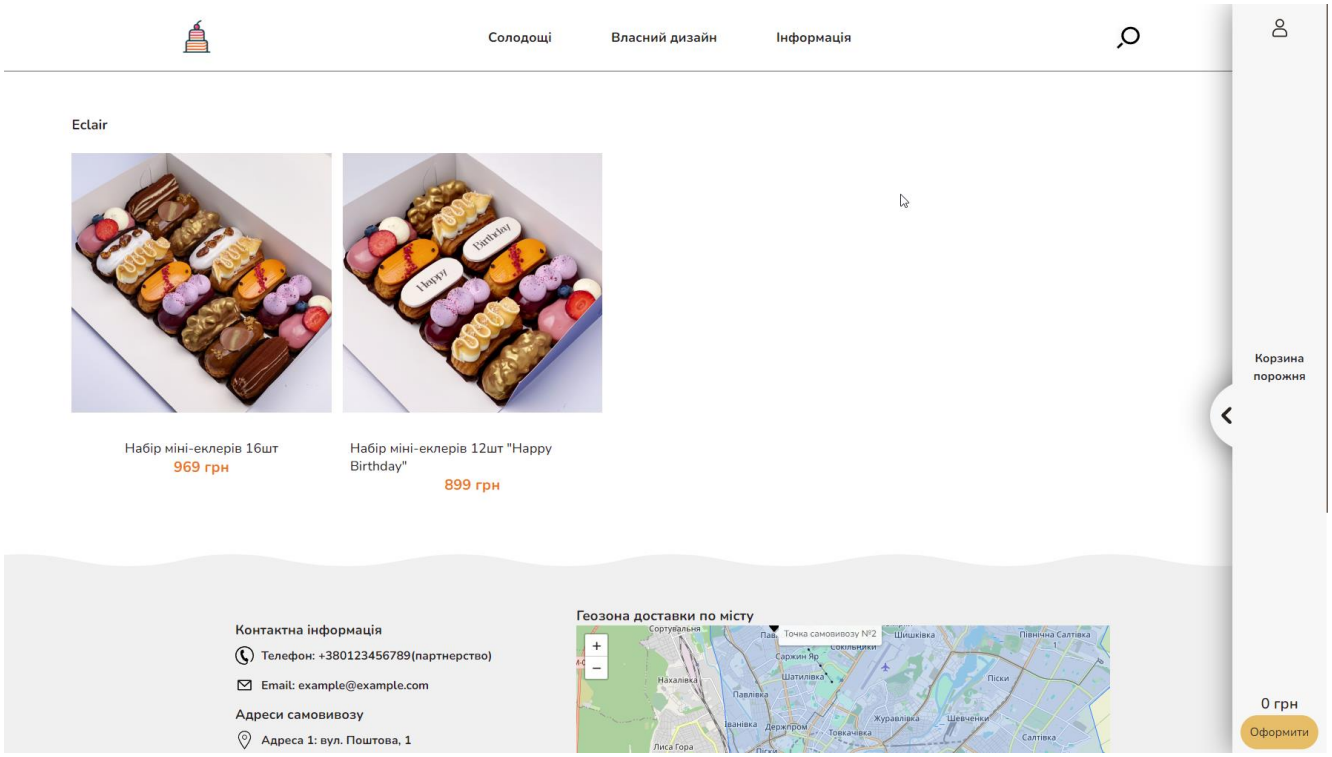

Рисунок 4.3 – Приклад відображення товарів(рисунок виконаний самостійно)

На рис. 4.3 можна побачити як перейшовши по посиланню на категорію еклери, відображається сіткою кожен товар, натиснувши на товар відкривається сторінка з деталями товару, де можна замовити цей продукт(див. рис. 4.4).

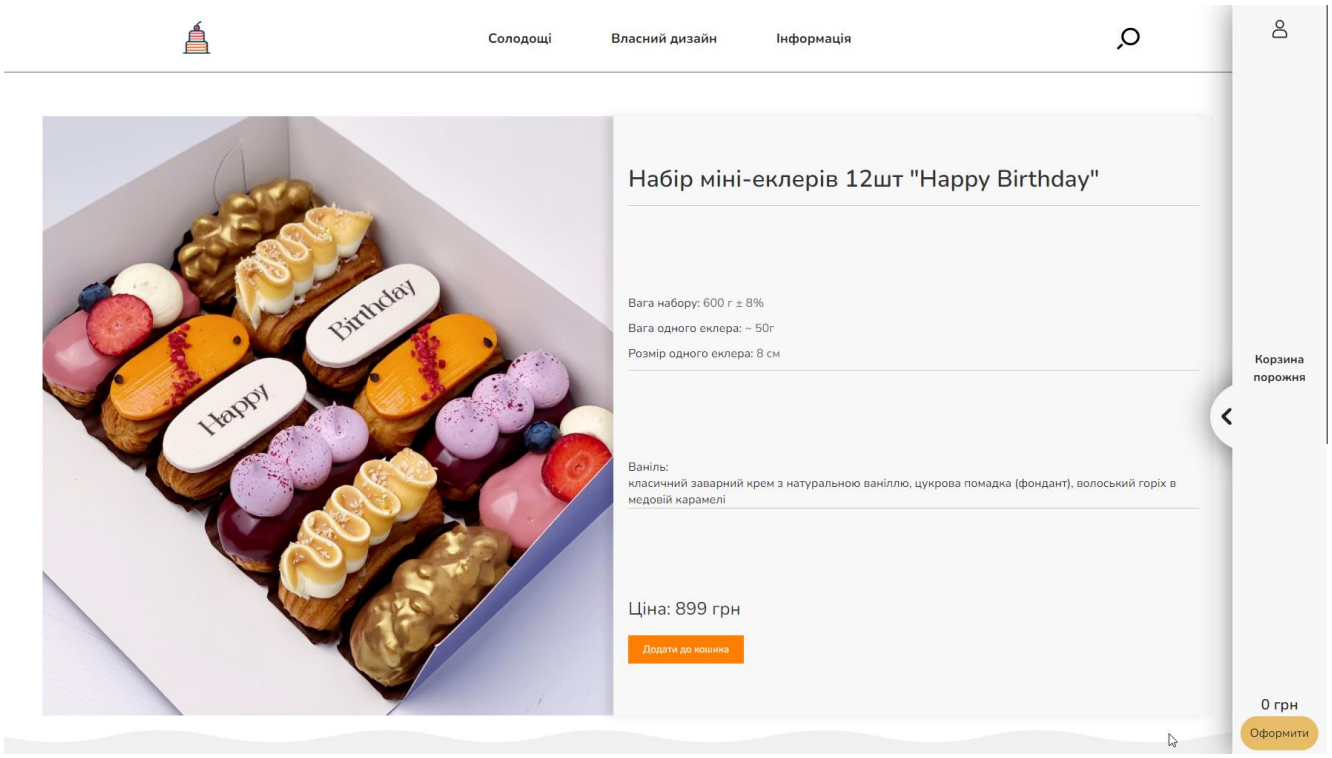

Рисунок 4.4 – Приклад відображення деталей товару(рисунок виконаний самостійно)

На рис. 4.4 ми бачимо всю інформацію про товар, та можемо додати його до кошику(див. рис. 4.5).

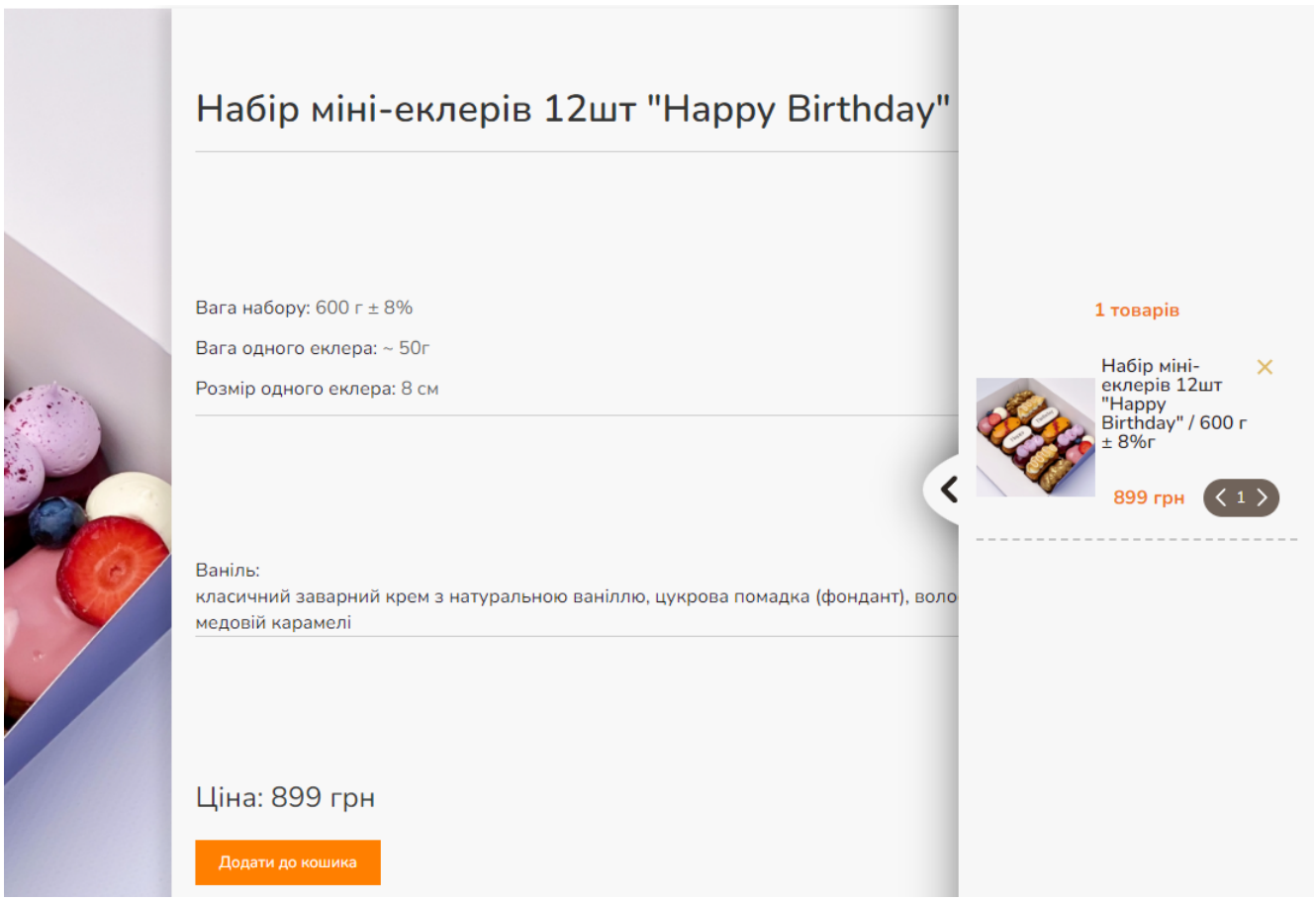

Рисунок 4.5 – Товари у кошику(рисунок виконаний самостійно)

На рис. 4.5 видно кошик та товар у ньому, тут можна видалити його, збільшити кількість обраного товару та перегляну різну інформацію про нього.

Також можна створити власний дизайн торту(див. рис. 4.6).

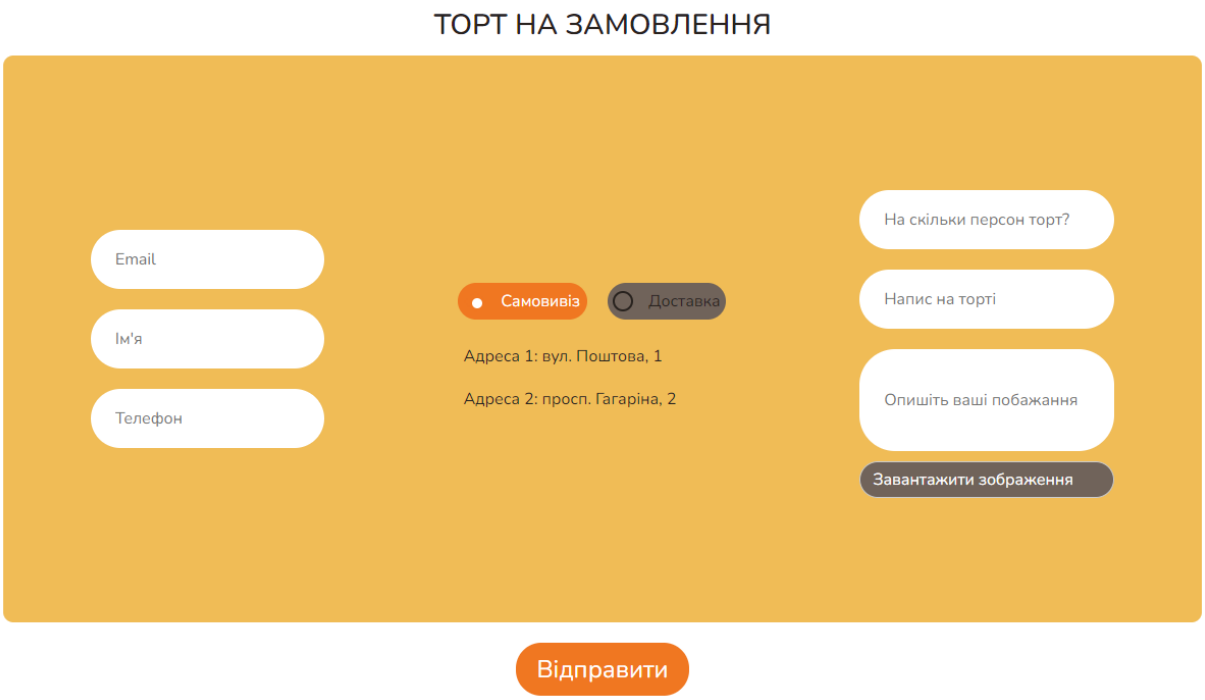

Рисунок 4.6 – Форма створення власного дизайну(рисунок виконаний самостійно)

На рис. 4.6 видно форму для створення власного торту, тут користувач додає власні контактні данні, обирає спосіб доставки та описує що він очікує побачити, може додати приклад дизайну, після цього відправляє цю форму і менеджер буде зв’язуватись з клієнтом та обговорювати це замовлення.

Схеми замовлень звичайних тортів та зі своїм дизайном дещо відрізняються(див. рис. 4.7).

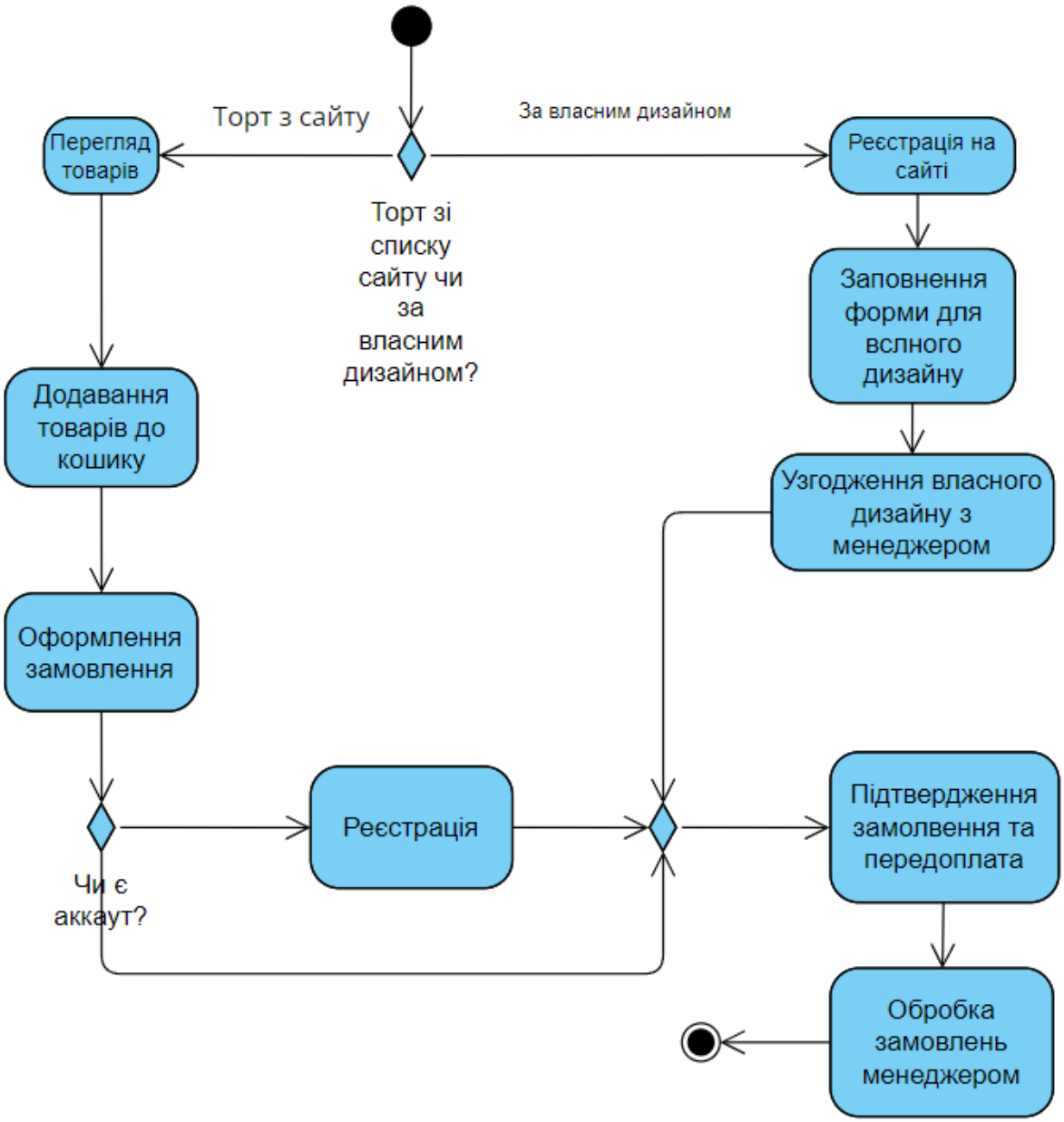

Рисунок 4.7 – UML діаграма активності(рисунок виконаний самостійно)

Діаграма показує процес оформлення замовлення на торт на сайті, який включає два основні шляхи: замовлення торта зі списку на сайті або замовлення торта за власним дизайном[14]. Користувач починає з перегляду товарів, додає обраний торт до кошика і оформляє замовлення, реєструючись на сайті за потреби. Якщо користувач обирає торт за власним дизайном, він заповнює відповідну форму і узгоджує дизайн з менеджером. Після підтвердження замовлення і передоплати менеджер обробляє замовлення.

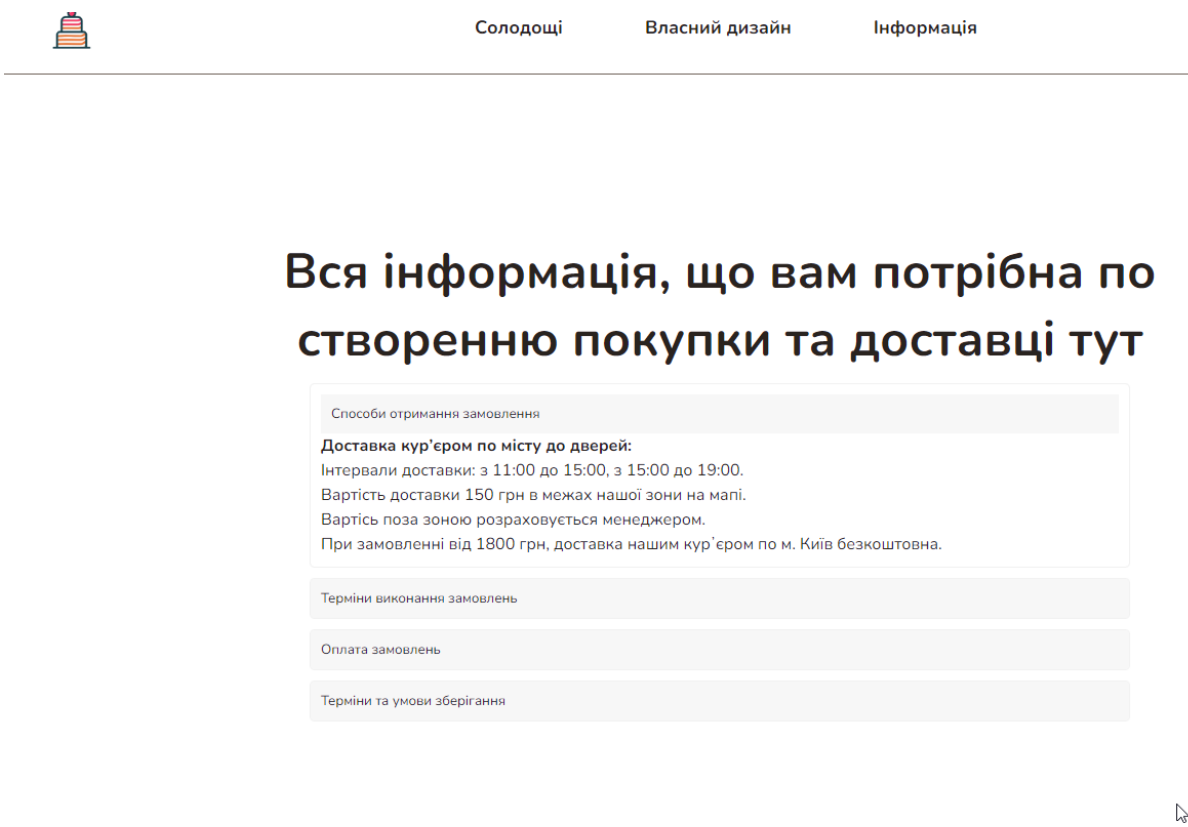

Рисунок 4.7 – Сторінка з популярними питаннями(рисунок виконаний самостійно)

На рис. 4.7 видно меню з списком питань, які часто задають, і можна переглянути відповіді на ці питання.

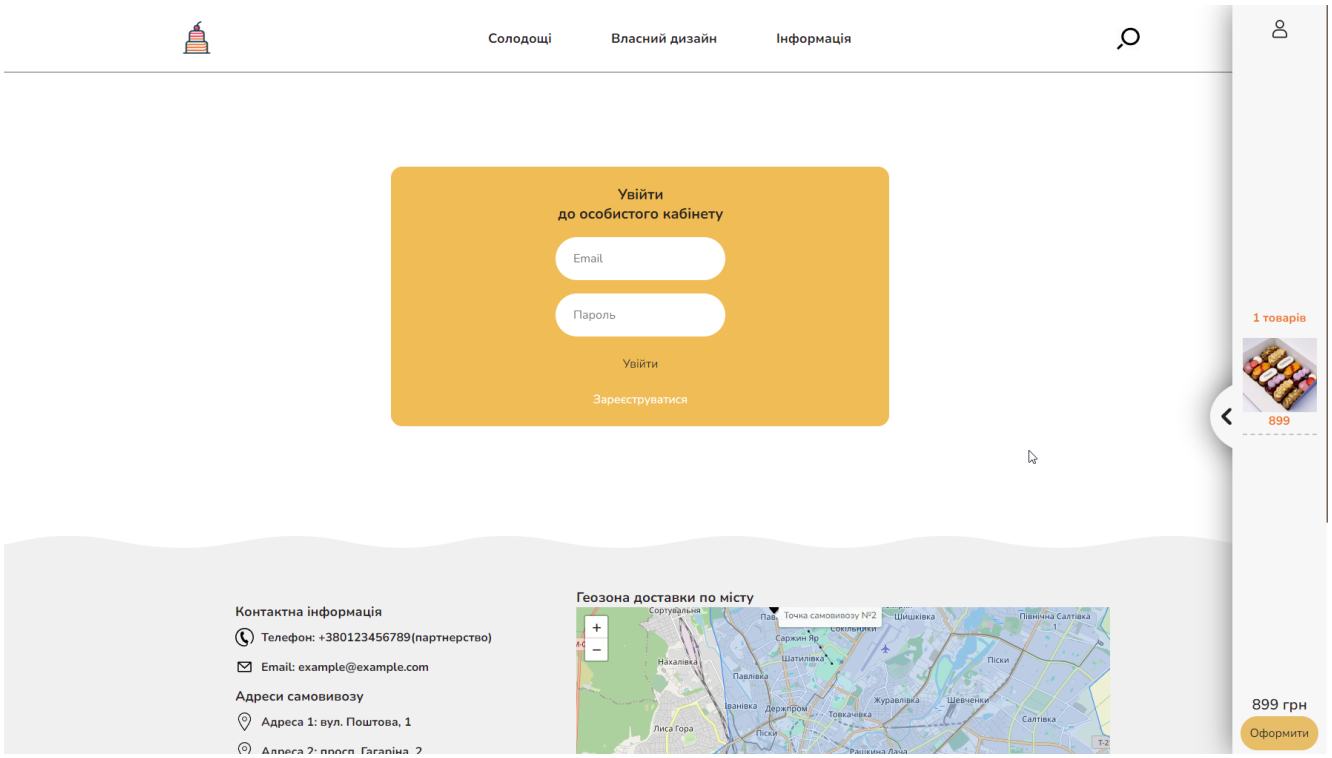

Рисунок 4.7 – Форма логіну(рисунок виконаний самостійно)

На рис. 4.7 зображена форма для входу в аккаунт, або якщо його немає його можна створити, для цього використовується email та пароль(див. рис. 4.8).

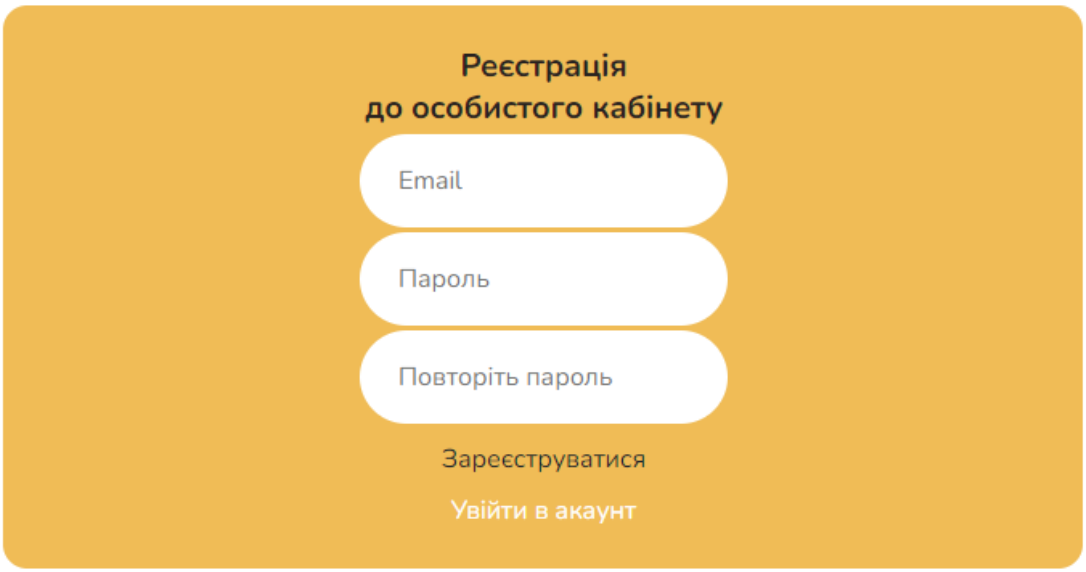

Рисунок 4.8 – Форма реєстрації(рисунок виконаний самостійно)

Зареєструватись може будь хто, При реєстрації данні користувача проходять необхідну валідацію, як на стороні клієнта, так і на стороні серверу.Пароль користувача зберігається в зашифрованому вигляді, а процес підтвердження аутентифікації проходиться за допомогою токену.

Після входу в аккаунт користувач може додати особисту інформацію для оформлення замовлень, переглянути замовлення та вийти з акаунту(див. рис. 4.9).

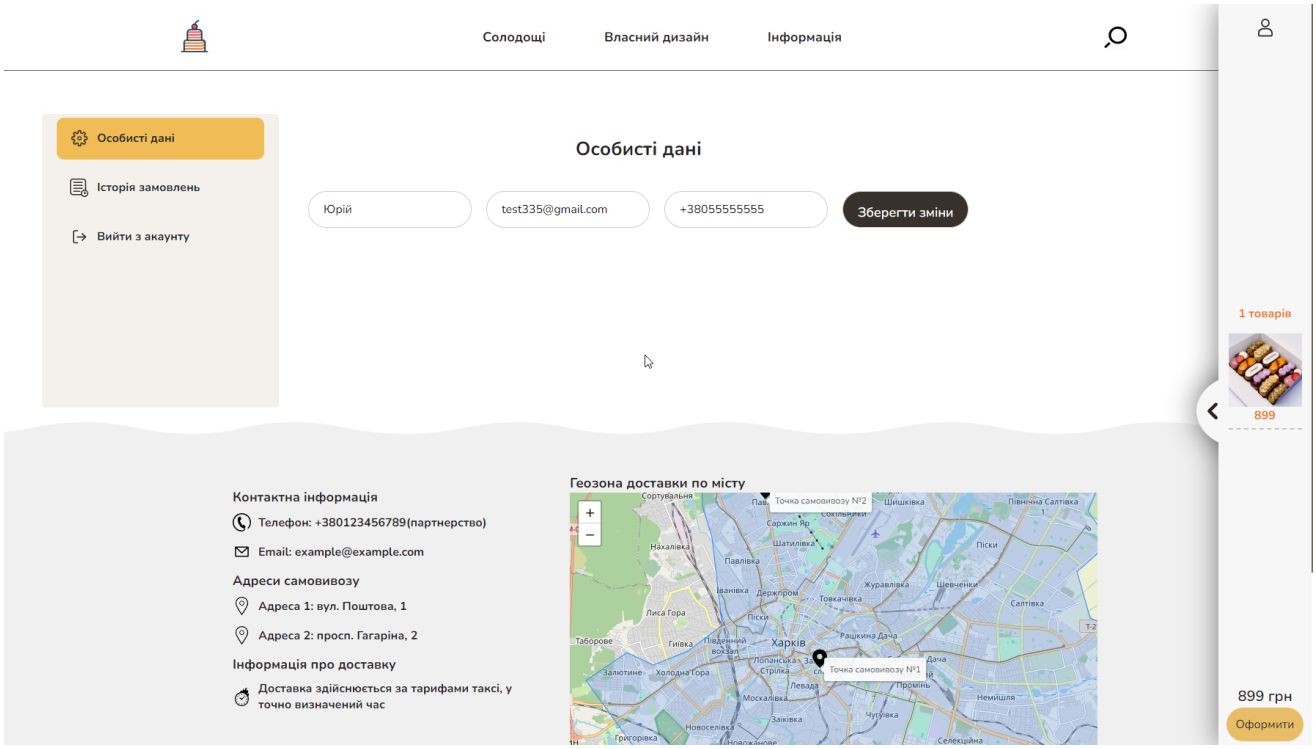

Рисунок 4.9 – Сторінка аккаунту(рисунок виконаний самостійно)

На цій ж сторінці можна перейти до панелі менеджера, її можна побачити і відкрити тільки, якщо у користувача є відповідна роль(див. рис. 4.10)..

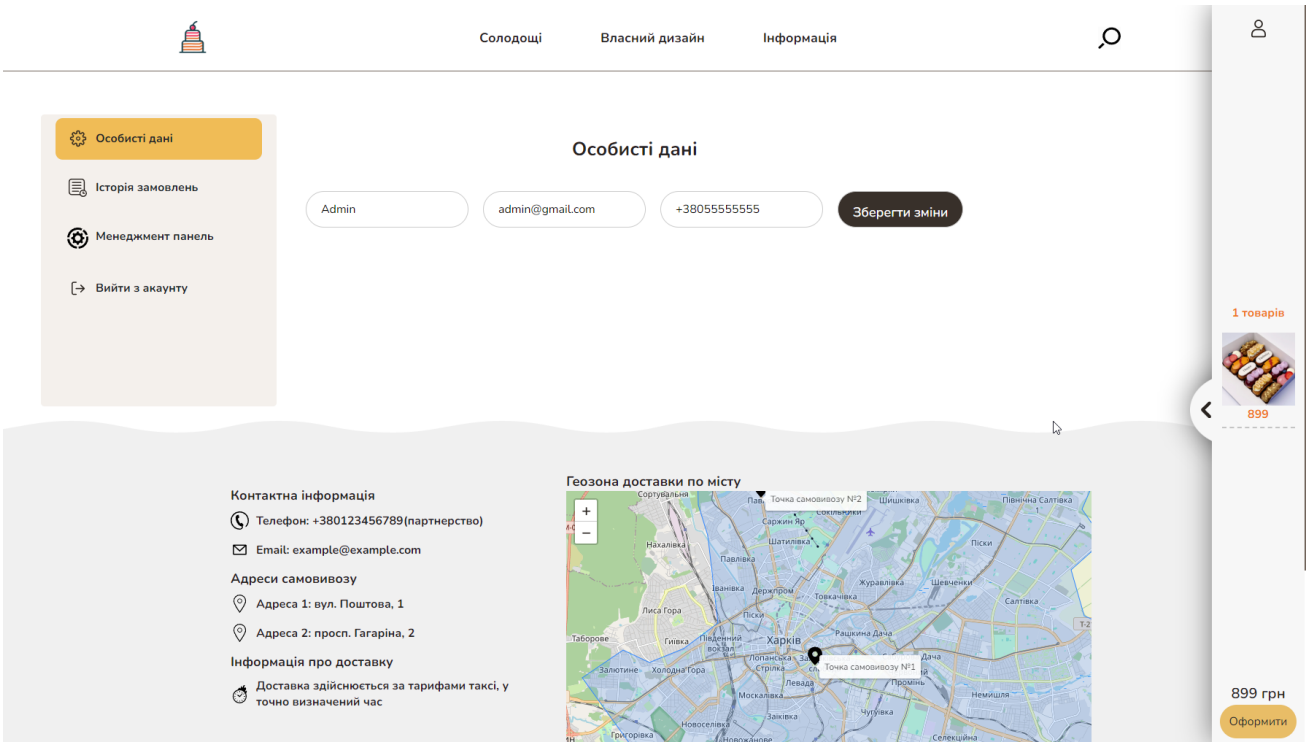

Рисунок 4.8 – Приклад аккаунту від лиця адміна(рисунок виконаний самостійно)

На сторінці менеджера можна редагувати, додавати та видаляти різні товари, також можна взаємодіяти з замовленнями, чи видавати адмінську роль іншим користувачам(див. рис. 4.11).

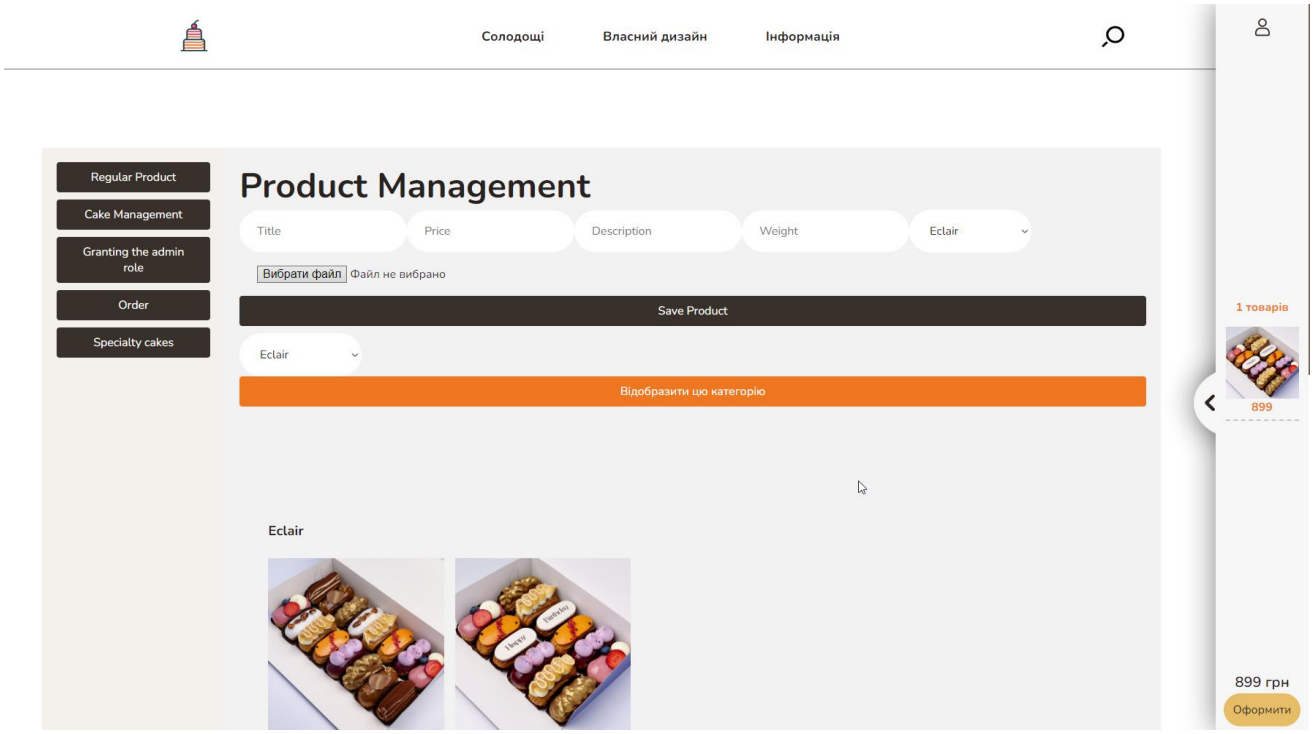

Рисунок 4.11 – Сторінка панелі менеджера(рисунок виконаний самостійно)

Розгорнувши корзину, зверху з’являється кнопка зміни мови, доступна українська і англійська(див. рис. 4.12 та 4.13).

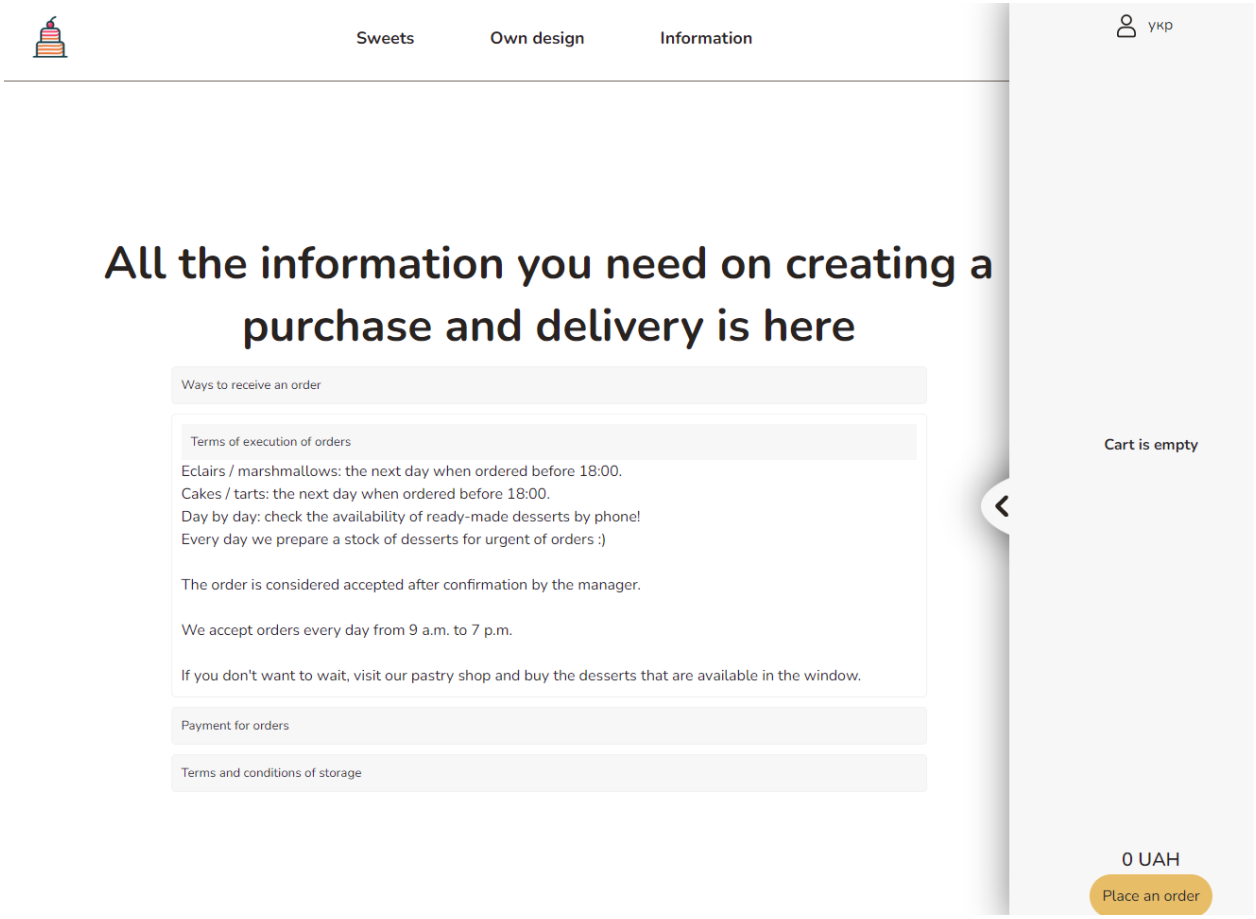

Рисунок 4.12 – Приклад англійської локалізації(рисунок виконаний самостійно)

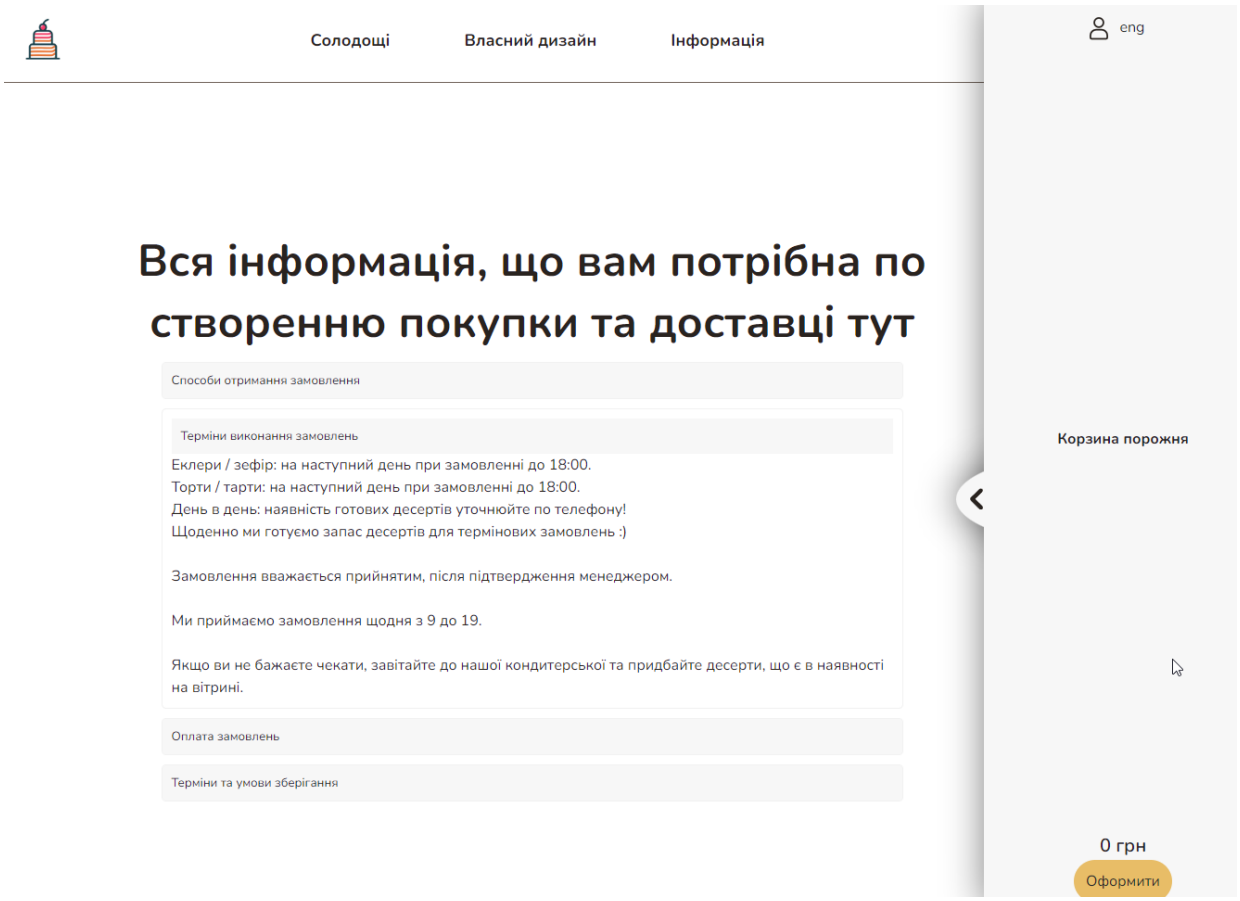

Рисунок 4.13 – Приклад української локалізації(рисунок виконаний самостійно)

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Тестування застосунку

Тестування розробленого програмного забезпечення є критичним етапом у процесі розробки, оскільки воно дозволяє виявити помилки, недоліки та забезпечити високу якість продукту перед його випуском на ринок[15].

Тестування front–end включає в себе перевірку коректності та зручності відображення інтерфейсу користувача. Основною метою такого тестування є переконатися, що веб–сторінка або додаток виглядають та працюють так, як очікується користувачем.

Під час функціонального тестування front–end перевіряється коректність роботи функціональних елементів, таких як кнопки, форми, меню тощо. Це включає перевірку взаємодії з користувачем, відправку запитів на сервер, обробку відповідей тощо. Наприклад, перевірка може включати натискання на кнопки, введення тексту у форми, перевірку валідації введених даних та інші функції.

Нефункціональне тестування front–end орієнтується на аспекти, що не входять безпосередньо в функціональність, але мають велике значення для користувача. Сюди входить тестування швидкості завантаження сторінки, адаптивності до різних пристроїв та розмірів екранів, сумісності з різними веб–браузерами та оперативними системами. Також важливим аспектом є тестування доступності, що включає перевірку можливості користувачів з обмеженими можливостями використовувати додаток або веб–сторінку[15].

Крім того, під час тестування front–end можуть проводитися також тестування відповідності дизайну стандартам та перевірка на виявлення візуальних дефектів, таких як зламані макети, неправильне розташування елементів тощо.

Усі ці види тестування допомагають забезпечити високу якість front–end частини програмного забезпечення та задовольнити потреби користувачів у зручності, ефективності та надійності.

Таблиця 5.1 – Тест–кейс №1 (таблиця виконана самостійно)

Інформація про тест–кейс			
Ідентифікатор тесту:	Тест–кейс №1		
Опис функції:	Система облікових записів користувачів		
Власник тесту:	Кривушин Юрій Олександрович		
Дата створення:	05.06.2024		
Мета тесту:	Перевірити коректність функцій реєстрації та авторизації		
Передумова			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити веб–застосунок	Користувач має доступ до сайту, який відкритий	Пройдено
2	Користувач хоче зареєструватися	Відкривається сторінка реєстрації	Пройдено
Реєстрація користувача			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити сторінку реєстрації	З’являється форма для реєстрації	Пройдено
2	Заповнити обов’язкові поля «Емейл» та «Пароль»	Всі поля заповнені коректно, кількість символів відповідає вимогам	Пройдено
3	Натиснути кнопку «Зареєструватися»	Користувача зареєстровано, відображається повідомлення про успішну реєстрацію	Пройдено
4	Користувача перенаправлено на сторінку профілю	Користувач бачить свій профіль	Пройдено

Кінець таблиці 5.1

Авторизація користувача			
№	Опис випадку	Очікуваний результат	Висновок
1	Відкрити сторінку авторизації	З’являється форма для авторизації	Пройдено
2	Ввести коректні дані для авторизації	Користувача авторизовано, перенаправлено на сторінку профілю	Пройдено
3	Ввести некоректні дані для авторизації	Відображається повідомлення про помилку авторизації	Пройдено
4	Скинути пароль через функцію «Забули пароль?»	Відображається повідомлення про успішну відправку листа для скидання паролю	Пройдено
5	Відновити пароль за посиланням з електронної пошти	Користувач вводить новий пароль, відображається повідомлення про успішне скидання паролю	Пройдено
Результати тестування			
Тестувальник: Кривушин Ю. О.	Дата прогону тесту: 05.06.2024	Результат тесту (P/F/B): ПРОЙДЕНО (P)	

ВИСНОВКИ

За результатами цієї роботи, ми провели глибокий аналіз області розробки, вивчили конкурентне середовище, і визначили ключові вимоги до нашого застосунку. Цей аналіз дозволив нам отримати краще розуміння потреб та вимог користувачів, що стало ключовим етапом перед початком проекту.

У роботі визначено важливість забезпечення легкого доступу користувачів до будь–якої частини сайту, надійності автентифікації, ефективного управління кошиком та можливості створення власних шаблонів тортів. Ці вимоги були враховані при розробці діаграм, які допоможуть в розробці нашого застосунку.

Розроблені діаграми, зокрема Use–case діаграма для клієнта та адміністратора, а також діаграма екранів, стануть цінними інструментами під час подальшої розробки. Вони не лише допомагають уявити майбутній вигляд застосунку, але й створюють чітке розуміння функціональності та взаємодії різних компонентів.

Архітектура FrontEnd частини, заснована на використанні Redux та інших сучасних технологій, обіцяє забезпечити швидку та ефективну роботу застосунку, зробивши його більш масштабованим та легким у підтримці. Підхід до проєктування структури зберігання даних також виявився обґрунтованим, забезпечуючи оптимальний рівень взаємодії між клієнтом та сервером.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Musse [Електронний ресурс] – URL: https://musse.com.ua (дата звернення: 13.05.2024)

2. Vatsak [Електронний ресурс] – URL: https://vatsak.com.ua/custom–products/ (дата звернення: 13.05.2024)

3. «Карамелька» [Електронний ресурс] – URL: https://www.карамелька.com.ua/product–category/torty–fasovannye/ (дата звернення: 13.05.2024)

4. Screen diagram [Електронний ресурс] – URL: https://documentation.softwareag.com/webmethods/compendiums/v10–7/C_Design_and_Implement_BPMs/index.html#page/design_implement_bpm_compendium/D77434.html (дата звернення: 13.05.2024)

5. Томас Т. і Харрінгтон Д. "Дизайн інтерфейсів користувача: принципи і практика". – Київ: Видавництво Інтерсервіс, 2019. – 320 с.

6. ReduxToolkit [Електронний ресурс] – URL: https://redux–toolkit.js.org (дата звернення: 13.05.2024)

7. Adobe Color [Електронний ресурс] – URL https://color.adobe.com/create/color–wheel (дата звернення 05.06.2024)

8. Розуміння Клієнт-Серверної Архітектури [Електронний ресурс] – URL https://dou.ua/forums/topic/44636/ (дата звернення 05.06.2024)

9. Роберт. С. "Чиста архітектура" – Харків: Видавництво Фабула, 2019 – 368

10. React [Електронний ресурс] – URL https://legacy.reactjs.org/docs/getting-started.html(дата звернення 05.06.2024)

11. Правильне поєднання та підбір кольорів для сайту [Електронний ресурс] – URL https://impulse-design.com.ua/ua/vybor-tsveta-dlya-sajta.html(дата звернення 05.06.2024)

12. TypeScript [Електронний ресурс] – URL https://www.typescriptlang.org(дата звернення 05.06.2024)

13. Роберт. С. "Чистий код" – Харків: Видавництво Фабула, 2019 – 448

14. Як будувати UML діаграми [Електронний ресурс] – URL https://dou.ua/forums/topic/40575/(дата звернення 05.06.2024)

15. Тестування веб-проєктів [Електронний ресурс] – URL https://qalight.ua/baza-znaniy/testuvannya-veb-proektiv-osnovni-etapi-ta-poradi/(дата звернення 05.06.2024)

ДОДАТОК А

Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ

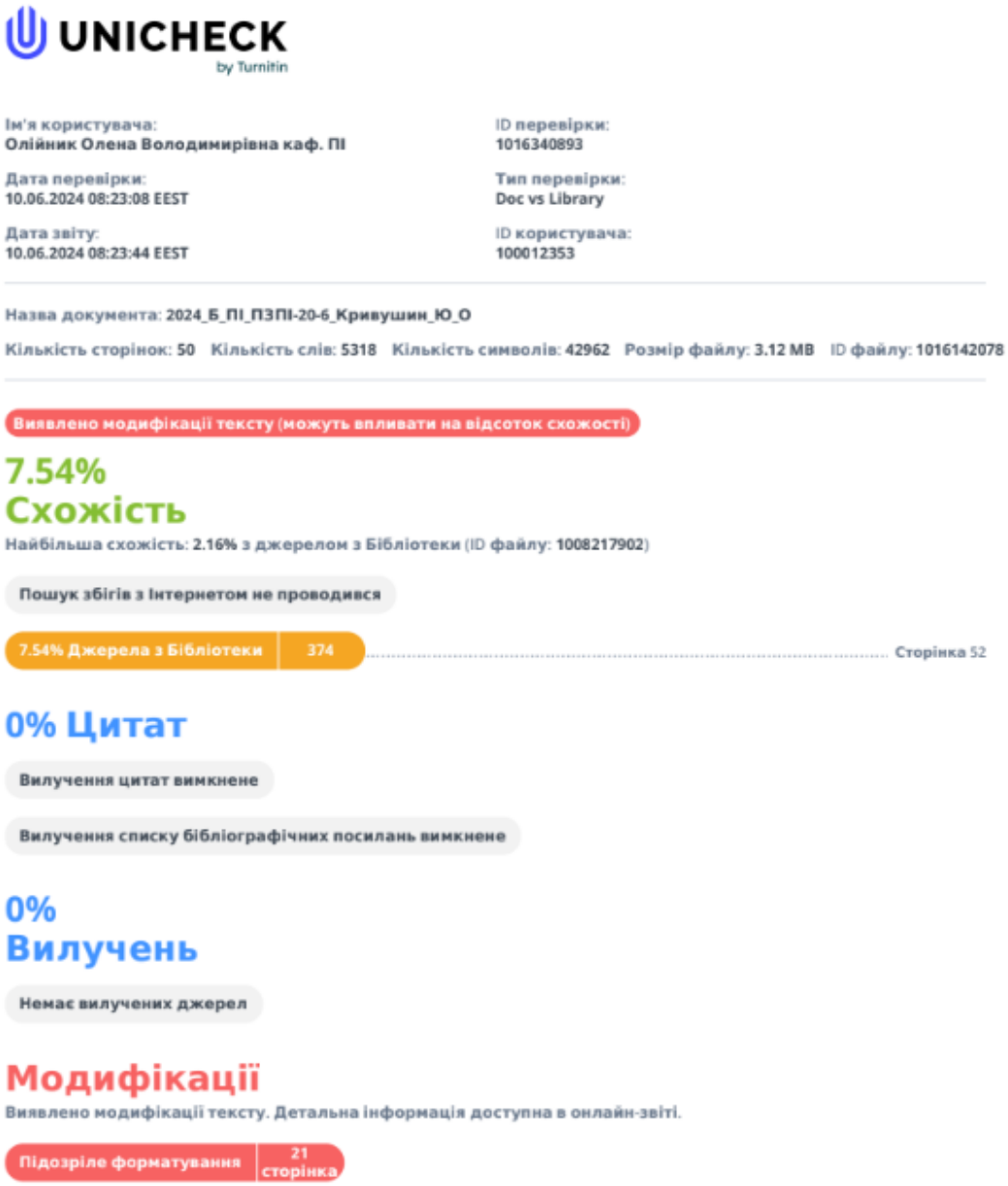

Рисунок А.1 ¬ Звіт результатів перевірки на унікальність тексту в базі ХНУРЕ (рисунок виконаний самостійно)

ДОДАТОК Б

Слайди презентації

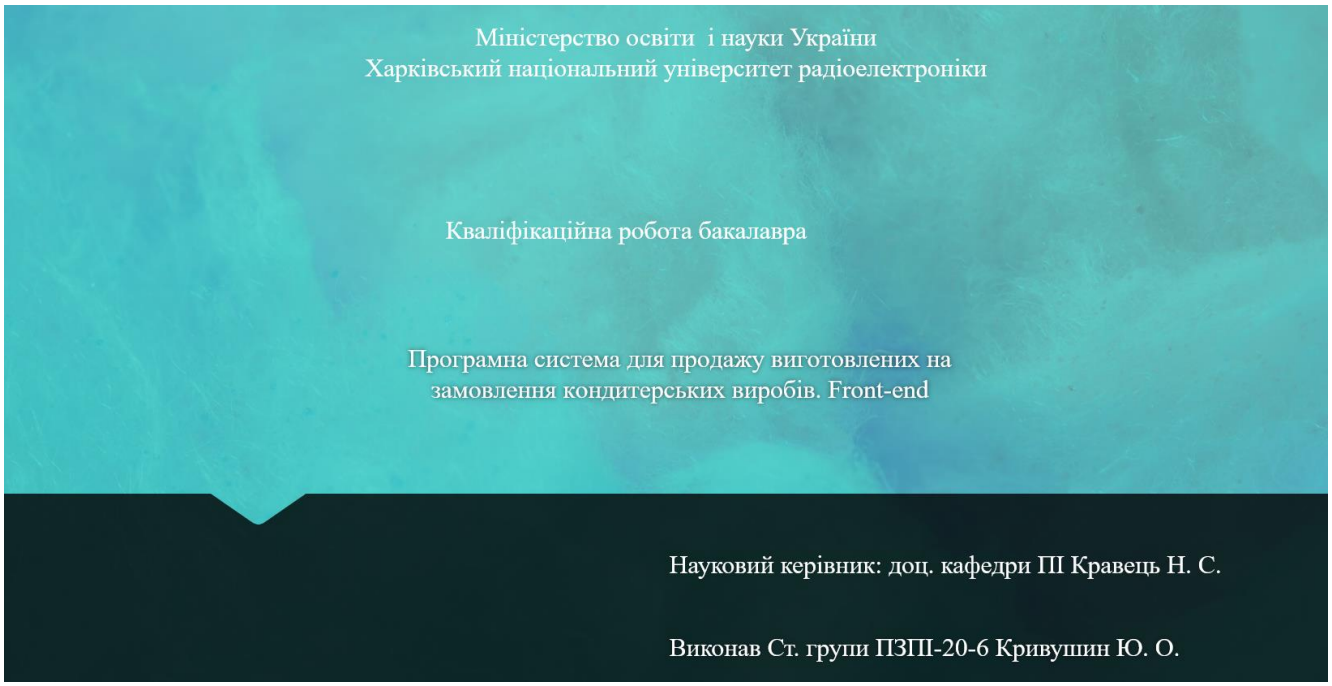

Рисунок А.1 – Титульний слайд (рисунок виконаний самостійно)

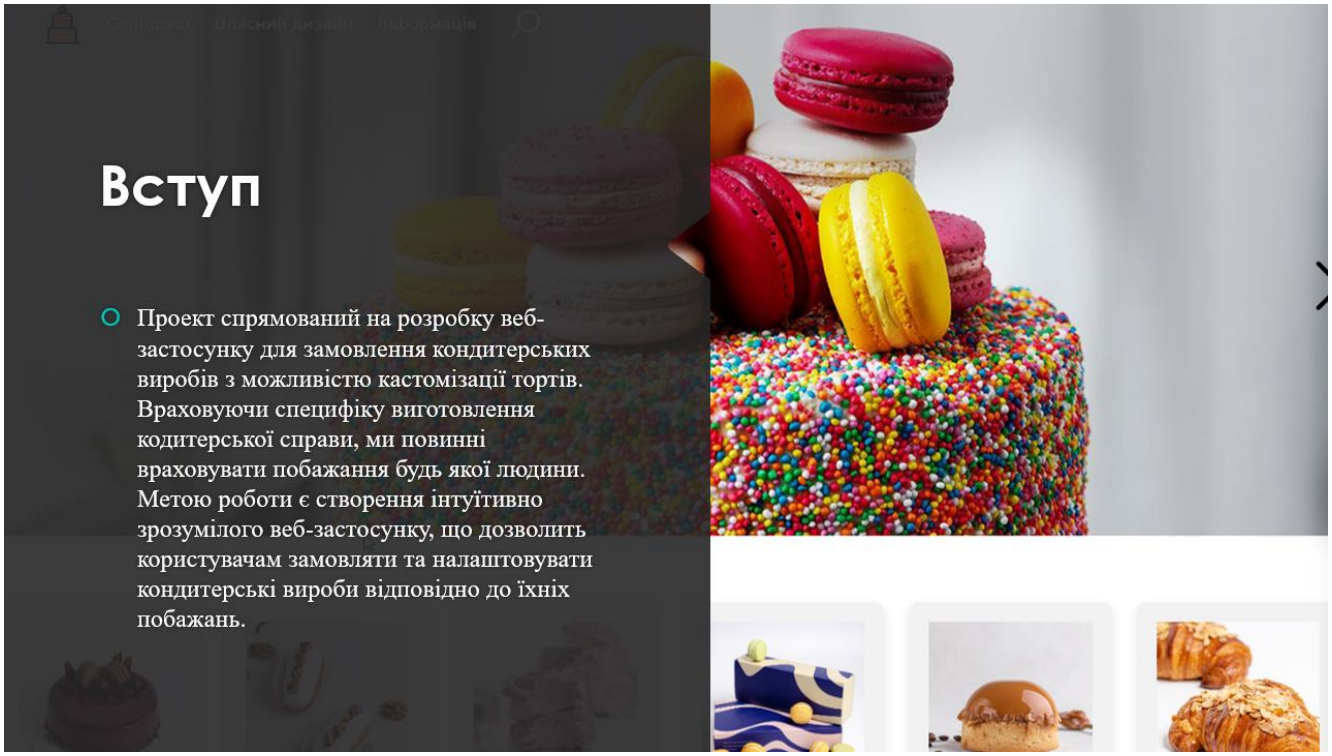

Рисунок А.2 – Слайд вступ(рисунок виконаний самостійно)

Аналіз предметної галузі

- Основними конкурентами на ринку є Musse, Vatsak, та Карамелька. Аналіз їхніх сайтів виявив значні недоліки: складні інтерфейси, погана навігація, відсутність фільтрів та локалізації. Ці проблеми створюють незручності для користувачів та знижують загальний рівень задоволення від користування сервісом.
  Також до уваги були взяті хороші рішення, що до Ui та UX дизайну.

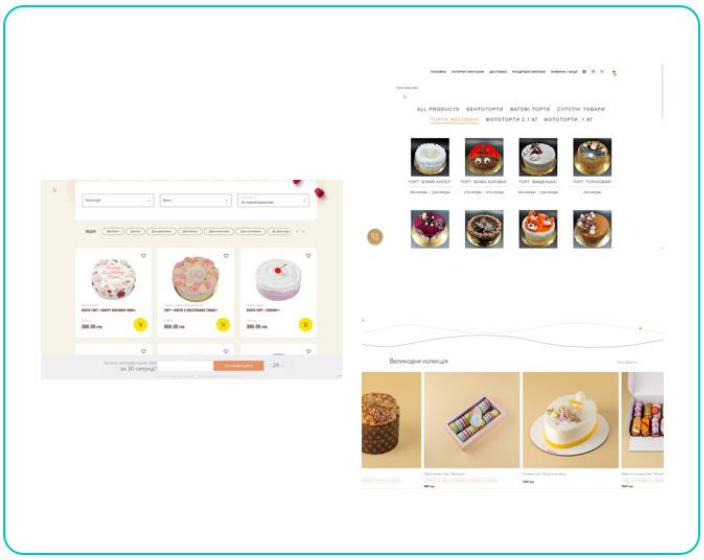

Рисунок А.3 – Слайд аніліз предметної галузі (рисунок виконаний самостійно)

Постановка задачі

- Цільова аудиторія включає любителів солодкого, організаторів подій, ділових людей та активних користувачів соціальних мереж.
- Варто розуміти свою цільову аудитрію, це аудиторія що не буде піддаватись імпульсним покупкам, але якщо людині сподобається, великий шанс що вона прийде ще раз та порекомендує нас іншим, а отже потрібно шукати підхід до різних людей індивідуально.
- Основні вимоги до системи включають інтуїтивний інтерфейс, можливість замовлення та кастомізації продуктів, реєстрацію та авторизацію користувачів, що забезпечить зручність і функціональність веб-застосунку.

Рисунок А.4 – Слайд постановка задачі (рисунок виконаний самостійно)

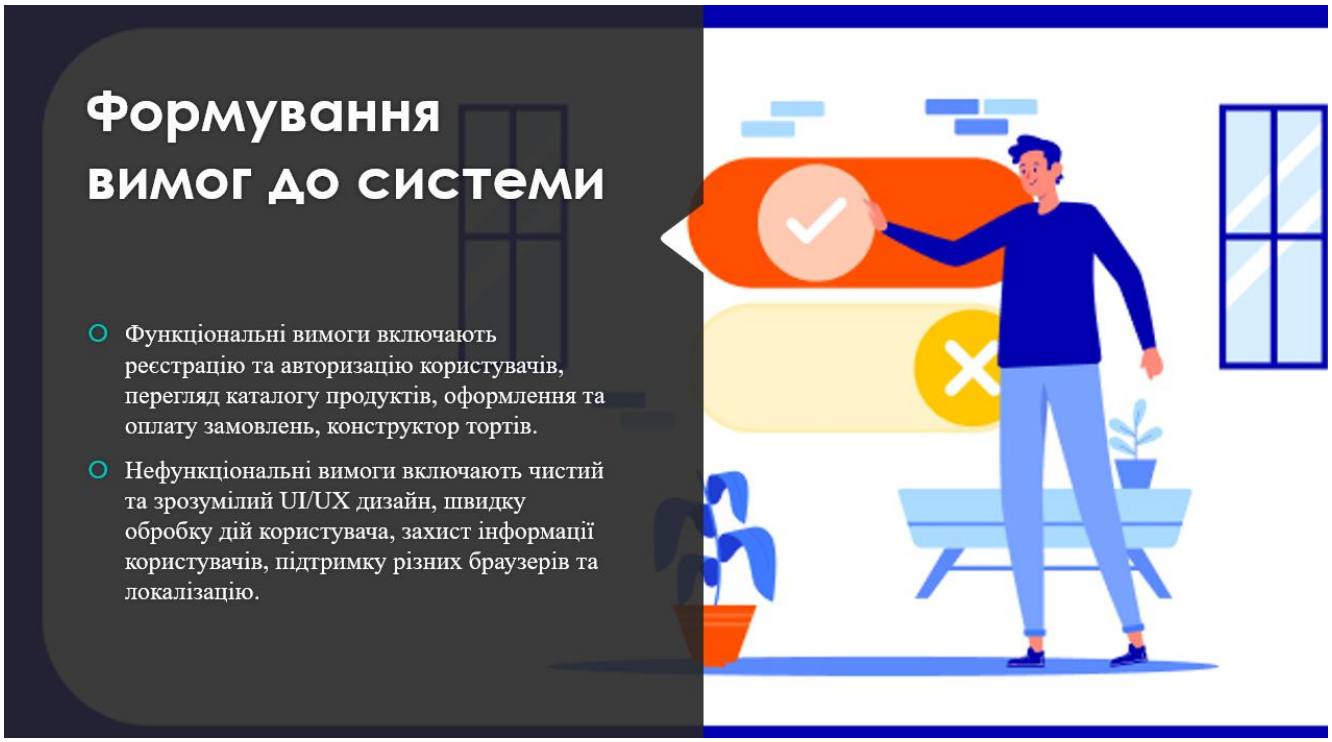

Рисунок А.5 – Слайд формування вимог до програмної системи (рисунок виконаний самостійно)

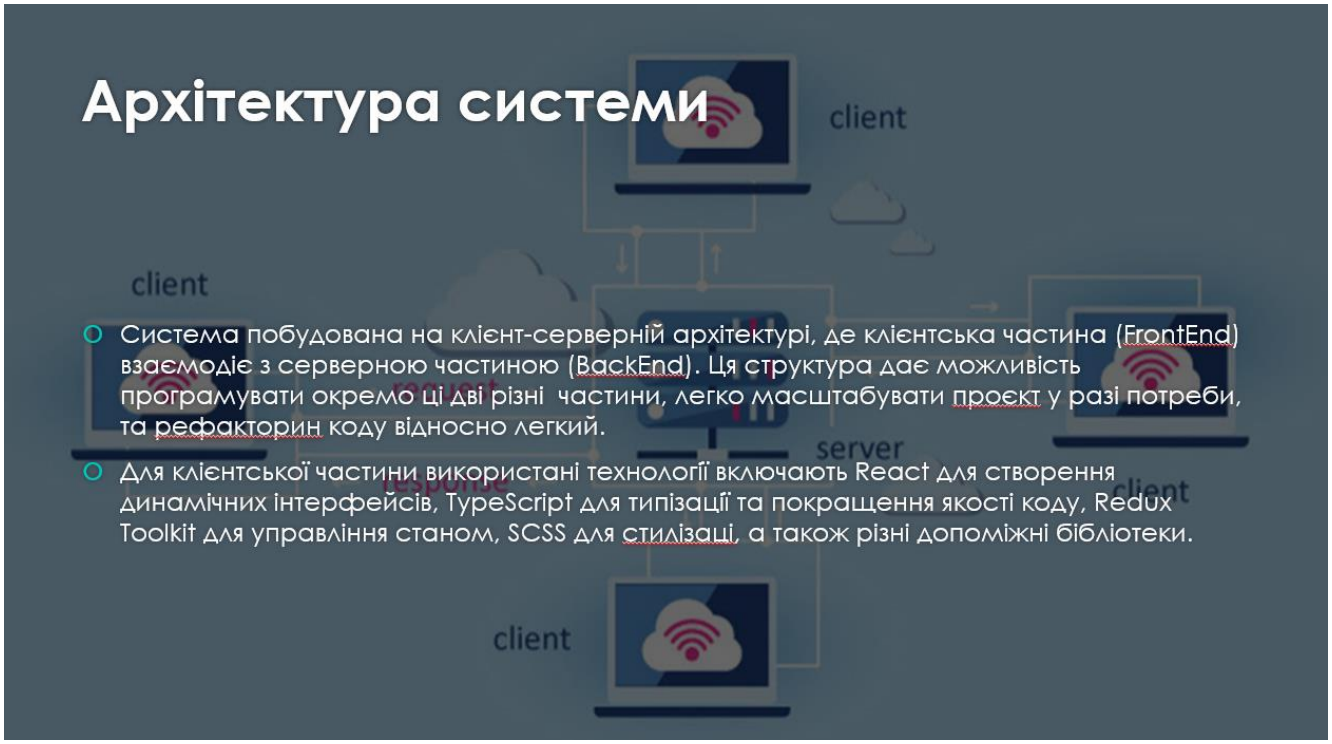

Рисунок А.6 – Слайд архітектура системи (рисунок виконаний самостійно)

Рисунок А.7 – Слайд проєктування ПЗ(рисунок виконаний самостійно)

Рисунок А.8 – Слайд розробка інтерфейсу користувача (рисунок виконаний самостійно)

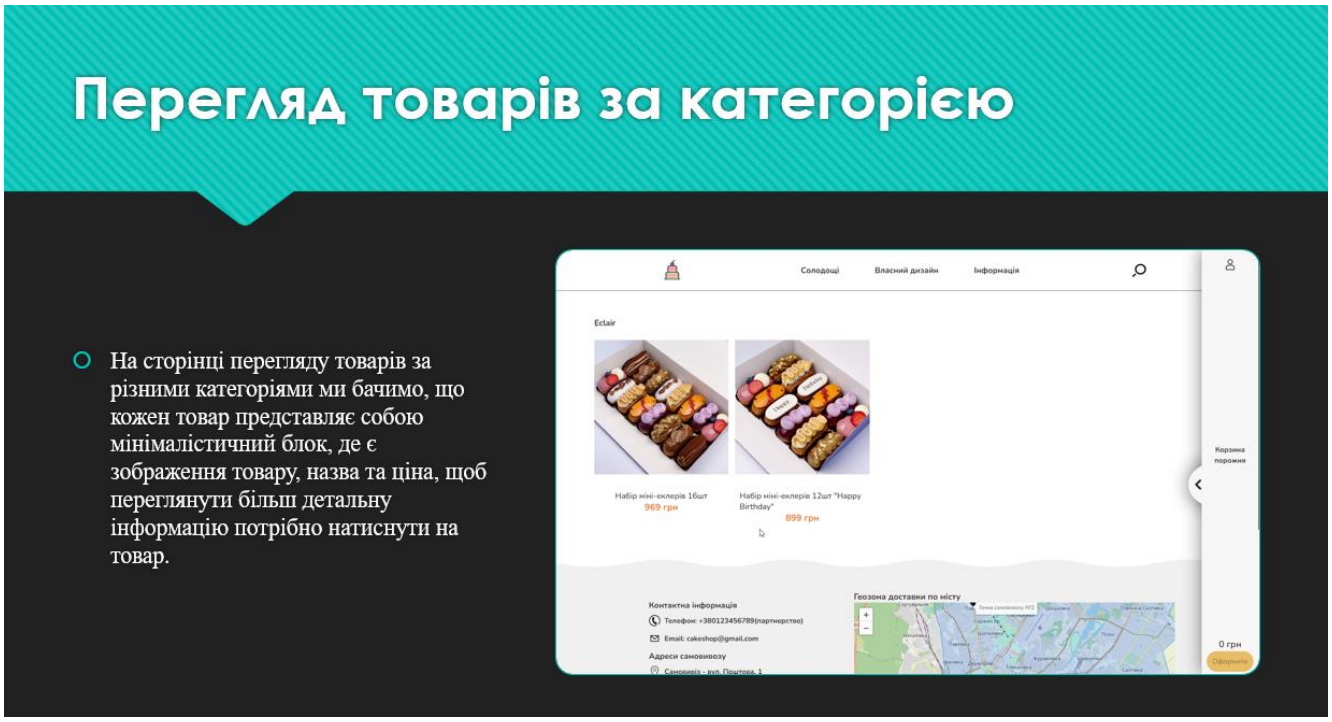

Рисунок А.9 – Слайд перегляд товарів за категорією (рисунок виконаний самостійно)

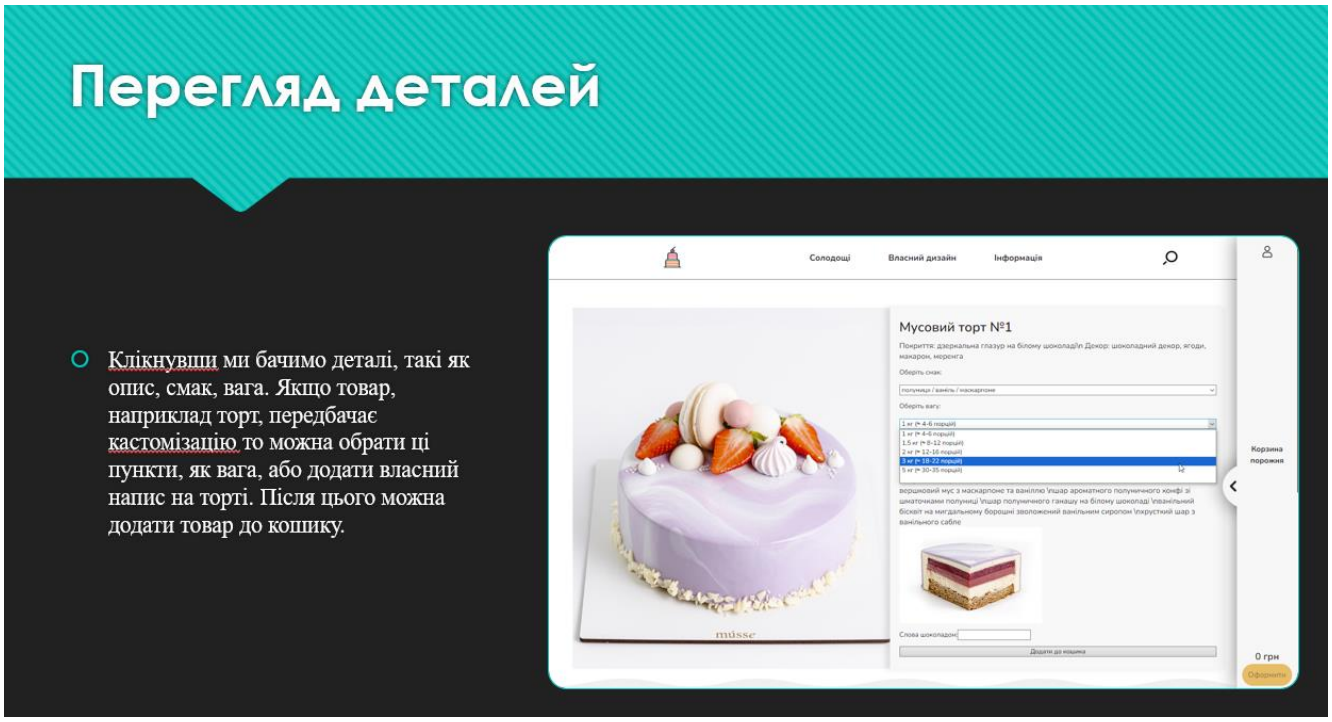

Рисунок А.10 – Слайд перегляд деталей (рисунок виконаний самостійно)

Кошик

- Кошик користувач бачить постійно, він знаходиться в згорнутому вигляді, в такому стані видно лише кількість товарів, фото та їх ціну.
- Розгорнувши можна побачити більше деталей що стосуються товарів, можна видалити продукцію з кошику або змінити кількість.

Рисунок А.11 – Слайд кошик (рисунок виконаний самостійно)

Оформлення замовлення

- При оформленні замовлення користувач повинен заповнити данні, які потрібно для доставки, а саме: особиста інформація, адреса, спосіб та час доставки.
- Після чого користувач може оформити замовлення, та зробити передоплату, в нашому випадку це важлива частина, адже продукція готується для кожного клієнту, і потрібно убезпечити себе від збитків із-за відміни замовлення.

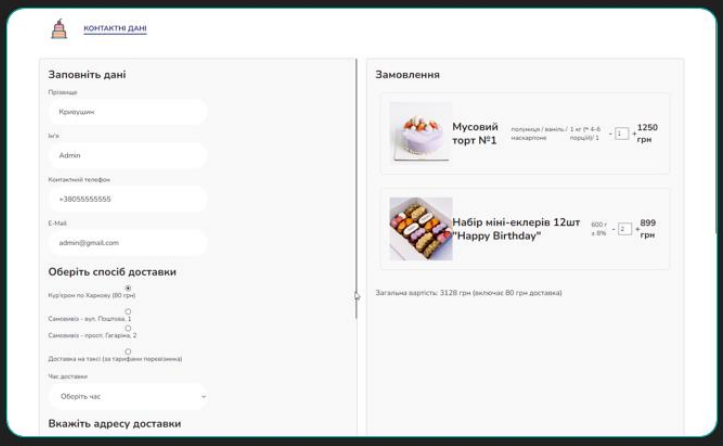

Рисунок А.12 – Слайд оформлення замовлення (рисунок виконаний самостійно)

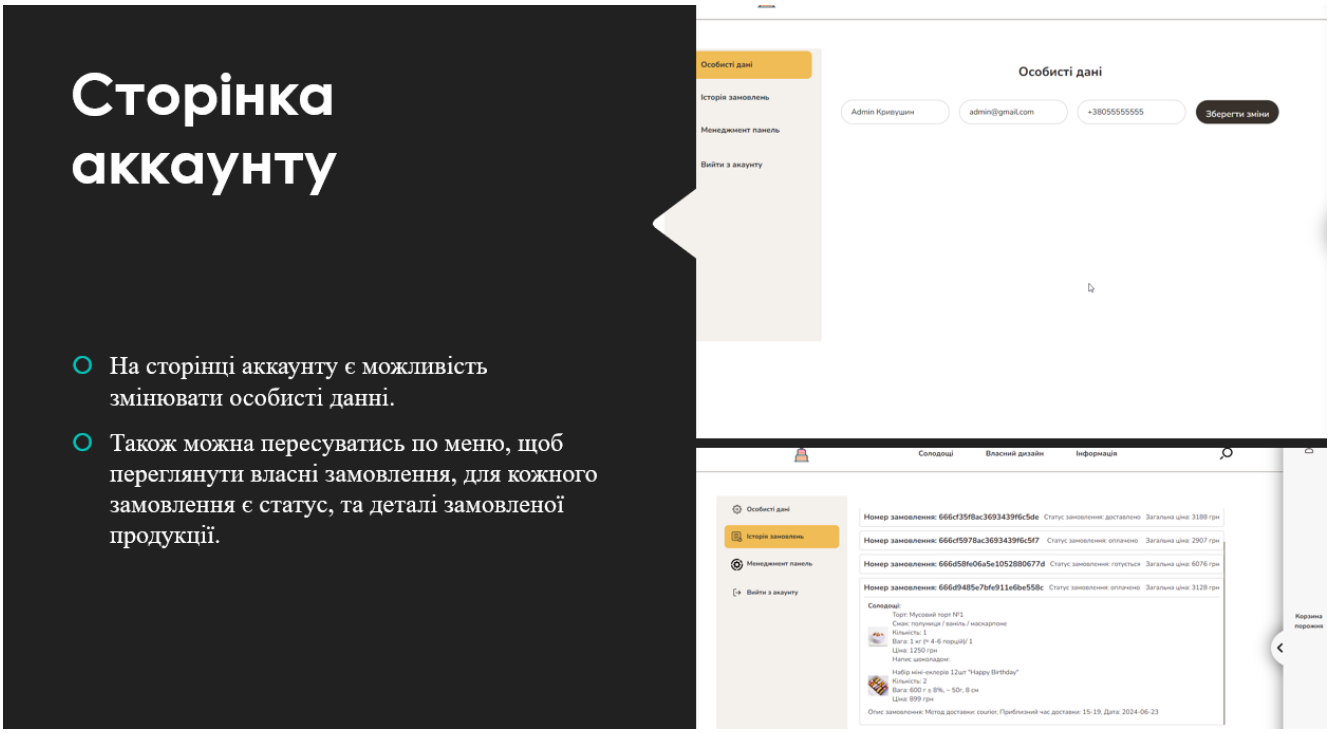

Рисунок А.13 – Слайд сторінка аккаунту (рисунок виконаний самостійно)

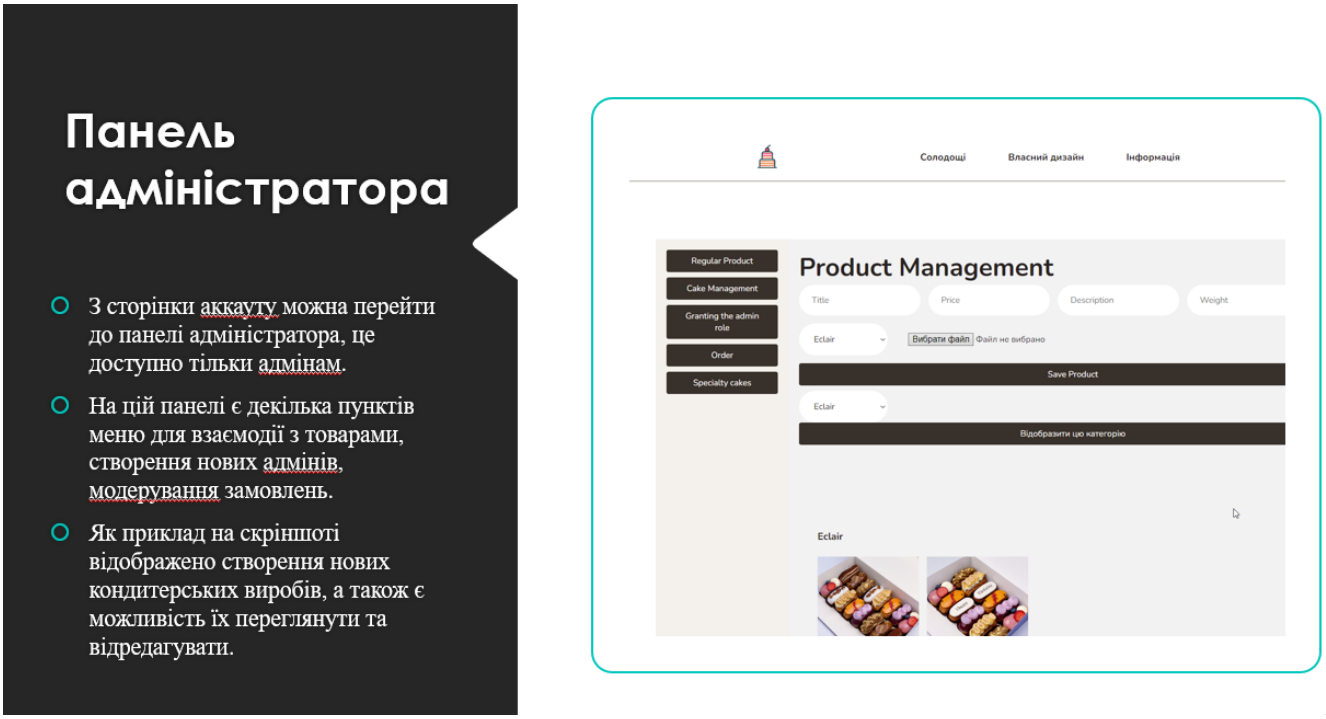

Рисунок А.14 – Слайд панель адміністратора (рисунок виконаний самостійно)

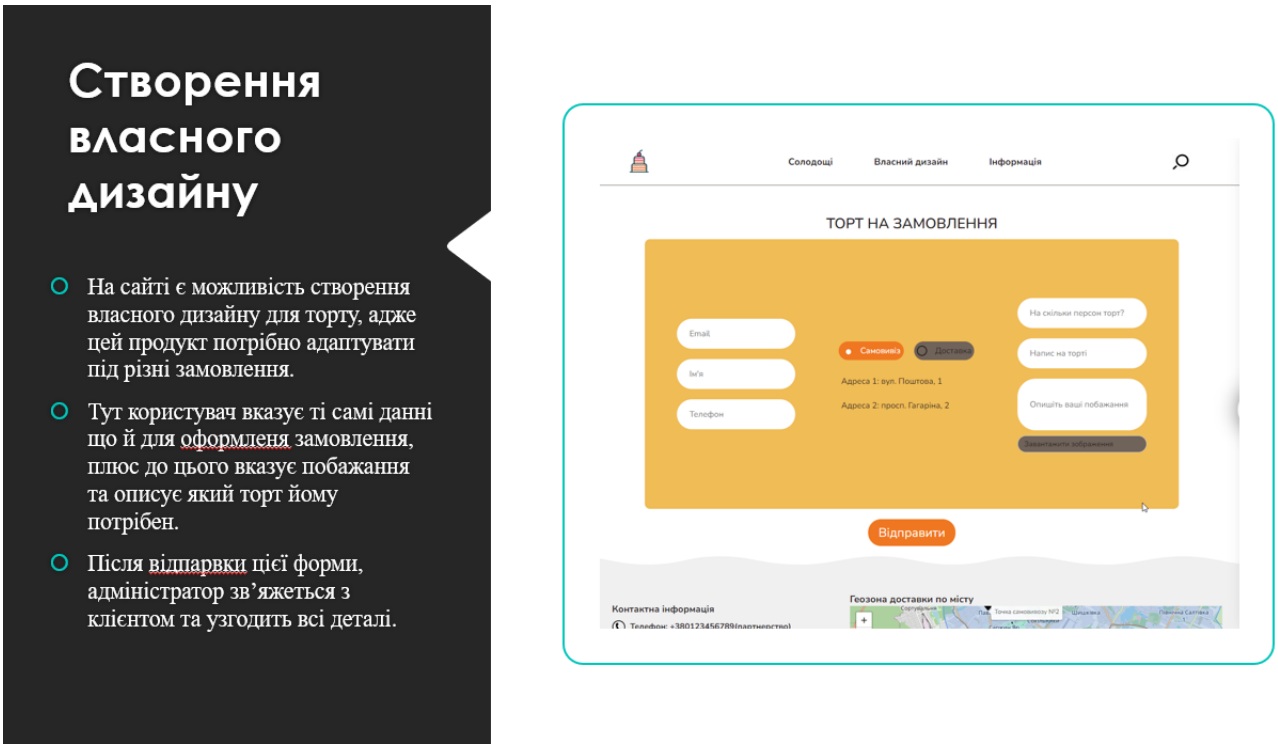

Рисунок А.15 – Слайд створення власного дизайну (рисунок виконаний самостійно)

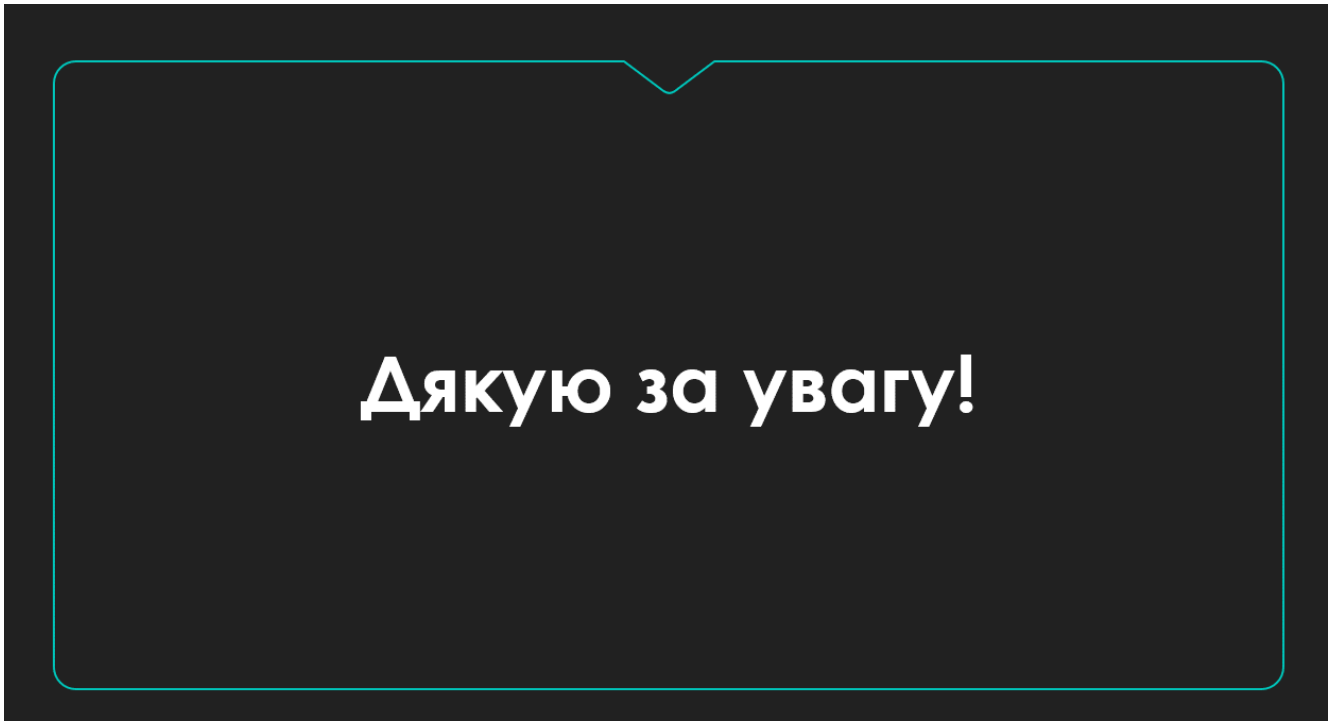

Рисунок А.16 – Фінальний слайд (рисунок виконаний самостійно)

ДОДАТОК В
Специфікація ПЗ

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет навчально-науковий центр заочної форми навчання
Кафедра програмної інженерії

СПЕЦИФІКАЦІЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
Програмна система для продажу виготовлених на
замовлення кондитерських виробів.

Студент гр. ПЗПІ-20-6 __Кривушин Ю. О.____

Харків
2024

ЗМІСТ

1 Вступ 65

1.1 Огляд проєкту 65

1.2 Мета 66

1.3 Межа 66

1.4 Посилання 67

1.5Визначення та абривіатури 67

2 Загальний опис 68

2.1 Перспективи продукту 68

2.2 Функції продукту 69

2.3 Характеристика користувачів 70

2.4 Загальні обмеження 70

2.5 Припущення і залежності 72

3 Специфікація вимог 74

3.1 Вимоги до зовнішніх інтерфейсів 74

3.1.1 Інтерфейс користувача 74

3.1.2 Програмні інтерфейси 75

3.1.3 Комунікаційний протокол 76

3.2 Атрибути програмного продукту 76

3.2.1 Надійність 77

3.2.2 Доступність 78

3.2.3 Безпека 78

3.2.4 Вимоги баз даних 79

1 ВСТУП

1.1 Огляд проєкту

Проєкт Cake Shop представляє собою комплексний веб-застосунок для замовлення тортів. Його основною метою є забезпечення зручного та інтуїтивно зрозумілого інтерфейсу для користувачів, які бажають замовити торти для різних подій, таких як дні народження, весілля або інші свята. Система дозволяє користувачам вибирати торти з каталогу готових шаблонів, а також створювати індивідуальні замовлення з урахуванням особливих побажань щодо смаку, ваги та дизайну торта.

Проєкт включає в себе як фронтенд, так і бекенд компоненти. Фронтенд частина розроблена з використанням сучасних веб-технологій, таких як React, що забезпечує динамічність та високу продуктивність інтерфейсу. Бекенд частина реалізована на Node.js та Express, що дозволяє обробляти запити користувачів, управляти даними в базі даних та взаємодіяти з зовнішніми сервісами. Зберігання даних реалізоване за допомогою MongoDB, що забезпечує масштабованість та надійність.

Основні функціональні можливості системи включають реєстрацію та авторизацію користувачів, створення та управління замовленнями, перегляд каталогу тортів, додавання тортів до кошика та оформлення замовлення. Крім того, система підтримує завантаження зображень тортів та їх подальше зберігання у хмарному сховищі Google Cloud Storage. Це дозволяє користувачам бачити візуальні приклади тортів та робити більш обґрунтований вибір.

Проєкт Cake Shop також має адміністраторську панель, яка дозволяє керувати користувачами, замовленнями та каталогом тортів. Адміністратори можуть додавати нові шаблони тортів, редагувати існуючі та видаляти непотрібні. Вони також можуть обробляти замовлення, переглядати їх статус та змінювати його у разі необхідності.

Таким чином, проєкт Cake Shop представляє собою багатофункціональну систему для зручного та ефективного управління процесом замовлення тортів, що відповідає потребам як кінцевих користувачів, так і адміністраторів системи.

1.2 Мета

Метою проєкту Cake Shop є створення зручного та ефективного веб-застосунку для замовлення тортів, який дозволяє користувачам легко обирати та замовляти торти з каталогу або створювати індивідуальні замовлення з урахуванням особистих вподобань. Проєкт також передбачає забезпечення адміністративних функцій для управління каталогом тортів, обробки замовлень та взаємодії з користувачами, що сприяє оптимізації бізнес-процесів і покращенню користувацького досвіду.

1.3 Межа

Проєкт Cake Shop має чітко визначені межі доступу та функціональності для трьох основних категорій користувачів: незареєстрованих користувачів, зареєстрованих користувачів та адміністраторів. Незареєстровані користувачі можуть вільно переглядати каталог тортів, ознайомлюватися з описами та зображеннями, що надає їм можливість побачити доступний асортимент. Однак, ці користувачі не мають доступу до можливостей додавання тортів до кошика або оформлення замовлень, що мотивує їх зареєструватися на сайті для отримання повного функціоналу.

Зареєстровані користувачі, після успішної авторизації, отримують розширений доступ до функцій системи. Вони можуть створювати замовлення, вибираючи торти з каталогу, персоналізуючи їх за смаком, вагою та додаванням тексту на торт. Крім того, зареєстровані користувачі можуть переглядати історію своїх замовлень, редагувати свій профіль та отримувати персоналізовані рекомендації. Цей рівень доступу дозволяє їм повністю використовувати можливості платформи для замовлення тортів.

Адміністратори мають найвищий рівень доступу в системі. Вони можуть керувати користувачами, включаючи блокування або видалення облікових записів, якщо це необхідно. Адміністратори також мають можливість додавати нові торти до каталогу, редагувати деталі існуючих тортів та видаляти ті, що більше не актуальні. Крім цього, вони обробляють замовлення, слідкуючи за їх статусом від створення до виконання. Це забезпечує гладку роботу системи та задоволення користувачів. Такий розподіл прав та обов'язків між різними категоріями користувачів забезпечує ефективне управління проєктом та безпеку даних.

1.4 Посилання

Усі посилання є у переліку роботи.

1.5Визначення та абривіатури

БД – База Даних
ПС – Програмна Система
API – Application Programming Interface
GCS – Google Cloud Storage
JSON – JavaScript Object Notation
HTTP – Hypertext Transfer Protocol
REST – Representational State Transfer
SEO – Search Engine Optimization

2 ЗАГАЛЬНИЙ ОПИС

2.1 Перспективи продукту

Проєкт Cake Shop має значний потенціал для розвитку та розширення, враховуючи поточні тенденції в онлайн-торгівлі та попит на персоналізовані продукти. Однією з ключових перспектив є розширення асортименту тортів, включаючи нові рецепти та дизайни, що дозволить задовольнити смаки та вподобання більшої кількості клієнтів. Це може включати розробку спеціальних тортів для різних свят, дієтичних варіантів для людей з особливими потребами в харчуванні та ексклюзивних колекцій для особливих випадків.

Іншою важливою перспективою є інтеграція з соціальними мережами та платформами для обміну зображеннями, що дозволить користувачам легко ділитися своїми замовленнями та отримувати відгуки від друзів та родичів. Це не тільки збільшить залученість користувачів, але й посприяє природному маркетингу продукту через рекомендації.

Автоматизація та оптимізація бізнес-процесів також є важливим напрямком розвитку. Впровадження системи автоматичного відстеження статусу замовлень, управління запасами інгредієнтів та інтеграція з платіжними системами забезпечить більш ефективну роботу платформи та підвищить задоволеність клієнтів. Крім того, розвиток мобільного застосунку дозволить користувачам зручно робити замовлення зі своїх смартфонів, що відповідає сучасним тенденціям мобільності та швидкості.

Перспективним напрямком є також розширення географії доставки, що дозволить обслуговувати клієнтів у нових регіонах та містах. Це може бути досягнуто через партнерство з місцевими пекарнями та службами доставки, що забезпечить швидке та якісне виконання замовлень.

Нарешті, збір та аналіз даних про вподобання клієнтів дозволить впроваджувати індивідуальні рекомендації та пропозиції, що підвищить лояльність користувачів та їх задоволення від використання платформи. Використання

технологій штучного інтелекту для аналізу замовлень та поведінки користувачів відкриває нові можливості для персоналізації та покращення сервісу.

2.2 Функції продукту

Програмна система передбачає існування декількох ролей: адміністратора, клієнта і незареєстрованого користувача. Незареєстрований користувач може створити акаунт, заповнивши форму реєстрації зі збереженням основних даних, таких як ім'я, електронна пошта, телефон і т.д.

Для зареєстрованого користувача (клієнта) передбачено клієнтську частину з функціоналом авторизації за допомогою логіна та пароля, забезпеченням безпеки даних, можливістю перегляду та зміни особистих даних, вибором продуктів з каталогу з можливістю фільтрації за категоріями та характеристиками, інтеграцією з платіжними системами для оплати замовлень, можливістю створення власних кондитерських виробів через конструктор, оформленням та відстеженням замовлень.

Адміністратор має можливість авторизуватися в системі за допомогою логіна та пароля, додавати нові позиції до каталогу кондитерських виробів з вказанням інформації про продукт, ціни та характеристики, обробляти замовлення, взаємодіяти з клієнтами щодо замовлень та управляти акаунтами користувачів, включаючи призначення статусу адміністратора.

У системі існують нефункціональні вимоги, включаючи захист особистих даних, відмовостійкість, зручний інтерфейс для користувачів та адміністраторів, оптимізація продуктивності та сумісність з різними пристроями та браузерами. Frontend реалізований на React з компонентною архітектурою, використанням Redux для управління станом, механізмами аутентифікації та обробкою помилок. Backend розроблений на Node.js з реалізацією REST API, забезпеченням безпеки даних, валідацією та підключенням до бази даних MongoDB.

2.3Характеристика користувачів

Програмна система буде цілеспрямована на різні сегменти аудиторії, включаючи приватних клієнтів, які шукають персоналізовані кондитерські вироби для особистих святкувань та подарунків. Цей сегмент включає індивідуальних покупців усіх вікових груп, зокрема молодь, що цінує нестандартні десерти, і осіб середнього віку, які організовують святкові заходи для родини та друзів.

Корпоративні клієнти також є важливим сегментом, зокрема компанії, які замовляють великі партії кондитерських виробів для корпоративних заходів та рекламних акцій. Цей сегмент включає бізнес-замовлення на свята, конференції та інші події, де важлива якість та ефектність продуктів.

До організаторів подій та кейтерингових компаній належать організатори свят та кейтерингові служби, які шукають унікальні кондитерські вироби для весільних церемоній та подій різного масштабу. Ці клієнти цінують індивідуальний підхід та можливість персоналізації продуктів.

Основні характеристики цільової аудиторії включають вікові групи від молоді до середнього віку, з високими доходами та здатністю оплачувати преміум-продукцію. Їх географічне розташування зазвичай великі міста, де є підвищений попит на унікальні та персоналізовані кондитерські вироби.Ця аудиторія активно використовує соціальні мережі для пошуку ідей та організації подій, вони цінують якість, унікальність та індивідуальний підхід у обслуговуванні.

2.4 Загальні обмеження

При розробці та впровадженні проєкту Cake Shop слід враховувати кілька важливих обмежень, які можуть вплинути на функціональність та ефективність роботи системи. Ці обмеження стосуються як технічних аспектів, так і користувацького досвіду.

Технічні обмеження: Система має обмеження на обсяг даних, що зберігаються в базі даних, а також на розмір файлів, що завантажуються користувачами (наприклад, фотографій тортів). Обмеження на обсяг оперативної

пам'яті та процесорних ресурсів на сервері може вплинути на швидкість обробки запитів та загальну продуктивність.

Обмеження доступу: Незареєстровані користувачі мають обмежений доступ до функціональності платформи. Вони можуть переглядати каталог тортів, але не мають можливості робити замовлення або залишати відгуки. Зареєстровані користувачі мають доступ до особистого кабінету, історії замовлень та можливості персоналізації тортів, але деякі функції, такі як доступ до адміністративних інструментів, для них недоступні.

Адміністративні обмеження: Адміністратори мають повний доступ до управління користувачами, замовленнями та каталогом тортів. Однак, вони обмежені в можливостях змінювати критично важливі налаштування системи, такі як конфігурація сервера або бази даних, що забезпечує безпеку та стабільність платформи.

Юридичні обмеження: Система повинна відповідати нормам законодавства щодо захисту персональних даних (наприклад, GDPR) та умовам використання платіжних систем. Це включає необхідність отримання згоди користувачів на обробку їхніх даних та забезпечення конфіденційності фінансової інформації.

Обмеження на інтерфейс: Інтерфейс користувача має бути оптимізований для роботи на різних пристроях (настільних комп'ютерах, планшетах, смартфонах). Однак, існують обмеження щодо підтримки застарілих браузерів та операційних систем, що може вплинути на доступність платформи для деяких користувачів.

Мережева інфраструктура: Швидкість та надійність інтернет-з'єднання можуть впливати на користувацький досвід, особливо при завантаженні великих зображень або здійсненні онлайн-платежів. Обмеження на швидкість мережі можуть призводити до затримок у обробці запитів та оновлення даних в реальному часі.

Розуміння та врахування цих обмежень є важливими для забезпечення стабільної та ефективної роботи платформи Cake Shop, а також для задоволення потреб користувачів та дотримання юридичних вимог.

2.5 Припущення і залежності

У розвитку і впровадженні проекту Cake Shop існують кілька ключових припущень і залежностей, які варто враховувати для успішного виконання завдань і досягнення поставлених цілей.

а) Технічні припущення:

– Хостинг і серверна інфраструктура: Платформа буде розгорнута на хостинг-провайдері Render з використанням git-репозиторію для автоматизації процесу деплою. Припущення базується на надійності і швидкодії інфраструктури Render,

– Технології: Використання Node.js для серверної частини дозволить ефективно керувати запитами та оптимізувати швидкодію системи. Фронтенд буде реалізований з використанням React.js для покращення користувацького досвіду,

б) Функціональні припущення:

– Реєстрація і авторизація користувачів: Платформа передбачає можливість реєстрації, авторизації та керування особистими даними користувачів. Це припущення базується на необхідності забезпечення конфіденційності та безпеки персональних даних,

– Інтеграція платіжних систем: Для здійснення онлайн-платежів передбачається інтеграція з платіжними системами, такими як Stripe або PayPal. Це припущення базується на забезпеченні зручного та безпечного способу оплати для користувачів,

в) Організаційні залежності:

– Залучення команди розробників: Для успішної реалізації проекту необхідно мати кваліфіковану команду розробників з досвідом у розробці веб-додатків та знанням необхідних технологій,

– Управління проектом: Ефективне управління проектом та звітність перед замовником є ключовими аспектами для забезпечення вчасної поставки продукту,

г) Юридичні залежності:

– Дотримання законодавства: Платформа повинна відповідати вимогам щодо захисту персональних даних та правилам використання інформації, що регулюються GDPR та іншими законодавчими актами.

Ці припущення і залежності є важливими для правильного планування, реалізації та успішного впровадження проекту Cake Shop.

3 СПЕЦИФІКАЦІЯ ВИМОГ

3.1 Вимоги до зовнішніх інтерфейсів

Вимоги до зовнішніх інтерфейсів системи включають інтуїтивно зрозуміле та зручне користувацьке середовище як для клієнтів, так і для адміністраторів. Клієнтський інтерфейс має бути легким у використанні і ефективно взаємодіяти з користувачем, забезпечуючи швидкий доступ до всіх функцій системи. Наявність інтуїтивних елементів керування та навігації дозволить клієнтам легко знаходити потрібну інформацію і здійснювати покупки без зайвих труднощів.

Для адміністраторського інтерфейсу важливо мати зручний доступ до управління та моніторингу замовлень, продуктів і користувачів. Інтерфейс повинен надавати інструменти для швидкої обробки замовлень, а також можливість додавати і редагувати інформацію про продукцію без зайвих складнощів.

Обидва інтерфейси повинні бути адаптивними та сумісними з різними типами пристроїв (комп'ютери, планшети, мобільні телефони), щоб забезпечити зручний доступ до системи незалежно від того, де знаходиться користувач. Також важливо, щоб інтерфейси були естетично приємними і відповідали сучасним тенденціям дизайну, що позитивно впливає на користувацький досвід і сприяє залученню клієнтів до використання системи.

3.1.1 Інтерфейс користувача

Навігаційне меню: На головній сторінці повинно бути присутнє зручне навігаційне меню, яке включає категорії продуктів: торти, еклери, випічка, татрти, тощо. Також необхідні посилання на сторінки про нас, контакти, кошик, Вхід/Реєстрація. Важливо мати динамічні зображення зі знижками та спеціальними пропозиціями, щоб привертати увагу користувачів до акційних товарів. На головній сторінці повинні бути представлені найпопулярніші товари, щоб користувачі могли швидко ознайомитися з хітами продажів.

Кожен товар має бути представлений зображенням, назвою, коротким , ціною та кнопкою «Переглянути деталі».

На сторінці кожного продукту повинна бути детальна інформація, включаючи назву продукту, детальний опис, список інгредієнтів, вагу та ціну. Поле для вибору кількості товару повинно бути доступним для зручного додавання потрібної кількості до корзини.

У кошику має бути список продуктів із зображенням, назвою, кількістю, ціною та можливістю редагування кількості або видалення товару. Відображення вартості товарів, податків та загальної суми до оплати допоможе користувачам контролювати свої витрати. Кнопка "Оформити замовлення" повинна забезпечувати можливість переходу до оформлення замовлення. Опції кур'єрської доставки та самовивозу повинні бути доступними для вибору.

Користувачі повинні мати можливість заповнити особисті дані, включаючи ім'я, прізвище, контактний номер та електронну пошту. Також повинна бути можливість вказати адресу доставки. перевірка всіх даних та підтвердження замовлення дозволить користувачам бути впевненими у правильності введеної інформації перед оформленням покупки.

3.1.2 Програмні інтерфейси

Програмні інтерфейси системи мають бути добре документовані та доступні для інтеграції з іншими системами або сервісами. Для клієнтської частини системи (Frontend) передбачається API, яке забезпечує взаємодію з сервером через HTTP-запити. Це включає отримання інформації про продукти, обробку замовлень, аутентифікацію користувачів та управління їх профілями.

Backend частина системи (Node.js) також надає API для взаємодії з базою даних та обробки бізнес-логіки. API повинно бути стабільним, ефективним та безпечним. Зокрема, важливо застосовувати стандарти безпеки для захисту від SQL-ін'єкцій, XSS-атак та інших потенційних загроз безпеці.

Документація API повинна містити чіткі описи кожного ендпоінту, формати даних, необхідні параметри запитів та очікувані формати відповідей. Це дозволяє іншим розробникам легко інтегрувати вашу систему та ефективно використовувати її можливості в їх додатках чи сервісах.

3.1.3 Комунікаційний протокол

Комунікаційний протокол системи базується на стандартах веб-комунальних технологій, зокрема HTTP і HTTPS для забезпечення безпеки та шифрування даних. Всі взаємодії між клієнтською та серверною частинами програмної системи здійснюються за допомогою RESTful API.

HTTP (Hypertext Transfer Protocol) використовується для передачі запитів і відповідей між клієнтом (веб-браузером чи іншим клієнтським додатком) та сервером. Всі запити, що ініціюються користувачами або іншими системами, адресуються через HTTP-методи GET, POST, PUT, DELETE та інші, залежно від потреби.

HTTPS (HTTP Secure) використовує TLS (Transport Layer Security) або його попередника SSL (Secure Sockets Layer) для захищеної передачі даних між клієнтом і сервером. Це забезпечує шифрування даних під час їх транспортування, що робить важливим захист особистої інформації користувачів.

REST (Representational State Transfer) є архітектурним стилем для взаємодії між компонентами програмного забезпечення. Він використовує стандартні HTTP-методи і URL для доступу до ресурсів (наприклад, замовлень, продуктів) та використовує JSON як формат даних для обміну інформацією між клієнтом і сервером.

Цей комунікаційний протокол забезпечує ефективну та безпечну взаємодію між всіма частинами системи, зменшуючи затримки і забезпечуючи стабільність роботи відповідно до вимог сучасних веб-додатків.

3.2 Атрибути програмного продукту

Атрибути програмного продукту включають різноманітні характеристики, які визначають його функціональність, якість і здатність задовольняти потреби користувачів. Основними атрибутами програмного продукту є:

Функціональність: Програмний продукт повинен забезпечувати всі функції, необхідні для задоволення вимог користувачів. Це включає функції авторизації,

управління профілем, вибір товарів, оформлення замовлень, оплату і відстеження статусу замовлень.

Ефективність: Продукт повинен працювати ефективно навіть при великому обсязі даних та великій кількості користувачів одночасно. Він повинен мати швидкий час відгуку і обробки запитів.

Надійність: Програмний продукт повинен бути стійким до відмов і збоїв. Він повинен відновлювати свою працездатність після помилок та забезпечувати високу доступність для користувачів.

Легкість використання: Інтерфейс користувача повинен бути зручним і інтуїтивно зрозумілим для всіх категорій користувачів: незареєстрованих користувачів, клієнтів і адміністраторів.

Безпека: Програмний продукт повинен захищати конфіденційні дані користувачів та інформацію про замовлення. Він повинен використовувати сучасні методи шифрування і захисту даних.

Сумісність: Продукт повинен бути сумісним з різними пристроями (комп'ютери, планшети, смартфони) і браузерами (Chrome, Firefox, Safari, Edge тощо).

Складність: Рівень складності програмного продукту повинен відповідати технічним знанням і вмінням користувачів. Він повинен бути достатньо простим для новачків і достатньо гнучким для досвідчених користувачів.

Ці атрибути допомагають визначити якість та придатність програмного продукту для використання в реальних умовах та задоволення потреб його користувачів.

3.2.1 Надійність

Надійність системи є однією з важливих характеристик, яка визначає здатність програмного забезпечення працювати без збоїв і відмов протягом тривалого часу. В нашому проєкті ця надійність забезпечується завдяки застосуванню передових технологій, ретельному архітектурному дизайну і систематичному тестуванню. Ми приділяємо особливу увагу тестуванню перед

кожним випуском нових функцій і оновлень, що дозволяє виявляти та усувати помилки на ранніх етапах розробки і після випуску продукту.

Автоматизоване тестування відіграє ключову роль у нашому процесі, прискорюючи перевірку функціональності і покращуючи загальну якість програмного забезпечення. Окрім цього, постійний моніторинг системи допомагає вчасно виявляти і усувати проблеми, що забезпечує безперебійну роботу нашої системи в умовах реального використання.

3.2.2 Доступність

Всі зображення, включаючи банери, продукти та інші графічні елементи, повинні мати альтернативний текст, що описує їх вміст. Всі тексти повинні мати достатній контраст із фоном для забезпечення читабельності.

Всі елементи списку повинні мати чітко визначені межі для екранних читачів та бути доступними для навігації з клавіатури. Кожен товар у списку повинен мати достатній опис, який буде доступний екранним читачам. Кнопки "Додати до корзини" повинні бути легко доступними з клавіатури.

Вся текстова інформація про продукт повинна бути доступною для екранних читачів. Зображення продукту повинні мати альтернативний текст. Всі поля форми повинні мати мітки для екранних читачів та бути доступними для навігації з клавіатури. Інструкції для заповнення повинні бути чіткими та зрозумілими. Поля для введення платіжної інформації повинні мати мітки для екранних читачів та бути доступними з клавіатури.

3.2.3 Безпека

Безпека є однією з найважливіших складових нашого проекту. Ми приділяємо особливу увагу захисту особистих даних користувачів, забезпечуючи їх конфіденційність і цілісність. Наш підхід до забезпечення безпеки включає декілька ключових аспектів.

По-перше, ми застосовуємо сучасні стандарти безпеки для захисту даних, включаючи шифрування і захист від несанкціонованого доступу. Всі особисті дані

передаються через захищені канали з використанням протоколів шифрування, таких як SSL/TLS, що гарантує конфіденційність під час трансмісії.

По-друге, ми ретельно валідуємо і фільтруємо всі введені дані на рівні сервера для запобігання SQL-ін'єкціям, XSS атакам та іншим видам атак на додаток. Це дозволяє уникнути введення шкідливих даних та зберегти інформацію в цілісному стані.

3.2.4 Вимоги баз даних

Вимоги до баз даних в нашому проекті включають кілька ключових аспектів, які гарантують ефективність, надійність і безпеку зберігання і обробки даних.

По-перше, ми використовуємо базу даних MongoDB для зберігання і управління даними нашої програмної системи. MongoDB вибрана через її гнучкість, швидкість і масштабованість, що дозволяє ефективно виконувати завдання зберігання та доступу до даних в умовах великого обсягу інформації.

По-друге, структура бази даних ретельно продумана для відповідності потребам нашого додатку. Ми використовуємо нормалізовану структуру даних там, де це необхідно для забезпечення цілісності і ефективного управління даними. Деякі таблиці і сутності можуть бути денормалізовані для забезпечення оптимальної продуктивності під час операцій зчитування.

По-третє, забезпечення безпеки даних є невід'ємною частиною нашого підходу до роботи з базами даних. Ми використовуємо механізми шифрування для захисту конфіденційної інформації під час зберігання та передачі даних, а також застосовуємо механізми контролю доступу для обмеження прав доступу до бази даних.

Додатково, для забезпечення відмовостійкості системи, ми реалізуємо резервне копіювання і відновлення даних, що дозволяє відновлювати інформацію в разі випадкового видалення або системних збоїв.

Всі ці вимоги до баз даних спрямовані на створення надійного, безпечного і ефективного середовища для зберігання і обробки даних користувачів нашого проекту.