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 «Інформаційне забезпечення робототехнічних систем»
спеціальності 126 «Інформаційні системи та технології»
на тему: «Вебзастосунок інтернет магазину сирів»

Виконав :
студент IV курсу, групи ІК-91,
Дацький Микита Владиславович _____

Керівник:
Керівник д.т.н., доцент кафедри ІСТ,
Поліщук Михайло Миколайович _____

Рецензент:
Старший викладач кафедри ІПІ,
Марченко Олена Іванівна _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент_____

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 «Інформаційне забезпечення робототехнічних систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту

Дацькому Микиті Владиславовичу

1. Тема проєкту «Вебзастосунок інтернет магазину сирів», керівник проєкту Поліщук Михайло Миколайович, доцент, 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: структура вебзастосунку та вимоги для автоматизації та управління магазину сирів.
4. Зміст пояснювальної записки:
    1. Аналіз вимог
    2. Вибір технології
    3. Розробка ПЗ
    4. Тестування
5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): Фізична модель бази даних (А3), Схема архітектури застосунку (А3), Діаграма авторизації BPMN (А3), Діаграма налаштувань BPMN (А3)
6. Дата видачі завдання 27 лютого 2023 року

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Ознайомлення з завданням	02.05.23 – 08.05.23	
2	Аналіз предметної області	09.05.23 – 15.05.23	
3	Аналіз існуючих рішень	16.05.23 – 22.05.23	
4	Проектування моделі	23.05.23 – 29.05.23	
5	Розробка програмного забезпечення	30.05.23 – 05.06.23	
6	Оформлення документації	05.06.23 – 12.06.23	

Студент Микита ДАЦЬКИЙ

Керівник Михайло ПОЛІЩУК

АНОТАЦІЯ

Пояснювальна записка проекту складається з чотирьох розділів і містить 18 таблиць та 19 джерел - всього 74 сторінок.

Дана дипломна робота присвячена розробці веб-додатку. Мета проекту – метою даної дипломної роботи є розробка та впровадження функціональних можливостей, необхідних для ефективного управління інтернет-магазином сиру. Основні завдання включають розробку користувальницьких інтерфейсів замовлення, оплати та адміністративні функції для керування продуктами, замовленнями та клієнтами. Також сформовано основні алгоритми роботи системи, тестування продуктивності веб-додатків, моделювання та проектування програмного забезпечення, аналізу та вибору засобів розробки.

Структурно-архітектурні діаграми та ER-діаграма сутностей бази даних. Також у складі роботи веб-додаток тестується на працездатність і надійність. Детальне тестування дозволяє виявити та усунути проблеми, які можуть виникнути під час роботи системи. В результаті досліджень і розробок, розроблено основні алгоритми системи. створено систему, яка забезпечує ефективне управління та функціонування інтернет-магазину сиру. Загалом ця робота дозволить реалізувати веб-додаток для інтернет-магазину сиру з використанням сучасних технологій розробки, що забезпечує зручний та функціональний інтерфейс для користувачів, а також ефективне управління та контроль системи.

Ключові слова: адміністрування, база даних, безпека, інтернет-магазин, сервер, веб-додаток, веб-технології.

ANOTATION

The explanatory memorandum of the project consists of four sections and contains 18 tablesand 19 sources - a total of 74 pages.

This thesis is devoted to the development of a web application. The purpose of the project - the purpose of this thesis is to develop and implementfunctional capabilities necessary for effective management of an online cheese store.

The main tasks include the development of user interfacesordering, payment, and administrative functions to manage products, orders, and customers. The main algorithms of system operation, web application performance testing, software modeling and design, analysis and selection of development tools were also formed.

Structural and architectural diagrams and ER diagram of database entities. Also in as part of the work, the web application is tested to check its operability and reliability. Detailed testing allows you to identify and eliminate problems that may arise during system operation. As a result of research and development, the main algorithms of the system have been developed. a system has been created that ensures effective management and functioning of the online cheese store. In general, this work will implement a web application for an on-line cheese store using modern development technologies, providing a convenient and func-tional interface for users, as well as effective management and control of the system.

Keywords: administration, database, internet store, security, server, web application, web technologies.

Номер рядка	Формат	Позначення	Найменування	Кільк. аркушів	Номер екзем.	Примітка
1			Документація загальна			
2						
3			Знову розроблена			
4						
5	А4	ІК91.150БАК.005 ПЗ	Пояснювальна записка	77		
6	А3	ІК91.150БАК.005 Д1	Вебзастосунок інтернет мага-	1		
7			зину сирів.Фізична модель			
8			БД			
9	А3	ІК91.150БАК.005 Д2	Вебзастосунок інтернет мага-			
10			зину сирів. Діаграма автори-	1		
11			зації BPMN			
12	А3	ІК91.150БАК.005 Д3	Вебзастосунок інтернет мага-	1		
13			зину сирів. Діаграма налаш-			
14			тувань BPMN			
15	А3	ІК91.150БАК.005 Д4	Вебзастосунок інтернет мага-	1		
16			зину сирів. Діаграма архітек-			
17			тури застосунку			
18						
19						
20						
21						
22						
23						
24						
25						
26						
27						
28						

Зм.	Аркуш	№ докум.	Підпис	Дата	ІК91.150БАК.005 ТП		
Розроб.		Дацький М.В.			Вебзастосунок інтернет магазину сирів. Відомість проєкту	Літ.	Аркуш 1 / Аркушів 1
Керівн.		Поліщук М.М.				Т	
Затв.					КПІ ім. Ігоря Сікорського Група ІК-91		

Пояснювальна записка
до дипломного проєкту
на тему: «Вебзастосунок інтернет магазину сирів»

Київ – 2023 року

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

WebStorm IDE – Integrated Development Environment – інтегроване середовище розробки.

ER – Entity-Relation diagram

ОС – Операційна система.

БД – База даних.

NodeJs – Платформа для виконання високопродуктивних мережевих застосунків, написаних мовою JavaScript.

REST – Підхід до архітектури мережевих протоколів, які надають доступ до інформаційних ресурсів.

API – Прикладни́й програ́мний інтерфе́йс (інтерфейс програмування застосунків).

PUG – шаблонізатор HTML для Node.js та браузерного середовища.

PK – Primary Key, Первинний ключ в базі даних

FK – Foreign key, Зовнішній ключ в базі даних

SSR – Server Side Rendering, Візуалізація на стороні сервера

ВСТУП

Зараз, коли кожен бізнес повинен мати свій сайт, коли соціальних мереж щоб рекламувати продукти вже не достатньо інтернет магазини стали не просто модним аксесуаром, а є невід'ємною частиною бізнесу в сучасному цифровому світі, де з’являється все більше веб-застосунків з різноманітними функціями та різним рівнем складності.

Більшість з них використовуються для онлайн присутності компаній та електронної комерції, також як інформаційний блог, для брендування та довіри. Розробка інтернет магазину є важливим етапом для багатьох компаній, які бажають розвивати свій бізнес, хочуть забезпечити збільшення кількості клієнтів через онлайн простір. Важливо, що сайт має мати певну функціональність для обробки замовлень, оформлення замовлення та ін. Однак, один із самих головних аспектів окрім функціональності та дизайну є безпека, конфіденційність даних клієнтів, безпечні операції з оплатою та захист від інших атак зловмисників. Але окрім написання сайту важливим етапом є вибір архітектури проекту, яка в майбутньому при різкому розширенні бізнесу, компанія матиме можливість швидкого масштабування без перебудови проекту, тобто розширення проекту на мобільні додатки, програми до смарт годинників та інше.

Врахування мобільної сумісності та пошукової оптимізації також є важливими аспектами при розробці інтернет-магазину. Програмування веб-сторінок, що адаптуються до різних пристроїв та оптимізуються для пошукових систем, сприятиме підвищенню видимості магазину та привертанню більшої аудиторії потенційних клієнтів. Це дозволить забезпечити зручний та ефективний досвід користувачів, незалежно від пристрою, який вони використовують для здійснення покупок. Крім того, з метою підтримки інтерактивності та залучення користувачів до інтернет-магазину, розробка додаткових функціональних можливостей, таких як відгуки клієнтів, особистий кабінет, рекомендації товарів та система знижок, може значно підвищити привабливість та конкурентоспроможність платформи.

1 АНАЛІЗ ВИМОГ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

1.1 Загальні положення

Веб-застосунок – це програма, яка працює на сервері і доступна для використання через веб-браузер. Програмне забезпечення яке виконує певні функції або надає певні послуги користувачам через інтернет. Компанії якраз і потребують такий інструмент, який допоможе ефективно та швидко вирішувати поставлену задачу.

Загалом, за останній час з’явилась велика кількість команд і організацій які займаються розробкою. Завдяки постійному розвитку технологій, конкуренція зростає і інтернет магазини стають все більш складними та функціональними. Тому таким компаніям потрібен веб застосунок, з можливістю майбутнього безпроблемного масштабування проекту.

1.2 Змістовний опис і аналіз предметної області

Веб-додатки — це програмне забезпечення або програма, яку можна відкрити за допомогою будь-якого браузера. Перевагами є те, що вони пропонують низку функцій, частина з яких є необхідними для взаємодії користувача та компанії. Проаналізувавши відгуки компанії яка продає сири, стало зрозуміло, що сервіси які дають змогу продавати свою продукцію не підходять для продажу товарів через свою неефективність та недостатню кількість функціоналу для взаємодії з клієнтами. У цьому випадку веб-застосунок буде найкращим рішенням для вирішення таких задач. В результаті він допоможе компанії покращити свою ефективність.

Однією з головних проблем розробки програмного забезпечення для браузерів є велика варіативність та різноманітна складність функцій. Ця проблема веде до збільшення часу розробки та складності проекту. А саме, розробка автоматизації відділу продаж та всіх заявок , ресурсів що надходять в компанію. Це дозволить зменшити ручну працю, покращити точність та швидкість обробки.

Іншою проблемою є різноманітність ринку таких застосунків, де кожна компанія хоче виділитись та додати якусь нову функцією для взаємодії з клієнтом, наприклад чат боти та інше. Розробникам необхідно враховувати особливості та обмеження які накладає на них компанія, необхідні вимоги та основні задачі.

Через велику кількість інструментів для розробки додатків одразу для декількох платформ, у цій роботі було обрано шлях розробки додатку тільки для однієї платформи – для браузерів. Також однією з цілей цього дипломногопроекту є створення додатку який зможе в майбутньому легко масштабуватись. Для цього буде розроблена спеціальна архітектура з урахуванням всіх задач додатку.

1.3 Аналіз існуючих технологій та успішних IT-проектів.

Було проаналізовано аналогічні застосунки, які можуть бути використанні для продажу продукції та взаємодії з клієнтами. Також розглянуті допоміжні засоби розробки та готові рішення.

1.3.1 Аналіз допоміжних програмних засобів та засобів розробки.

Одним із багатьох доступних варіантів IDE для розробки застосунків такого типу є WebStorm, Visual Studio та інші. WebStorm – це інтегроване середовище для розробки на JavaScript та пов'язаних з ним технологіях. Як і інші IDE JetBrains, WebStorm дозволяє автоматизувати рутинну роботу та легко справлятися зі складними завданнями. Один з головних плюсів IDE у тому, що вони поєднують усі необхідні інструменти. Використовуючи WebStorm для налагодження та тестування клієнтського коду та програм на Node.js, а також для роботи з системою контролю версій, такими як Git, дозволяє зручно відстежувати та керувати версіями коду, спрощуючи процес спільної роботи над проектом у команді. Крім того, можливості налагоджування та тестування, включаючи підтримку віддаленого налагодження та юніт-тестування

Є можливість користуватись інтегрованими лінтерами, інструментами збирання, терміналом та HTTP-клієнтом.

Існує досить багато мов програмування, за допомогою яких можна розробляти додатки, а саме Python , Java, PHP, Ruby, C# , Go, NodeJs . Мова JavaScript та програмна платформа NodeJs використовується з 2009 року для розробки додатків і все ще підтримується компанією. Звичайно, останнім часом все більш компаній використовують мову Python, яка є досить популярною мовою зі своїми особливими фреймворками для розробки додатків. Таким чином, можна створювати програми за допомогою кожної з мов, однак у поєднанні з середовищем виконання Node.js JavaScript може досягти високої швидкості на стороні сервера. Він відомий своїми керованими подіями, неблокуючою моделлю введення-виведення, що робить його підходящою мовою для побудови ефективних та здатних до масштабування веб-додатків.

Для розробки клієнтської сторони існує два популярних підходи – Client-Side rendering та Server-Side rendering. Та велика кількість фреймворків до кожного з цих підходів, для цього проекту я обрав підхід Server-Side Rendering (SSR) через його швидкість розробки. Також перевагами SSR є покращений час завантаження початкової сторінки, тому що він може негайно відобразити контент і не потрібно чекати завантаження та виконання JS перед відображенням сторінки. Також на вибір вплинула пошукова оптимізація, покращена оптимізація на слабких пристроях та простіша реалізація певних функцій порівняно з Client-Side Rendering (CSR).

CSR має і свої переваги як в розробці так і в кінцевому результаті, а саме розширена інтерактивність, динамічний вміст сторінок, гнучкість та відокремлення модулів під час розробки. Плюсом буде знижене навантаження на сервер, особливо важливо якщо програма буде обробляти мільйони запитів.

Ось деякі приклади фреймворків Client-Side Rendering:

– Angular.js;

– Vue.js;

– Ember.js;

– React.js;

– Svelte.js.

Зараз існує також широкий спектр серверних фреймворків на вибір, зокрема:

– Next.js;

– Nuxt.js;

– Nest.js.

Відтворення як на стороні клієнта, так і на стороні сервера має багато переваг і недоліків. Для прискорення розробки я вирішив використати Pug, це популярний механізм для створення шаблонів, який використовується для рендеренгу на сервері (SSR). Він доволі оптимізований та добре працює у зв’язці з Express, фреймворком для Node.js.

Він забезпечує ефективний спосіб генерувати динамічний HTML на сервері та доставляти попередньо відрендерений вміст клієнту, покращуючи продуктивність, SEO та загальну взаємодію з користувачем.

1.3.2 Аналіз відомих програмних продуктів

Для таких типів веб застосунків доступно багато різних варіацій, кожна з яких має свої унікальні функції та особливості. Наведемо деякі популярні сайти для продажу крафтової сирної продукції та їх особливості.

The Henri Willig є однією з популярних європейських компаній, яка спеціалізується на виготовленні та продажу продукції. Вони мають широкий асортимент сирів власного виробництва.

Інтернет-магазин The Henri Willig пропонує необхідний стандартний набір функціоналу, як для інтернет-магазину, який дозволяє зручно та легко здійснювати покупки онлайн. Також, інтернет-магазин The Henri Willig пропонує різні способи оплати, щоб задовольнити різні потреби клієнтів. Клієнти можуть переглядати різні види сирів, ознайомлюватись з їхніми характеристиками та відгуками, додавати продукти до кошика і оформляти замовлення:

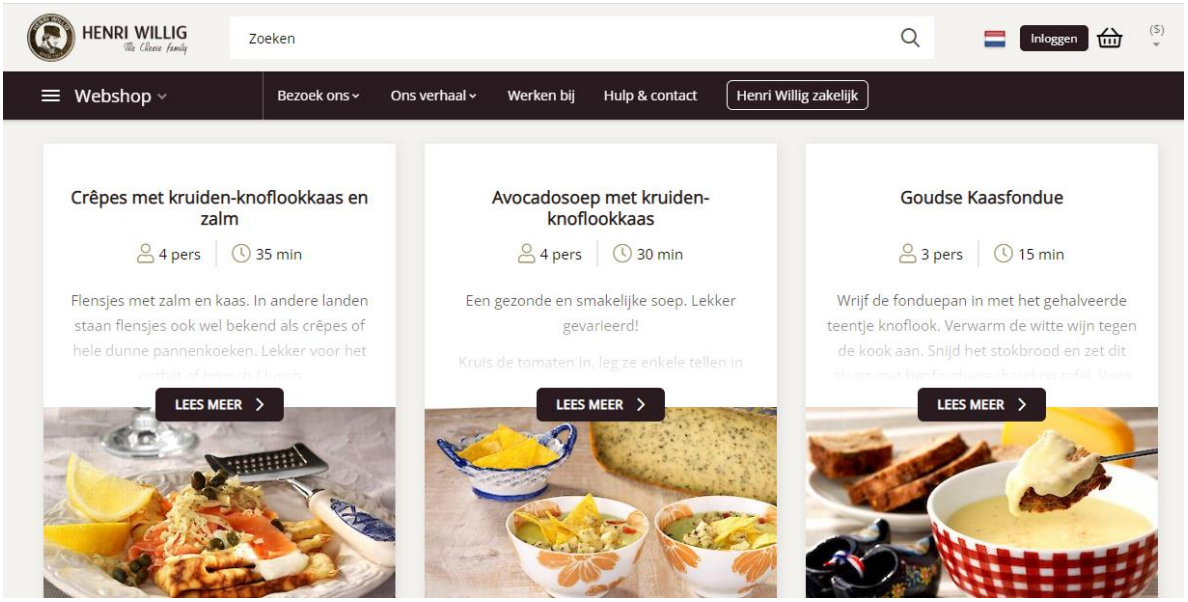

Рисунок 1.1 – Інтерфейс веб-застосунку Henri Willing

До переваг ПЗ можна віднести:

– зрозуміла навігація в широкому ассортименті продукції, ПЗ надає користувачам зручний та легкий спосіб орієнтуватися в широкому асортименті продукції. Користувачі можуть швидко знайти потрібний товар або послугу завдяки чіткій структурі та логічному розміщенню інформації;

– різні типи ролей при реєстрації , від клієнта до реселера, ПЗ надає можливість реєстрації в системі з різними типами ролей, що дозволяє користувачам вибрати підходящу роль відповідно до їхніх потреб, наприклад, від клієнта, який здійснює покупки, до реселера, який продає товари далі;

– велика кількість різних можливостей для оплати;

– система відгуків та інформаційні канали;

До недоліків ПЗ відноситься:

– відсутність мобільного додатку з доставкою;

– немає адміністративної панелі для бізнесу з необхідними метриками та візуально приємними графіками для оцінки місячних результатів;

– через велику кількість різних підменю важко знайти потрібну інформацію, якщо ПЗ має складну структуру з багатьма підменю та категоріями, то користувачам може бути важко знайти потрібну інформацію;

Antonelli’s Cheese Shop – популярний веб-застосунок з Техасу та незвичайним функціоналом, має меню спеціальних можливостей, в якому можна прочитати текст, збільшити шрифт та змінити контраст та вибрати будь-яку мову. Також веб-застосунок має свого власного чат бота який жопоможе відповісти на базові питання.

Головну сторінку додатку можна побачити на рисунку 1.2

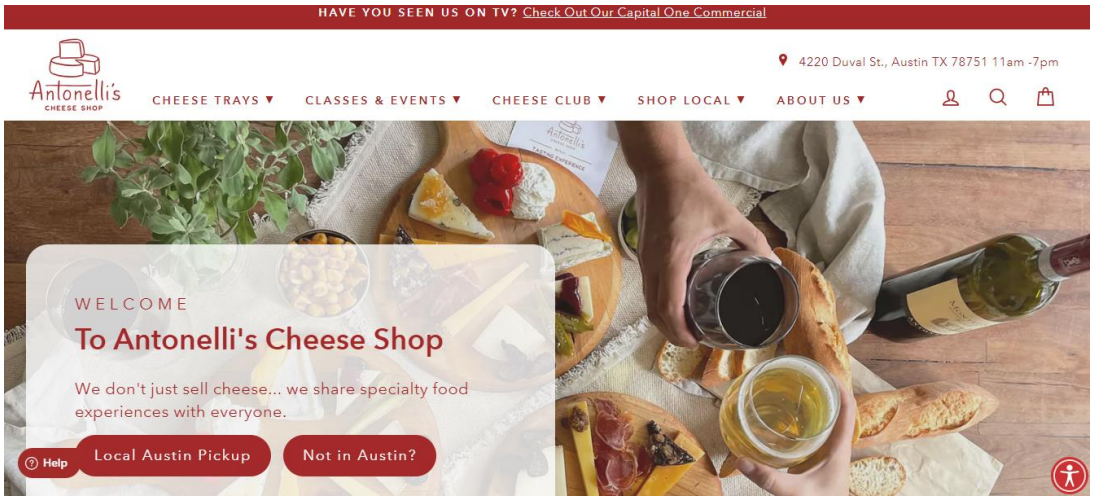

Рисунок 1.2 – Сторінка додатку Antonelli’s Cheese Shop

До переваг ПЗ можна віднести:

– наявність свого власного чат бота, який надає користувачам зручний спосіб отримати швидку та автоматизовану підтримку. Чат-бот може відповідати на запитання користувачів, надавати рекомендації та допомогу у вирішенні проблем;

– меню спеціальних можливостей;

– привабливий дизайн;

До недоліків ПЗ відноситься:

– через велику кількість плаваючих меню, викликає проблеми на слабких пристроях, де обробка та відображення багатьох елементів може бути повільним та незручним. Важливо забезпечити оптимізацію та адаптацію ПЗ під різні типи пристроїв;

– немає можливості заплатити криптовалютою, обмежує гнучкість і можливості користувачів, які бажають використовувати цей спосіб оплати.

EuroSir – популярний інтернет-магазин в Україні з приємним дизайном, без великої кількості підменю яке може заважати в навігації. Максимально інформаційно описаний кожен продукт, швидке завантаження сторінок.

Головну сторінку додатку в магазині додатків можна побачити на рисунку 1.3

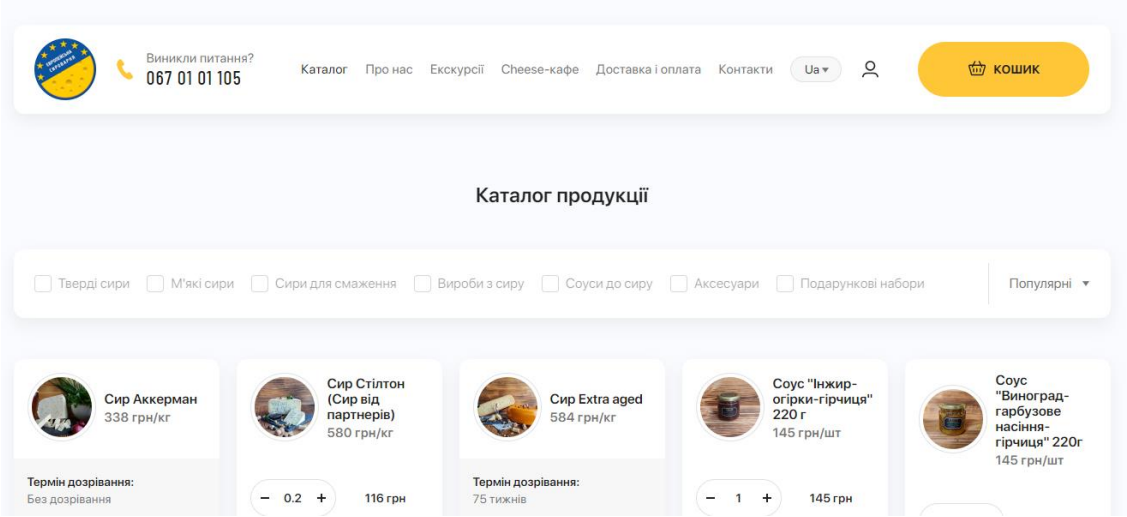

Рисунок 1.3 – Сторінка додатку EuroSir

До переваг ПЗ можна віднести:

– приємний дизайн, створює позитивне сприйняття та візуальний комфорт для користувачів. Якісно розроблений дизайн сприяє залученню користувачів та поліпшує їхній загальний враження від продукту;

– швидкість роботи на слабких пристроях, відкриття сторінок, виконання різних операцій займають менше часу через швидке відкриття сторінок, миттєву відповідь на користувацькі запити і ефективне виконання різних операцій;

– програма лояльності, додаток має вбудовану програму лояльності, яка дозволяє користувачам збирати бали або отримувати знижки з кожним замовленням. Це спонукає постійних клієнтів повертатися і робити покупки знову;

– зручне замовлення: у додатку зручно та легко замовляти сири. Користувачі можуть легко вибрати потрібні товари, додати їх у кошик і оформити замовлення всього за кілька кроків. Це забезпечує зручність і ефективність процесу оформлення замовлення;

До недоліків ПЗ відноситься:

– немає чат бота;

– невелика кількість способів оплати , тільки приват банк та нова пошта;

– відсутність можливості оглядів та відгуків: Застосунок не має функціональності, яка дозволяє користувачам залишати огляди та відгуки про сири. Це може зменшити довіру користувачів до продуктів і обмежити інформацію, яку вони можуть отримати перед покупкою.

Загалом відмінності між цими веб-додатками зводяться до їхніх особливостей і типів продукцію та послуг, які вони пропонують. Деякі програми зосереджені на соціальних функціях і чат ботах, а інші пропонують персоналізовані знижки та великий асортимент продукції , цікаві підменю.

1.4 Аналіз вимог до програмного забезпечення

Після проведення аналізу схожих застосунків та уважного розгляду вимог, поставлених перед розроблюваним програмним забезпеченням у рамках дипломного проекту, можна скласти такий перелік функціональних можливостей, які будуть включені в систему:

– реалізація архітектури back-end для взаємодії з користувачем та опрацюванням вхідних даних , операцій на сервері відповідно;

– реалізація рівнів доступу та відповідно різних прав для користувачів – користувач , адміністратор , продавець;

– функціональні можливості для створення , редагування , видалення більшості даних (продукти , відгуки , користувачі);

– функціональні можливості для редагування профілю користувача

– особистий кабінет користувача та співробітника;

– функціональні можливості для відображення геопросторових міток країн походження сирів;

– функціональні можливості для відправки електронних листів до користувача, ця функціональність дозволить забезпечити автоматизовану і ефективну комунікацію;

– функціональні можливості для оплати кредитними картами через сторонній сервіс. Список функцій можна побачити на діаграмі 1.6.

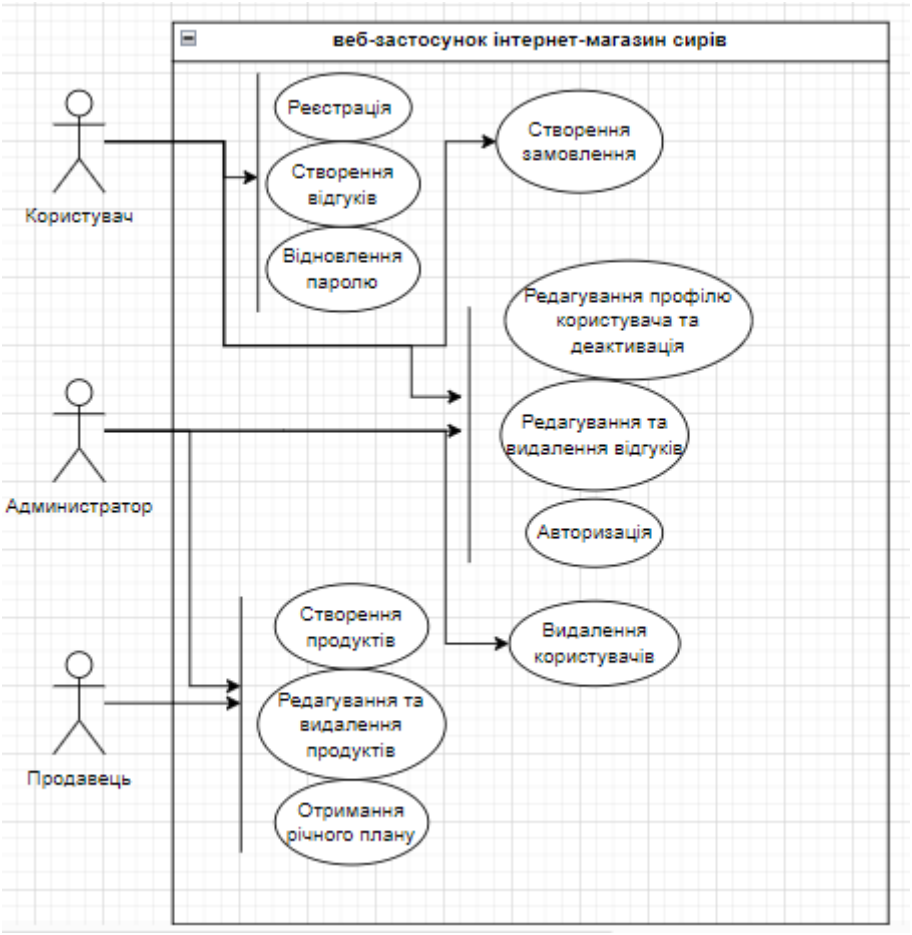

Рисунок 1.6 – Діаграма варіантів використання

Окрім цього ПЗ має відповідати таким нефункціональним вимогам:

– Front-end – інтерфейс який повинен мати приємний та мінімалістичний вигляд для взаємодії користувачів з серверною частиною додатку, де користувач матиме доступ до асортименту продукції, профілю, та свого кабінету;

– Захищеність особистих даних користувачів та співробітників компанії.

Всі ці вимоги розробленні на основі переваг та недоліків представлених аналогів. Створене програмне забезпечення повинно бути універсальним для можливості взаємодії з сторонніми сервісами , мати необхідну безпеку. Розробка програмного забезпечення повинна включати механізми та заходи для забезпечення надійності, конфіденційності та цілісності даних. В цьому контексті

розробка механізмів автентифікації та авторизації, шифрування та захисту передачі даних, а також виявлення та реагування на потенційні загрози є невід'ємною частиною програмного забезпечення.

1.4.1 Розроблення безпекових вимог

Оцінка ризиків ПЗ – є важливим етапом розробки , вона допомагає виявити потенційні загрози які можуть вплинути на безпеку в майбутньому, стабільність та можливість програми коректно працювати. Під час проектування проекту розглянуті потенційні загрози та наслідки, серед найімовірніших ризиків які можуть нашкодити роботі застосунку є:

– скомпрометована база даних;

– атака грубою силою (brute force attacks);

– міжсайтові скриптові атаки (cross-site scripting XSS attacks);

– атака відмови в обслуговувані (denial-of-service DOS attack);

– інєкція NoSQL запитів (NoSQL query injection).

Та також інші корисні практики для безпеки застосонку:

– завжди використовувати HTTPS (Hypertext Transfer Protocol Secure) для безпечного зв’язку;

– створювати випадкові токени відновлення паролю з термінами використання;

– відмовляти в доступі після зміни паролю за старими JWT токенами;

– не викладати конфіденційні конфігураційні дані до Git;

– не відправляти деталей помилок програми до клієнта;

– робити запит на повторну автентифікацію перед важливою дією;

– створити чорний список ненадійних jwt токенів. підтверджувати e-mail адресу після реєстрації;

– регулярно оновлювати та патчувати програмне забезпечення для усунення вразливостей та коригування виявлених помилок;

– використовувати механізми обмеження спроб введення пароля;

– тримати користувача авторизованим з токенами які можуть оновлюватись.

– запобігати забрудненню параметрів, що може спричинити неперехоплену помилку.

Аналіз вразливостей до веб-застосунку інтерне-магазин сирів та рішення як їх усунути представлено на таблиці 1.1

Таблиця 1.1 – Аналіз вразливостей застосунку

№	Вразливість	Наслідки	Рішення	Допоміжні Засоби
1	Вільний доступ в базі до паролей, не зашифровані токени відновлення паролю	Викрадення конфіденційних даних	Надійно защифрувати паролі, токени відновлення паролів з використанням хешування	Використати алгоритм (SHA 256) та бібліотеку bcrypt
2	Відсутність обмеження на довжину пароля	знижує рівень безпеки системи	Встановити мінімальну довжину паролю та підвищити його складність додаванням символів різних регістрів	Перевірка пароля при реєстрації за допомогою Mongoose

№	Вразливість	Наслідки	Рішення	Допоміжні Засоби
3	Немає обмеження на кількість спроб авторизації	Проникнення в захищений акаунт, крадіжка акаунту	Тимчасово блокувати обліковий запис після певної кількості невдалих спроб, встановлення таймауту між спробами авторизації	Бібліотеки bcrypt, express-rate-limit
4	Вхідні дані до серверу не перевіряються належним чином	Проникнення та введення шкідливих скриптів , крадіжка даних, видалення даних	Зберігати токен JWT в HTTPOnly cookies. Очищувати введені користувачем дані. Встановити спеціальні заголовки HTTP	Бібліотека helmet
5	Вхідні дані не перевіряються, немає обмежень в запитах до серверу	Навмисне перевантаження.Припинення або обмеження досупу до застосунку	Обмеження по кількості запитів нас сервер, Встановити обмеження на об’єм файлів що потрапляють на сервер	Бібліотека express-rate-limit

1.5 Постановка задачі

Планується зробити веб застосунок для продажу сирів та зв’язку з клієнтами. Застосунок буде складатися з серверної частини, де будуть оброблятись дані та передньою частиною(Front-end) для взаємодії з сервером та необхідним інтерфейсом, для розробки можна використовувати більшість мов програмування та бібліотек і фреймворків до них.

Застосунок буде орієнтований на продаж продуктів та автоматизацію. Він матиме необхідну кількість безпекових особливостей такі як захист від NoSQL-ін'єкцій, перевірка введених даних, шифрування паролів, обмеження доступу до адміністративних функцій тощо.

Застосунок матиме основну сторінку з сирами, на якій користувачі зможуть переглядати асортимент сирів, дізнаватися про їх характеристики, ціни та інші деталі. Особистий Кабінет, де користувачі зможуть переглядати та редагувати свої особисті дані, переглядати історію замовлень, відстежувати статуси, систему для реєстрації, авторизації та відновлення втраченого паролю, можливості адміністрування, для адміністраторів системи передбачається окремий доступ до функцій адміністрування, де вони зможуть керувати асортиментом сирів, оновлювати ціни, додавати нові товари та систему відгуків.

Висновки до розділу

В цьому розділі було проаналізовано вимоги до програмного забезпечення додатку для послуг електронної комерції, продажу сирів.

Було проведено змістовний опис і аналіз предметної області. У цьому підрозділі було наведено загальну інформацію про мови програмування які можна використати для розробки, про проблеми і складнощі які виникають при проектуванні. Проаналізовано також найпоширеніші вразливості які можуть зашкодити роботі програмного забезпечення та призвести до викрадення конфіденційних даних користувачів застосунку.

Проаналізовано методи та рекомендації до архітектури проекту щоб була можливість як протистояти цим типам атак. Також було проаналізовано технології, завдяки яким можна виконати поставлену задачу. Незважаючи на наявність багатьох різних мов програмування і ще більшої кількості фреймворків та бібліотек як для створення користувацького інтерфейсу так і для побудування внутрішньої частини додатку. Вибір цих рекомендованих технологій не тільки дозволить створити сучасний продукт, який буде відповідати сучасним тенденціям розробки веб-застосунків, а й без проблем підтримувати цей застосунок у майбутньому та легко масштабувати його в майбутньому реалізовуючи складний функціонал.

Однією з найважливіших проаналізованих тем в ході розробки проекту "Вебзастосунок інтернет магазину сирів" був аналіз вже існуючих програмних продуктів, які пропонують подібну функціональність. Після детального дослідження різноманітних застосунків і платформ було зроблено висновок, що жоден з них не задовольняє повністю наші вимоги щодо інформативності та різноманітності налаштувань. Багато програмних продуктів, що були проаналізовані, мали певні переваги, але не забезпечували повного набору функцій, які були потрібні для успішного розвитку нашого інтернет-магазину сирів. Тому, з метою забезпечення максимальної інформативності та різноманітності налаштувань, ми вирішили розробити власний вебзастосунок, який відповідає всім нашим потребам та вимогам

Додатково були проаналізовані безпекові вимоги, які допоможуть захистити програмне забезпечення від злому, інших видів атак та крадіжки конфіденційних даних користувачів застосунку. Проаналізовано за яких вразливостей є можливість зробити різні типи атак ,та додаткові засоби які допоможуть реалізувати захист застосунку.

2 ВИБІР ТЕХНОЛОГІЇ РОЗРОБКИ ДЛЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Обґрунтування вибору мови програмування

Оскільки програмне забезпечення має бути представлене у вигляді застосунку, вибір мови програмування складає важливий етап в проектуванні.

Вибір мови програмування був на основі аналізу найпопулярніших та найбільш актуальних мов програмування, популярних фреймворків та бібліотек для реалізації веб-застосунків.

Декілька головних аспектів для вибору мови створення додатку:

– Функціональність, мова повинна мати достатній набір функціональних можливостей, щоб задовольнити вимоги застосунку. Мати функціонал для роботи з базами даних;

– Продуктивність, мова повинна бути оптимізованою для веб застосунків через великий обсяг даних та велике навантаження;

– Масштабування, мова повинна бути гнучкою та мати можливість легко впроваджувати новий функціонал;

– Підтримка, мова має активну спільноту розробників, велику кількість фреймворків та бібліотек;

– Безпека, мова повинна мати велику кількість бібліотек, як вбудованих так і сторонніх, для протидії ризикам та атакам, мати інструменти для тестування, перевірки і контролю вхідних даних та обробки помилок.

2.1.1 Аналіз мови та фреймворків для реалізації серверної частини застосунку

Мова програмування Python (фреймоворк Django). Django – це безкоштовний та популярний Python-фреймворк, проект для ефективної розробки веб-додатків. Проект Django почався в 2003 році і з тих пір набув великої популярності та активно використовується спільнотою розробників. Фреймворк Django надає розробникам потужні інструменти. та готові рішення.

Проект підтримується компанією Django Software Foundation. З позитивних аспектів можна виділити:

– Продуктивність розробки, швидкий старт завдяки готовим компонентам і вбудованим функціям, що дозволяє швидко створювати додатки з меншим обсягом написаного коду;

– Масштабність, Django підтримує масштабованість без втрати в продуктивності, підтримує кешування та оптимізацію запитів до бази даних;

– Безпека, фреймоврк надає обширний захист від популярний атак, таких як CSRF (Cross-Site Request Forgery), захист від SQL-ін'єкцій та ін.;

– Багатоплатформність, Django підтримує багато різних баз даних, включаючи PostgreSQL, MySQL, SQLite, сумісний з багатьма сервісами, серверами та операційними системами.

З іншого боку, незважаючи на велику кількість переваг, які надає фреймворк для веб-розробки , в нього є також є недоліки. Ось головні із них:

– Висока складність, багато власних правил та вимог до розробки, вимагає вивчення архітектури, структури та інших особливостей;

– Обмежене гнучкість фреймворку, хоч він і забезпечує багато функцій та компонентів "з коробки", в деяких випадках це може вплинути на проект, через свої шаблонні механізми і специфічні правила розробки, що може викликати труднощі в реалізації незвичного функціоналу. Однак, це також може обмежувати вибір інших технологій та бібліотек, які можуть бути більш підходящими для певних завдань або вимог проекту.

– Вплив на продуктивність, через свій об'єм та велику кількість вбудованих функцій може мати вплив на продуктивність, особливо при великій кількості запитів, при навантаженні, що в свою чергу вимагатиме додаткової роботи розробників для оптимізації проекту.

– Серверне навантаження: Django, як і більшість фреймворків, є серверно-орієнтованим. Це означає, що він вимагає наявності серверу, який оброблятиме запити та відповідатиме на них. Для деяких проектів це може бути непотрібним навантаженням, особливо якщо вони спрямовані на легковагість та швидкість.

Мова програмування Go (або Golang), яка стала досить популярною для веб розробки через свої особливості. Сама мова почала розроблюватись всередині компанії Google ще з 2007 року. Опис переваг які надає мова в розробці веб-застосунків:

– Швидкодія, мова є компільованою мовою з швидким виконанням, має низький рівень абстракції і просту синтаксичну структуру, що сприяє швидкодії застосунку.

– Конкурентність і паралелізм, є однієї з найголовних переваг мови, має вбудовану модель для створення легких потоків виконання, що в свою чергу дозволяє легко обробляти багатопоточні завдання.

– Простота розробки, є ще одним плюсом мови, синтаксис, простий та зрозумілий, сприяє швидкій розробці веб-застосунків, також має безліч вбудованих інструментів тестування та розгортання.

– Надійність, Go пропонує вбудовану підтримку обробки помилок і виключень, автоматично вивільняє пам’ять і зменшує ймовірність витоків пам’яті.

Недоліками мови Go, хоч це і досить ефективний інструмент для розробки, є:

– Обмежена кількість бібліотек і фреймворків , порівняно з такими мовами, як Python, JavaScript, що в свою чергу призводить до того, що деякі завдання доведеться вирішувати з нуля;

– Обмежені можливості шаблонізації, система шаблонів менш потужна порівняно з іншими фреймворками, Django наприклад або Ruby. Це може збільшити час розробки, який буде витрачено імплементацію;

– Відсутність універсальності, мова Go була спроектована з цілю на певні сценарії програмування, зокрема на розробку мережевих додатків та веб-серверів. Тому вона може бути менш підходящою для інших типів проектів, таких як мобільні додатки або наукові обчислення;

– Відсутність уніфікованого стандарту, мові Go немає однозначно встановленого стандарту для багатьох аспектів розробки.

Мова Go не надає підтримку дзеркального відображення (reflection), що ускладнює деякі завдання, зокрема, автоматичну генерацію коду, динамічне створення об'єктів та зчитування інформації про типи даних. Це може обмежити гнучкість та потенційні можливості деяких програм.

Платформа NodeJs. Крос-платформне середовище виконання JavaScript коду за межами веб-браузера, побудована на двигуні V8 JavaScript , розробленому компанією Google, є надзвичайно корисним інструментом для розробки веб-додатків та інших програм, що базуються на JavaScript. Дозволяє виконувати JavaScript код на стороні сервера , що робить його ідеальним варіантом для розробки веб-застосунків, веб-сервери, стрімінгові сервіси тощо. NodeJs надає зручний зручний спосіб для роботи з мережевими протоколами і забезпечує швидку обробку запитів для високонавантажених систем, дає можливість взаємодіяти з серверами, клієнтами і іншими мережевими ресурсами.

Перевагами платформи є:

– Висока продуктивність, так як NodeJs використовує однопотокову, подієву модель вводу-виводу, яка дозволяє обробляти багатопоточні операції швидко та ефективно, також двигун V8 забезпечує швидкість роботи для браузера;

– Широкий вибір бібліотек та модулів, NodeJs має велику та активну спільноту розробників, що в свою чергу дає велику кількість готових модулів і бібліотек, доступних через пакетний менеджер (npm). Це спрощує розробку та зменшує час розробки;

– Можливість повторного використання коду, так як з NodeJs ми використовуємо одну мову програмування (JavaScript) як на серверній так і на клієнтській частині застосунку, це в свою чергу спрощує обмін кодом між ними. Це дозволяє створювати загальні модулі, функції та компоненти.

– Масштабування, можливість легко масштабувати веб-застосунки і обробляти велику кількість запитів , через підтримку асинхронних операцій, що допомагає створювати масштабовані додатки з високою пропускною здатністю, високим навантаженням на систему.

Хоча NodeJs має свої переваги в нього також є й недоліки. А саме його :

– Однопоточність, хоч це і сприяє високій продуктивності, він також є обмеженням при обробці великої кількості одночасних запитів на сервер, через що розробникам потрібно розглядати кластеризування, та інші шляхи вирішення проблеми, що в свою чергу вимагає знання аспектів платформи та різних стратегій масштабування.

– Менша ефективність для обробки великих обсягів обчислень NodeJs краще підходить для операцій вводу/виводу та обробки подій ніж для складних та важких обчислень, через довгий час виконання таких операцій.

2.1.2 Аналіз засобів для реалізації користувацького інтерфейсу застосунку

Так як, при розробці дипломного проекту необхідно забезпечити кращу продуктивність застосунку, підвищену доступність до слабких систем, покращену індексацію пошуковими системами та швидку розробку був обраний підхід Server-Side Rendering для відображення HTML коду, який буде генеруватись на сервері. Одним з ключових аспектів реалізації SSR є використання шаблонізаторів для веб-розробки. Шаблонізатори дозволяють створювати динамічні HTML-шаблони, в яких дані можуть бути вбудовані або динамічно підставлені. Вони спрощують процес створення та керування вмістом веб сторінок, а також дозволяють забезпечити більшу гнучкість та зменшення часу розробки.

Тому розглянемо популярні шаблонізатори для веб-розробки, їх переваги та недоліки:

1) Handlebars:

Переваги: простий синтаксис, який дозволяє швидко розібратись та почати реалізовувати проект, пропонує можливість використовувати чистий HTML, що дозволяє розробникам комфортно працювати зі структурою сторінок і шаблонів без необхідності вивчати новий синтаксис або мову. Handlebars можна легко інтегрувати з іншими фреймворками та бібліотеками, що робить його гнучким.

Недоліками ж є відсутність підтримки складних конструкцій, умови цикли та інше, це в свою чергу обмежує гнучкість шаблонування.

2) EJS (Embedded JavaScript Templates):

Переваги: інтеграція з мовою JavaScript , використання цієї мови прямо в шаблонах, відносно простота у використанні. EJS дозволяє повторне використання коду шаблонів та компонентів, що полегшує розробку та управління шаблонами. Розробники можуть створювати модульні шаблони і використовувати їх в різних частинах додатку.

Недоліками ж є більш складна синтаксична структура порівняно з іншими рішеннями, що потребує необхідність в додаткових знаннях мови JavaScript.

3) PUG :

Перевагами є досить зручний синтаксис через те, що шаблонізатор використовує мінімальну кількість знаків в ньому. Це дозволяє зменшувати кількість написаного кода та полегшує читання. Вбудовані можливості для різних функціональних рішень, а саме цикли, умови та інше. Легко інтегрується з NodeJs, що дозволяє забезпечувати єдиний стек технологій.

Недоліками ж є відсутність стандартного синтаксису HTML так як шаблонізатор використовує свій власний. Потреба в компіляції що може займати деякий час. Його компактний синтаксис та використання розкладки замість тегів можуть ускладнити процес локалізації та зробити код менш зрозумілим для перекладачів, якщо інтегрувати Pug з іншими інструментами або фреймворками, можуть виникнути проблеми взаємодії та сумісності.

У кожного з цих рішень є свої переваги та недоліки, тому вибір залежить від потреб застосунку, кожен шаблонізатор має свої переваги та недоліки, які потрібно враховувати при виборі для конкретного проекту. Handlebars пропонує простий та зручний синтаксис, але може бути обмеженим у складних логічних операціях. EJS дозволяє використовувати JavaScript безпосередньо в шаблонах і має широкі можливості, але може призводити до збільшення обсягу коду. Pug має компактний синтаксис та дозволяє використовувати розкладку замість тегів, що полегшує читання, але може бути менш зрозумілим для новачків

2.1.3 Обрані мови та фреймворки для вебзастосунку

Отже, визначивши необхідні умови та вимоги до проекту та проаналізувавши актуальні рішення для розробки ПЗ, була вибрана мова програмування JavaScript для реалізації серверної частини застосунку на платформі NodeJs так як вона має деякі переваги для мого дипломного проекту.

Так як NodeJs заснована на асинхронному неблокуючому введені/виведені (input/output), це дозволить оптимально використовувати системні ресурси та обробляти одночасно велику кількість з'єднань користувачів до серверу. Що ідеально підходить для інтернет магазину, бо зможе обробити велику кількість запитів в реальному часі.

Також обширна кількість різних бібліотек, модулів ,які доступні через пакетний менеджер npm дозволить прискорити розробку, реалізувати розширений функціонал, наприклад бібліотеки для захисту від різних типів атак на застосунок. Висока продуктивність, швидке виконання JavaScript коду дозволить досягти оптимізації та швидкодії веб-застосунку, важливий аспект, де користувачі очікують швидкої реакції на свої запити.

Що до реалізації користувацького інтрефейсу сторони додатку вибір впав на шаблонізатор Pug, який зможе забезпечити зручну та ефективну розмітку HTML, для реалізації швидкодії застосунку, яка позитивно вплине на досвід користувачів з слабкими системами. Що в свою чергу полегшить завантаження та відображення сторінок в порівнянні з ReactJs, який використовує віртуальну DOM і буде мати більше обчислювального навантаження. Pug буде швидшим в контексті відображення статичного вмісту сторінки. Він також є простим у використанні і не вимагає великої кількості різних додаткових інструментів та конфігурацій для початку роботи , що впливає на час розробки. Pug підтримує використання різних функцій та змінних в шаблонах, що дозволяє здійснювати розрахунки, маніпулювати даними та створювати динамічний вміст сторінок. Це робить Pug гнучким та розширюваним інструментом для шаблонування. Він легко взаємодіє з NodeJS в порівнянні з ReactJs.

2.2 Аналіз та вибір систем управління базами даних (СУБД)

Популярні SQL системи управління базами даних, такі як MySQL, PostgreSQL та Microsoft SQL Server мають свої недоліки та переваги.

Перевагами буде:

– Стандартизованість, SQL СУБД використовують стандартну мову запитів SQL, що дозволяє легко переносити код між різними системами та використовувати розповсюджені підходи.

– Надійність та стабільність, системи мають довгу історію розробки і тестування, високу стійкість до відмов та забезпечують механізми резервного копіювання та відновлення даних.

– Підтримка транзакцій, системи SQL підтримують транзакції , що дозволяє гарантувати цілісність даних.

Недоліками ж SQL систем управління базами є те що вони мають обмежене вертикальне масштабування, що означає зниження продуктивності при великому обсязі даних та навантажені. Також недоліком є складність налаштування таких систем та досягнення необхідної оптимізації для досягнення оптимальної продуктивності застосунку.

Тому вибір впав на NoSQL систему управління базами даних MongoDB, вона надає гнучкість та легку масштабованість для зберігання та обробки даних. Вона використовує модель документів, де дані зберігаються у JSON-подібних документах з динамічною схемою та може працювати в розподіленому середовищі з реплікацією та розподіленим шардуванням, що забезпечує високу доступність та надійність системи..

MongoDB має високу продуктивність і швидкий доступ до даних. Базу можна легко масштабувати горизонтально, додаючи нові сервери, що дозволяє обробляти великі обсяги даних і забезпечувати високу доступність. Також важливим для дипломного проекту є вбудована підтримка геоданих, що дозволяє збирати на опрацьовувати географічні дані, так як я хочу прив’язати сири до конкретного місцезнаходження.

Окрім цього MongoDB має потужну мову запитів, що дозволяє виконувати різноманітні операції, від простих запитів до складних агрегаційних операцій та пошуку з використанням індексації.

Також в порівнянні з іншими NoSQL СУБД MongoDB має більше функціональних можливостей та простоту масштабування.

Висновки до розділу

У другому розділі дипломного проекту було виконано аналіз та вибір засобів розробки, мов програмування та СУБД до програмного забезпечення.

У розділі були спочатку були описані популярні та актуальні мови програмування та фреймворки і платформи за допомогою яких розроблюються веб-застосунки. Було розглянуто три мови програмуванн , з подальшим аналізом кожної з ним, недоліками та перевагами. Такі дії допомагають краще розуміти систему в цілому, що дуже пришвидшить в майбутньому процес розробки та написання коду.

Також у цьому розділі було порівняно різні шаблонізатори, які їх переваги та недоліки, та чому вибір впав на них .

Також порівняльний аналіз баз даних реляційних СУБД, таких як MySQL, PostgreSQL і Oracle, що є добре розповсюдженими рішеннями, які підходять для багатьох типів додатків. Вони відомі своєю надійністю, стабільністю та підтримкою ACID-властивостей. Та NoSQL СУБД, таких як MongoDB, CouchDB і Redis, які пропонують гнучкість і швидкодію в роботі з нереляційними даними. Вони вже краще підходять підходять для веб-застосунків, які потребують горизонтального масштабування і високої продуктивності при роботі з великим обсягом даних. Однак, вони можуть мати обмежену підтримку транзакцій і складніші механізми запитів.

Одним з важливих елементів цього веб-додатку є вибір підходящої системи керування базами даних. Та чому важливо використовувати саме NoSQL базу даних MongoDB для веб-застосунку інтернет магазину-сирів, а не SQL.

3 МОДЕЛЮВАННЯ ТА КОНСТРУЮВАННЯ ЗАСТОСУНКУ

3.1 Моделювання програмного забезпечення

Програмне забезпечення у вигляді веб-застосунку інтернет магазину сирів призначене для продажу та ознайомлення клієнтів з продуктами. Одна з головних функцій цього застосунку - автоматизація управління магазином. Це означає, що процеси, пов'язані зі зберіганням, оновленням і відстеженням товарів, обробкою замовлень та взаємодією з клієнтами, можуть бути виконані швидко та ефективно завдяки програмному забезпеченню.

У веб-застосунку передбачено 5 типів користувачів:

1) Неавторизований користувач;

2) Користувач;

3) Адміністратор;

4) Головний виробник сиру;

5) Виробник сиру.

Неавторизований Користувач:

– Може переглядати вітрину продуктів та переглядати детальний опис продукту. Неавторизований користувач змінюється після реєстрації або авторизації в додаток на користувача. Гість веб-застосунку може відновити втрачений пароль, та одразу створити новий після чого знову авторизуватись. Проте він не може додавати, редагувати або видаляти продукти, залишати коментарі чи відгуки.

Користувач:

– Має доступ до особистого кабінету користувача, де він може здійснювати різні дії, пов'язані з управлінням своїм профілем. В рамках особистого кабінету, користувач має зручний доступ до своїх особистих даних, таких як ім'я, контактна інформація та інші дані, пов'язані з профілем. Окрім перегляду, користувач також може змінювати свої дані профілю. Він може редагувати інформацію, змінювати контактні дані тощо. Крім того, користувач має можливість завантажити фотографію профілю.

Користувач може замовити товар, та переглянути статус оплати і що він замовив. Він може створювати відгуки до продуктів, редагувати та видаляти свої відгуки. Користувач може деактивувати свій акаунт

Адміністратор:

– Адміністратор не має можливості створювати відгуки до продуктів та редагувати чи видаляти свої відгуки. Ці дії зазвичай залишаються під відповідальністю звичайних користувачів. Адміністратор має можливість створювати продукти, редагувати опис продукту, завантажувати фотографії продукту. Він може може переглядати всі замовлення, зокрема інформацію про замовника, список продуктів та статус замовлення.

– Він може редагувати та видаляти відгуки інших користувачів, має можливість переглядати всі замовлення, редагувати замовлення та видаляти його.

Головний виробник сиру, виробник сиру:

– Виробники сиру можуть переглядати місячний план, який містить інформацію про вироблену кількість сиру, продажі, запаси тощо. Також можуть створювати продукти. Не можуть створювати та редагувати відгуки. Також як і у всіх користувачів є профіль

На початку роботи з застосунком кожен гість може переглядати головну вітрину та додаткову сторінку з інформацією про продукт. Для доступу в особистий кабінет та можливості замовити Сир, залишати відгуки він може зареєструватись в системі, або авторизуватись. Якщо користувач вже зареєстрований але не може згадати пароль , він може його відновити за допомогою спеціального електронного листа в якому буде посилання на відновлення паролю, після чого користувач створить новий пароль і зможе увійти. У разі реєстрації нового користувача, на вказану ним електронну адресу буде надіслано запрошення до особистого кабінету. Це запрошення міститиме необхідну інформацію та посилання для входу в особистий кабінет. В особистому кабінеті новий користувач матиме можливість редагувати свій профіль, включаючи особисті дані та контактну інформацію, а також виконувати інші дії, пов'язані з управлінням своїм обліковим записом.

Доступні сторінки для користувачів :

1) Signup – форма реєстрації користувачів;

2) Login – форма авторизації користувачів для доступу до захищених шляхів;

3) Me – особистий кабінет;

4) My-orders – замовленні продукти;

5) Головна сторінка – вітрина товарів;

6) Product – Повна інформація про товар;

7) Reviews – Відгуки користувача;

8) Billing – Статус оплати.

3.2 Архітектура програмного забезпечення

В роботі буде використана мова програмування JavaScript , оточення для розробки – NodeJs. Програмне забезпечення буде побудовано з використанням клієнт серверної архітектури, де бекенд(Back-End) буде використовувати для обробки запитів REST API, API є абревіатурою від Application Programming Interface, що означає програмний інтерфейс програми. API представляє набір правил і функцій, що дозволяють двом різним програмам взаємодіяти один з одним. В свою чергу це репрезентативна передача стану (REST) – це архітектурний стиль, який здійснює реалізацію клієнта та сервера незалежно один від одного.

Сервіси в REST API взаємодіють за протоколом HTTP для взаємодії між клієнтом та сервером. Клієнтські додатки можуть виконувати різні типи запитів до серверу, такі як один з основних методів HTTP – GET, використовується для отримання даних з сервера. Клієнтський додаток надсилає GET-запит до сервера, який повертає відповідь з необхідною інформацією., POST для створення нових ресурсів, PUT для оновлення існуючих ресурсів та DELETE для видалення ресурсів. Ці методи надають стандартизований спосіб обміну даними між різними компонентами системи, що робить їхню взаємодію ефективною.

Перевагами такого підходу є те що веб-застосунок може бути легко масштабованим. Також перевагою та чому був обраний такий підхід є незалежність від платформи REST API може використовуватись на будь-якій платформі що підтримує HTTP.

Підтримка платформою HTTP може дозволити взаємодіяти між різними додатками написаними ні різних мовах та різними розробниками. Також архітектура дозволить легко інтегруватись з іншими системами та службами , так як використовуються стандартні протоколи HTTP, JSONдля передачі даних.

Зберігання даних, дані як використовуються в веб-застосунку будуть зберігатись у базі даних MongoDB, в свою чергу сервер та базу даних буде з'єднувати спеціальна ODM(Object Data Modeling) Mongoose бібліотека для роботи, яка дозволяє зіставляти об'єкти класів та документи колекцій із бази даних.

Рівень бізнес логіки знаходить на серверній частині додатку, саме на ньому будуть проходити основні процеси, на ньому буде проходити взаємодія між іншими рівнями системи. Архітектура показана на рисунку 3.1.

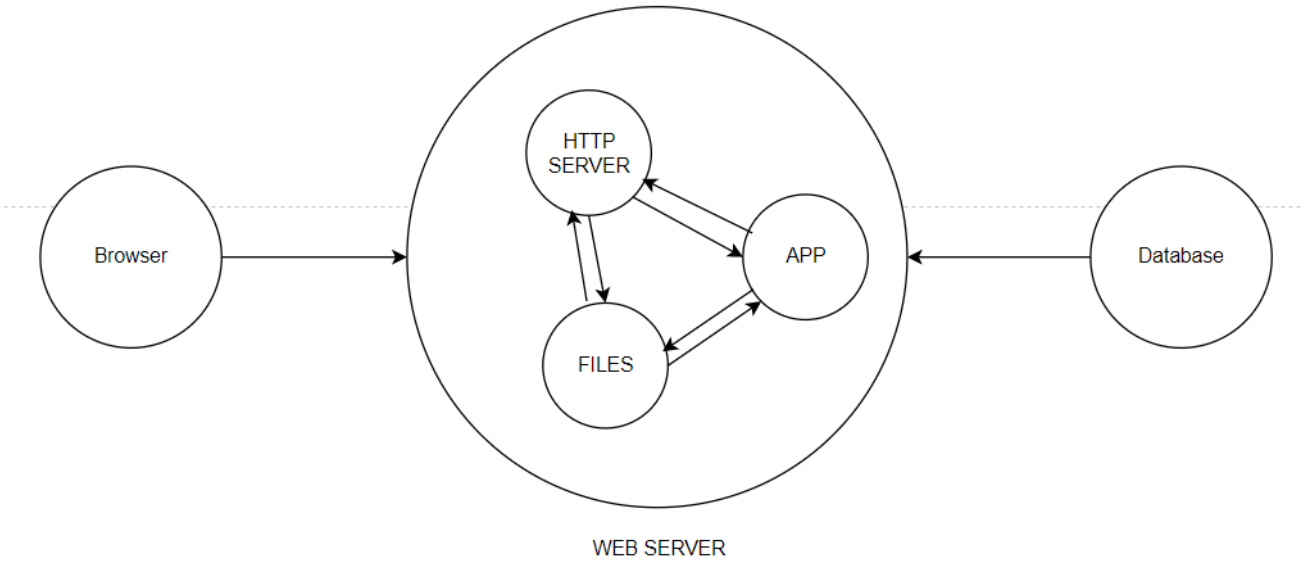

Рисунок 3.1 – Схема архітектури серверної частини

Загалом систему можна розподілити на 3 основні складові , що представляє структуру проекту.

Представлення даних – WEB клієнт, як дасть можливість користувачам використовувати функціонал веб-застосунку.

Логіка системи представлена у вигляді різних частин системи зібраних в одному місці, які будуть відповідати за обробку інформаіції, зберігання в базі передачу даних між різними рівнями системи. Логіка зв’язку між різними компонентами системи – вебзастосунок магазину сирів. Архітектура системи. Також там показано як буде інтегруватись платіжний сервіс Stripe.

На схемі показані з’єднання різних частин системи, Stripe платіжна система, яка дозволяє безпечно здійснювати оплату різних послуг та товару при цьому вона дозволяє не зберігати дані карток покупців на сервері, що полегшує роботу системи , так як ці дані не потрібно шифрувати та зберігати.

Mongo driver Це офіційні або сторонні бібліотеки, які надають API для взаємодії з MongoDB з різних мов програмування, таких як Node.js, Python, Java, PHP і т.д. Драйвери дозволяють виконувати операції зчитування, запису, оновлення і видалення даних в MongoDB. Зв’язок між сервером і MongoDB дає можливість програмному забезпечування, додатку отримувати та зберігати дані в базі, виконувати запити різноманітної складності, оновлювати та видаляти інформацію.

3.3 Розробка алгоритму роботи системи

Дана система має виконувати різноманітні функції:

– Авторизація. Розробка механізму авторизації, що дозволяє користувачам увійти в систему за допомогою ідентифікаторів, таких як логін і пароль.

– Керування користувачами;

– Керування продуктами;

– Керування відгуками;

– Керування замовленнями.

Основна задача кожної із функцій обробляти або надавати інформацію для автоматизації процесів адміністрування та продажу сирної продукції.

3.3.1 Авторизація

Важливий функціонал для безпеки додатку, також сюди входять функції реєстрації, ідентифікації користувача, захищення маршрутів до інших функцій застосунку, перевірка чи користувач пройшов ідентифікацію. Надання доступу різним ролям до захищених маршрутів. Функціонал для відновлення втраченого паролю та зміни старого паролю на новий.

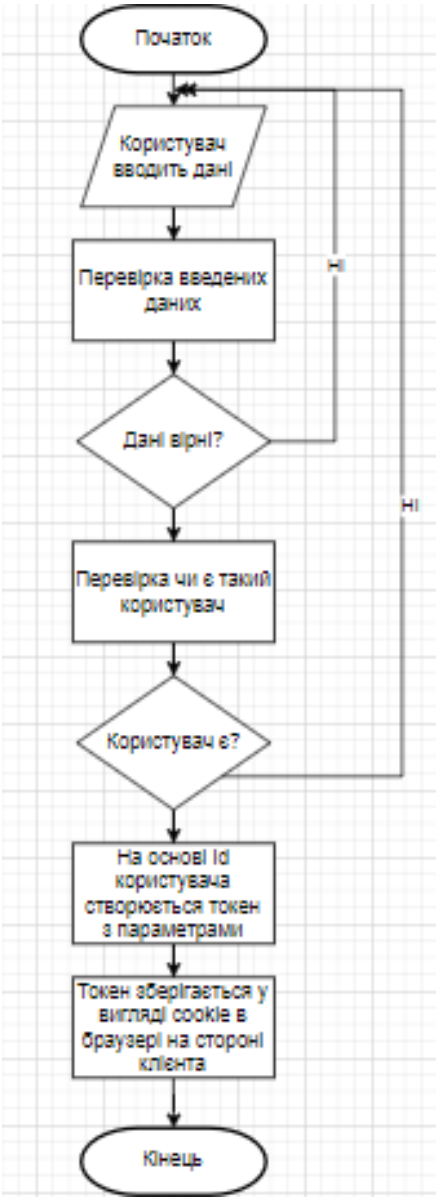

Рисунок 3.2 – Алгоритм авторизації користувача

В даному випадку основний алгоритм це авторизація користувача: спочатку користувач вводить дані, після чого в програмі перевіряється чи ці дані підходять для роботи програми, якщо все добре з цих даних береться електрона адреса та перевіряється чи існує такий користувач в базі та чи правильний пароль. Вже після цього в застосунку створюється спеціальний токен на основі секретного коду, id користувача та надаються опції про час валідності такого токена, тобто скільки часу він активний.

Рисунок 3.3 – Алгоритм оновлення даних

Керування користувачами включає в себе редагування свої даних даних, редагування даних інших користувачів та видалення. Розберемо алгоритм оновлення даних користувача. Спочатку програма отримує поточні дані які потрібно редагувати , це включає такі поля як, ім’я користувача , його електрона пошта пароль , та фотографія. Після чого буде відображена форма редагування , з полями для введення нових значень. Потім ці дані збираються після натискання кнопки і відправляються перевірятись на сервер і якщо вони валідні та підходять вимогам застосунку то зберігаються в базі. Якщо дії успішні -відображається повідомлення про успішне оновлення.

Рисунок 3.4 – Алгоритм знаходження відстані до продукту за графічними координатами

Керування продуктами схоже до керування користувачами, однак на відміну від них мають функціонал для фільтрації результатів на сервері. Різні типи фільтрів: кількість результатів та сторінки , продукти в певному радіусі від клієнта, та отримання дистанції до найближчого головного магазину сирів, відстань яка визначається може бути як в кілометрах так і в милях. Алгоритм фільтрації за дистанцією відбувається наступним чином, спочатку отримання географічних координат користувача, перевірка даних на валідність, вже після визначається множник дистанції в залежності в параметру (кілометри або милі). Далі з всіма цими даними відбувається агрегаційний запит до бази даних, вибірка певних параметрів виводу і відправка відфільтрованих товарів.

Керування відгуками та замовленнями в принципі не мають складного алгоритму, там застосовуються базові операції: створення, редагування, видалення, читання.

Алгоритм керування відгуками:

– Створення відгуку, отримання даних від користувача, включаючи текст та рейтинг відгуку. Створення нового запису в базі даних, з пов’язаними даними про користувача та ідентифікатор продукту.

– Редагування відгуку, отримання ідентифікатору користувача, потім завантажити з бази даних і показати йому всі відгуки які він створював. Отримання ідентифікатору відгуку , який буде відредагований , завантажити цей відгук з бази даних, отримати оновлену інформацію про відгук та зберегти.

– Видалення відгуку, видалення відгуку схоже на алгоритм редагування, спочатку отримуємо ідентифікатор відгуку який потрібно видалити, та видаляємо його з бази, якщо роль користувача адміністратор, то він може видаляти навіть чужі відгуки.

Алгоритм керування замовленнями, аналогічний алгоритму керування відгуками, передбачає наявність набору операцій, що забезпечують ефективну обробку та управління замовленнями. Даний алгоритм має наступний набір операцій, які виконуються для забезпечення належного керування замовленнями: створення, видалення та редагування.

3.4 Опис структури та організації бекенду (Back-end)

В контексті визначеної архітектури серверної частини проекту яка буде виконана у вигляді REST API, важливо зазначити архітектурний підхід до реалізації програмного забезпечення, а саме розділення додатку на три основні частини:

1) Модель – відповідає за управління бази даних, в контексті MongoBD також вміщає деякий функціонал в роботі з базою.

2) Представлення – відповідає за візуалізацію даних, так як програмне забезпечення буде мати шаблони, які потім будуть рендеритись на сервері в HTML код, тобто інформацію яка буде представлена у вигляді зрозумілому користувачу.

3) Контролер – відповідає за обробку запитів користувача, та відправку відповідних запитів до моделі та представлення. Контролер приймає вхідні дані від користувача, виконує необхідні дії та оновлює модель та представлення відповідно.

MVC дозволяє розділити логіку програми на незалежні компоненти, що спрощує розробку, підтримку та тестування додатків. Він також сприяє відокремленню презентації даних від логіки та забезпечує більшу гнучкість та перевикористання коду.

MVC є одним з найпоширеніших патернів в розробці програмного забезпечення і широко використовується в різних платформах та фреймворках, включаючи веб-розробку, мобільні додатки та десктопні застосунки. Спочатку, він забезпечує чітке розділення між логікою програми, візуалізацією даних та управлінням даними. Це дозволяє зберігати код організованим та легким для розуміння та підтримки.

На відміну від аналогів, таких як двошарова архітектура або безпосереднє включення логіки у користувацький інтерфейс, MVC (Model-View-Controller) забезпечує більш гнучке та модульне управління логікою програми.

Крім того, REST API також може використовувати інші підходи до організації коду та логіки, такі як сервіси (Services), репозиторії (Repositories) та маршрутизатори (Routers). Це залежить від конкретних потреб проекту та використовуваних фреймворків або бібліотек. Але саме у дипломному проекті веб-застосунок інтернет-магазин сирів використовується ще маршрутизатори (Routers).

Отже, хоча REST API не обов'язково має використовувати точно таку саму реалізацію MVC, підхід MVC може бути використаний для організації логіки та структури REST API.

Також використання MVC в REST API вебзастосунку інтернет магазину сирів краще за аналогічні патерни проектування тим що надає:

– Розділення відповідальностей, такий підхід дозволяє чітко розділити логіку застосунку на модель представлення та контролер, що в свою чергу спрощує управління кодом, спрощує організацію проекту і дозволяє розробникам зосередитись на розробці.

– Модульність, кожен компонент цього патерну може працювати незалежно від інших. Це дозволяє змінювати або розширювати один компонент, не впливаючи на інші. Наприклад, зміни в логіці контролера не повинні впливати на модель або представлення.

– Повторне використання коду, Через розділення логіки на окремі компоненти, можна перевикористовувати код в різних частинах додатку. Наприклад, модель може бути використана в інших контекстах або контролерах.

– Можливість тестувати компоненти незалежно, через розділення логіки на окремі компоненти, можна повторно використати код в різних частинах додатку. Ця розділеність компонентів у MVC дозволяє незалежне тестування кожного компонента, що є важливим аспектом в розробці програмного забезпечення. Тестування може бути проведене окремо для моделей, представлень та контролерів, що спрощує виявлення та усунення помилок у кожній окремій частині додатку.

Наприклад, модель може бути використана в інших контекстах або контролерах. Має такі характеристики:

– Масштабування, MVC надає гнучкість у розширенні функціональності додатку. Нові функції можна додавати, розширюючи або замінюючи окремі компоненти без впливу на решту системи. Тим більше при використані стилю REST API з задумкою на майбутнє масштабування проекту.

– Розподіленість, MVC робить систему більш розподіленою, що дає можливість розгортання різних компонентів на окремих серверах або використання різних сервісів для кожного компонента.

Загалом, використання патерну проектування MVC (Model-View-Controller) у контексті REST API веб-застосунків відкриває можливості для створення більш організованої та структурованої системи. , що сприяє полегшенню розробки, тестування та підтримки додатку. Крім того, такий підхід сприяє полегшенню розширення та перевикористання коду, що робить його більш ефективним та підтримуваним у довгостроковій перспективі. Отже, використання патерну проектування MVC є важливим елементом у процесі розробки веб-застосунків на основі REST API.

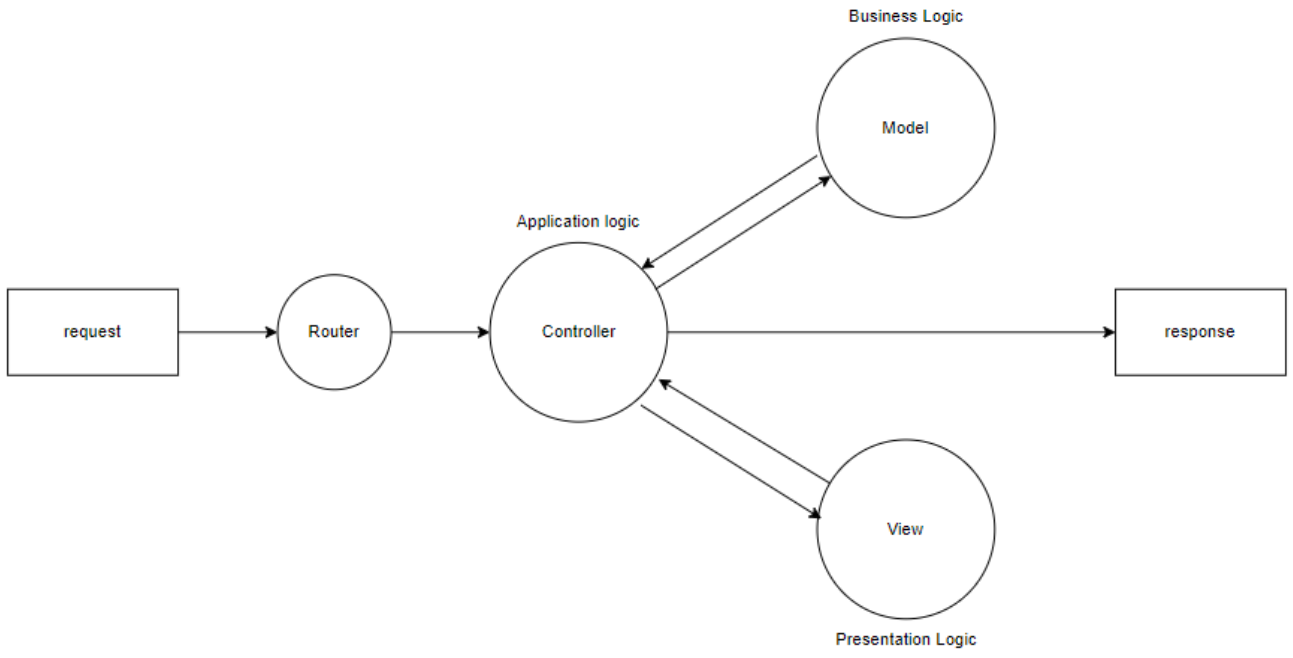

Рисунок 3.5 – Представлення Back-End архітектури

3.5 Розробка бази даних

В розробці програмного забезпечення була використана MongoDB та бібліотека mongoose для реалізації бази даних. Вона документоорієнтованою, тому дані там зберігаються у вигляді документів у форматі BSON (Binary JSON).

Це дозволяє ефективно працювати з даними різної структури без необхідності використовувати дуже обмежені схеми, як в SQL базах.

MongoDB використовує модель колекцій та документів, де кожна колекція містить групу документів з різною структурою, але зв'язаних спільною тематикою або типом даних. Це дозволяє зберігати та оновлювати дані без обмежень, що характерні для традиційних реляційних баз даних. Кожен документ у колекції містить дані у форматі ключ-значення, що спрощує роботу з ними. Це дозволяє легко зберігати та оновлювати дані без необхідності перетворювати їх у формат реляційної бази даних.

Що до Mongoose, то вона є бібліотекою для взаємодії з MongoDB в середовищі Node.js. Вона надає зручний інтерфейс для роботи з базою даних, що дозволяє визначати моделі даних, створювати, зчитувати, оновлювати та видаляти дані. Mongoose також надає можливість використовувати схеми для валідації даних та збереження взаємозв'язків між документами. Разом вони створюють потужний інструментарій для розробки сучасних веб-застосунків з ефективною роботою з даними. Цей поєднаний підхід MongoDB і Mongoose надає значну перевагу у розробці програмного забезпечення. Він спрощує створення моделей даних, валідацію та маніпуляцію збереженими даними. Крім того, MongoDB та Mongoose забезпечують швидкий доступ до даних і можливість.

Схема бази даних.

Для реалізації веб-застосунку було створено 4 таблиці:

1) User;

2) Review;

3) Booking;

4) Product.

Таблиця User.

Таблиця(колекція) містить у собі список користувачів системи з детальної інформацією. Більш детальна інформація про назви полів та типи даних наведена у табл. 3.1.

Таблиця 3.1 – Таблиця користувачів

Назва поля	*Тип Даних*
_id	ObjectId (PK)
name	String
email	String
photo	String
role	String
password	String
passwordConfirm	String
passwordChangedAt	Date
passwordResetToken	String
passwordResetExpires	Date
active	Boolean

Відносини: USER до REVIEW: зв’язок “один до багатьох”. Кожен користувач може мати декілька відгуків. Поле відгуків у таблиці USER містить масив значень ObjectId, які посилаються на відповідні відгуки. Від USER до BOOKING: відносини “один до багатьох”. Кожен користувач може мати декілька бронювань. Також USER містить масив значень ObjectId, які посилаються на пов’язані бронювання в BOOKING.

Таблиця Review.

У колекції "Відгуки до продуктів" кожен документ представляє окремий відгук користувача до певного продукту. Поле "_id" містить унікальний ідентифікатор відгуку, що дозволяє однозначно ідентифікувати кожен запис. Поле "productId" зберігає ідентифікатор продукту, до якого належить відгук.

Більш детальна інформація про назви полів та типи даних наведена у табл. 3.2.

Таблиця 3.2 – Таблиця відгуків

Назва поля	*Тип Даних*
_id	ObjectId (PK)
review	String
rating	String
createdAt	String
product	ObjectId (FK -> PRODUCT)
user	ObjectId (FK -> USER)

Відносини таблиці:

1) REVIEW до PRODUCT: зв’язок “багато до одного”. Кожен відгук відноситься до конкретного товару. Поле продукту в таблиці Review – це значення ObjectId, яке посилається на пов’язаний продукт.

2) REVIEW до USER: зв’язок “багато до одного”. Кожен відгук належить конкретному користувачеві. Поле користувача в таблиці перегляду є значенням ObjectId, яке посилається на пов’язаного користувача.

Таблиця Booking.

У колекції "Замовлення" кожен документ представляє окреме замовлення продукту. Поле "_id" містить унікальний ідентифікатор замовлення, що дозволяє однозначно ідентифікувати кожен запис. Поле "user" зберігає ідентифікатор користувача, який зробив замовлення. Також є посилання на користувача. Поле "product" містить ідентифікатор продукту, який був замовлений. Це зовнішній ключ, який посилається на ідентифікатор продукту у таблиці "PRODUCT".

Поле "price" вказує на вартість замовлення. Поле "createdAt" містить дату створення замовлення, "paid" вказує на статус оплати замовлення. Значення true вказує, що замовлення було сплачено, а значення false - що замовлення не було сплачено. Більш детальна інформація про назви полів та типи даних наведена у табл. 3.3.

Таблиця 3.3 – Таблиця замовлень

Назва поля	*Тип Даних*
_id	ObjectId (PK)
product	ObjectId (FK -> PRODUCT)
user	ObjectId (FK -> USER)
price	Number
createdAt	Date
paid	Boolean

Відносини таблиці:

1) BOOKING до PRODUCT: зв’язок “багато до одного”. Кожне бронювання пов'язане з конкретним продуктом. Поле продукту в таблиці Booking є значенням ObjectId, яке посилається на пов’язаний продукт.

2) BOOKING до USER: відношення “багато до одного”. Кожне бронювання пов’язане з певним користувачем. Поле користувача в таблиці BOOKING – це значення ObjectId, яке посилається на пов’язаного користувача.

Важливість відносин між таблицями "BOOKING", "PRODUCT" та "USER" полягає в установленні зв'язків і залежностей між об'єктами цих таблиць. Ці відносини надають додаткову функціональність і можливості для ефективного управління даними та забезпечення цілісності і консистентності системи.

Таблиця Product.

Таблиця "Product" (колекція "Product") створена з метою збереження розширеної інформації про продукти та надання широкого спектру даних щодо кожного продукту. Вона містить докладний перелік атрибутів, включаючи ціну, конкретну дату створення, опис та багато іншого. Крім того, в таблиці присутнє посилання на персонал, що додає додатковий рівень зв'язку між продуктами та їх персоналом. Це дозволяє відстежувати, які користувачі мають право доступу до конкретних продуктів та керувати цим доступом. Більш детальна інформація про назви полів та типи даних наведена у таблиці 3.4 нижче.

Таблиця 3.1 – Таблиця продуктів

Назва поля	*Тип Даних*
_id	ObjectId (PK)
name	String
slug	String
durationAging	Number
weight	Number
calories	String
ratingsAverage	Number
ratingsQuantity	Number
price	Number
priceDiscount	Number
summary	String
description	String
imageCover	String
images	[String]
createAt	Date
cheeseCreateDates	[Date]
secretProduct	Boolean
mainStoreLocation	(вкладення):
mainStoreLocation.type	String
mainStoreLocation.coordinates	[Number]
mainStoreLocation.address	String
mainStoreLocation.description	String
storeLocations	(вкладення): [ ]
storeLocations.type	Number
storeLocations.coordinates	[Number]

Назва поля	*Тип Даних*
storeLocations.address	String
storeLocations.description	String
storeLocations.shopType	Number
personal	ObjectId (FK -> USER)

Відносини в таблиці:

PRODUCT до USER (особисті): відношення «багато до багатьох». Кожен продукт може мати кілька пов’язаних користувачів, і кожен користувач може бути пов’язаний з кількома продуктами. Особисте поле в таблиці Product — це масив значень ObjectId, які посилаються на пов’язаних користувачів.

Також таблиця має вбудовані документи: mainStoreLocation: представляє головне розташування магазину де знаходиться продукт. Це вбудований документ, який містить такі поля:

1) type: Рядок, що представляє тип географічних координат;

2) coordinates: Масив чисел, що позначає координати місця;

3) address: Рядок, що представляє адресу розташування;

4) description: Рядок, що описує розташування.

Поле "storeLocations" відображає кілька магазинів, де можна знайти продукт. Воно представлене у вигляді масиву вбудованих документів, які мають таку ж структуру, що й поле "mainStoreLocation". Це дозволяє зберігати детальну інформацію про кожен окремий магазин, включаючи його назву, адресу, географічні координати та інші відповідні дані. Масив вбудованих документів у полі "storeLocations" дозволяє гнучко організовувати дані про магазини.

Також ця найбільша таблиця включає в себе багато інших полів:

– name: представляє назву продукту;

– slug: представляє slug версію назви;

– durationAging: означає тривалість витримки продукту;

– weight: позначає вагу продукту;

– calories: Відображає інформацію про калорійність продукту;

– ratingsAverage: Представляє середні оцінки продукту;

– ratingsQuantity: Показує кількість оцінок для продукту;

– price: Відображає ціну товару;

– priceDiscount: Відображає ціну товару зі знижкою;

– summary: Представляє короткий опис продукту;

– description: Являє собою детальний опис товару;

– imageCover: Представляє зображення обкладинки продукту;

– images: Являє собою додаткові зображення товару;

– createdAt: Представляє мітку часу, коли продукт було створено;

– cheeseCreateDates: Представляє масив дат, пов’язаних зі створенням сиру;

– secretProduct: Вказує, на те чи є це секретним продуктом чи ні (логічне значення).

3.5.1 Бізнес процеси

Бізнес процеси будуть представлені у вигляді BPMN (модель і нотація бізнес процесів) — це стандарт графічного представлення для моделювання бізнес процесів. Він надає візуальний спосіб описати потік дій, рішень і взаємодій у бізнес-процесі. У цьому розділі ми надамо огляд і опис діаграми BPMN для програмного забезпечення.

Всі моделі BPMN будуть представлені в кресленниках А3. Кожна діаграма містить опис послідовності кроків, рішень та взаємодій між різними компонентами системи. Використання стандарту BPMN дозволяє розробнику та зацікавленим сторонам однозначно розуміти та спілкуватись щодо бізнес-процесів веб-застосунку.

Це сприяє ефективнішому плануванню, вдосконаленню та управлінню роботою з системою. Застосування BPMN-діаграм дозволить забезпечити структурованість, прозорість та взаєморозуміння в бізнес-процесах веб-застосунку, сприяючи успішному виконанню завдань та досягненню бізнес-цілей.

3.6 Опис структури та організації клієнтської частини (Front-end)

Для візуалізації роботи додатку архітектура клієнтської частини вебзастосунку інтернет-магазину сирів побудована за допомогою SSR (Server-Side Rendering), що дозволяє відрендерети сторінки на сервері та надсилати готовий HTML контент на браузер клієнта.

Отримавши дані, сервер будує веб-сторінку, використовуючи шаблони (templates). Шаблони - це HTML-документи, які містять певні місця заповнення даними. Вони включають динамічні елементи, які будуть заповнені з отриманих даних. Використанні, наприклад шаблон для окремого товару, де будуть підставлятися дані про назву, опис, ціну та зображення. Шаблон для відгуків та інші.

Після того, як сторінка буде побудована з використанням шаблону і заповнена даними, вона відправляється на браузер клієнта у вигляді готового HTML-коду. Браузер отримує цей HTML і відображає його, показуючи користувачу готову сторінку.

Оскільки браузер розуміє HTML, CSS та JavaScript (JS), то структура програмного забезпечення включає веб-файли з CSS для стилізації сторінок і JS для взаємодії з користувачем. Це використано для обробки дій користувача наприклад реєстрації, авторизації користувачів, для зміни даних в особистому кабінеті та інше.

Вся архітектура дозволяє забезпечити ефективний рендер веб-сторінок з мінімальним часом який користувачі проведуть в очікуванні. Вона також надає можливість зберігати і оновлювати дані про сири в базі даних, про відгуки, замовлення, та користувачів , що дозволяє динамічно оновлювати веб-сторінки з останньою інформацією про той чи інший документ.

Загалом, SSR є потужним підходом до розробки веб-застосунків, особливо для інтернет-магазинів, де швидкість відображення даних є критичним фактором успіху. Архітектура такого підходу показана на рисунку 3.6.

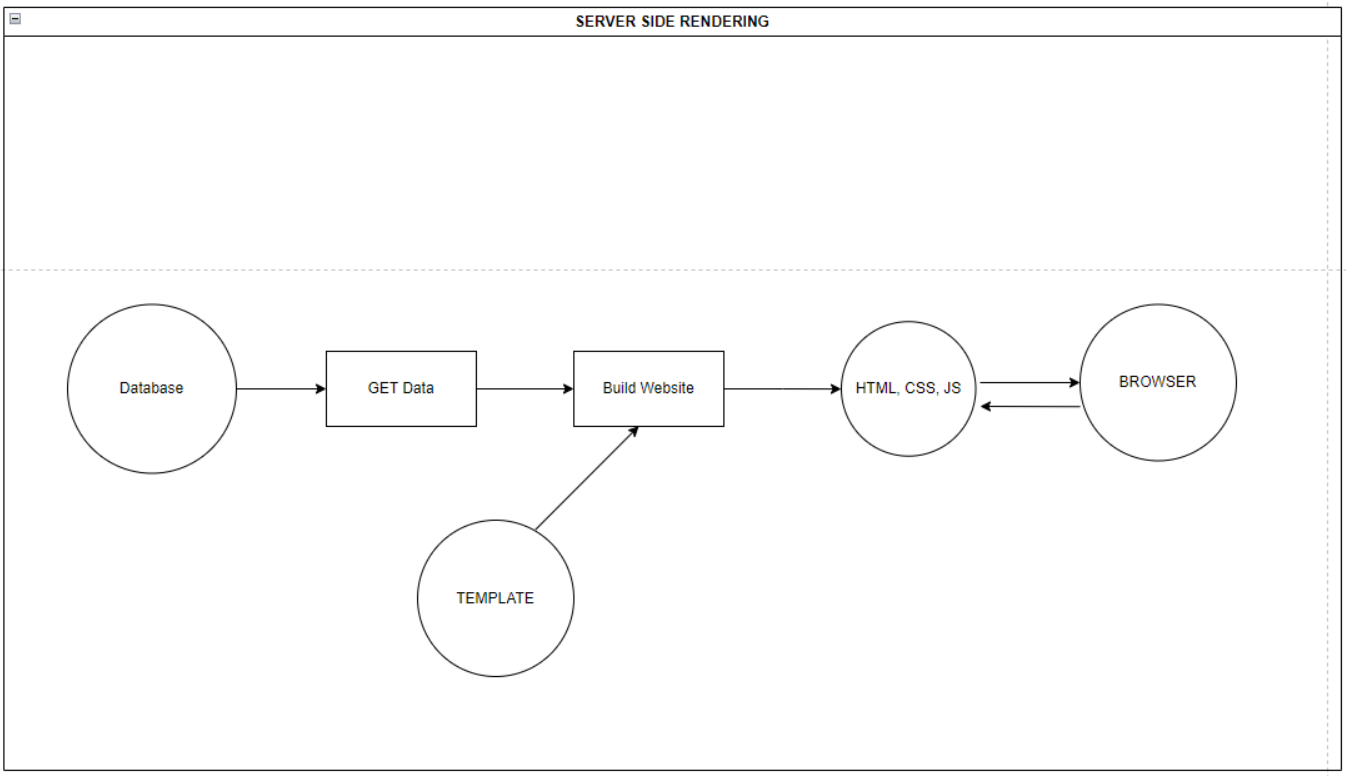

Рисунок 3.6 – Архітектура Front-end частини

Висновки до розділу

У третьому розділі дипломного проекту було виконано моделювання та конструювання програмного забезпечення, були розглянуті різні аспекти архітектури додатку та його компоненти.

У розділі були спочатку була розроблена системи ролей додатку, що дозволило чітко визначити функціональні можливості та обов'язки різних типів користувачів.

Далі була розглянута архітектура програмного забезпечення, де були визначені основні компоненти системи та залежності між ними. Процес розробки включав детальний аналіз та розуміння вимог інтернет магазину сирів, що дало змогу визначити ключові функціональні можливості, необхідні для ефективної роботи системи. На основі цього аналізу були розроблені алгоритми, які забезпечують правильну реалізацію функціональності системи. Були розроблені алгоритми для роботи з авторизацією користувачів, керуванням користувачами, продуктами, відгуками та замовленнями.

В розділі моделювання та конструювання програмного забезпечення було проведено детальний опис структури та організації бекенду (Back-end) системи. Зокрема, була розроблена база даних, яка забезпечує збереження і управління даними про користувачів, продукти, відгуки та замовлення. Це створює основу для зручної та ефективної роботи з даними в системі.

Також були розроблені бізнес-процеси веб-застосунку, що відображають послідовність дій та взаємодій у системі. Це дозволяє краще розуміти логіку роботи системи та взаємодію з користувачами. Розроблені бізнес-процеси стали основою для подальшої реалізації функціональності веб-застосунку.

Крім того, проведено опис структури та організації клієнтської частини (Front-end) веб-застосунку. Були визначені механізми взаємодії з користувачем, включаючи відображення продуктів, замовлень та відгуків. Це дозволяє забезпечити зручний та інтуїтивно зрозумілий інтерфейс для користувачів системи.

В цілому, розділ моделювання та конструювання програмного забезпечення виявився надзвичайно важливим. Він допоміг створити чітку архітектуру додатку, яка відповідає функціональним вимогам та забезпечує зручну та ефективну роботу для користувачів. Цей розділ є важливим кроком у розробці та реалізації вебзастосунку інтернет магазину сирів, оскільки він надає основу для подальшого програмування та розробки функціональності системи.

4 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

4.1 Впровадження інтерфейсу для користувача

Вся клієнтська частина веб-застосунку розроблялась за допомогою шаблонізатора PUG. Також використовувався CSS, як один із інструментів веб-розробки для стилізації та оформлення веб-сторінок.

Була розроблена сторінка Реєстрації, вона дозволяє новим користувачам створювати облікові записи та отримувати доступ до різноманітних функцій та можливостей, що пропонуються в додатку інтернет-магазині сирів. Форма представлена на рисунку 4.1.

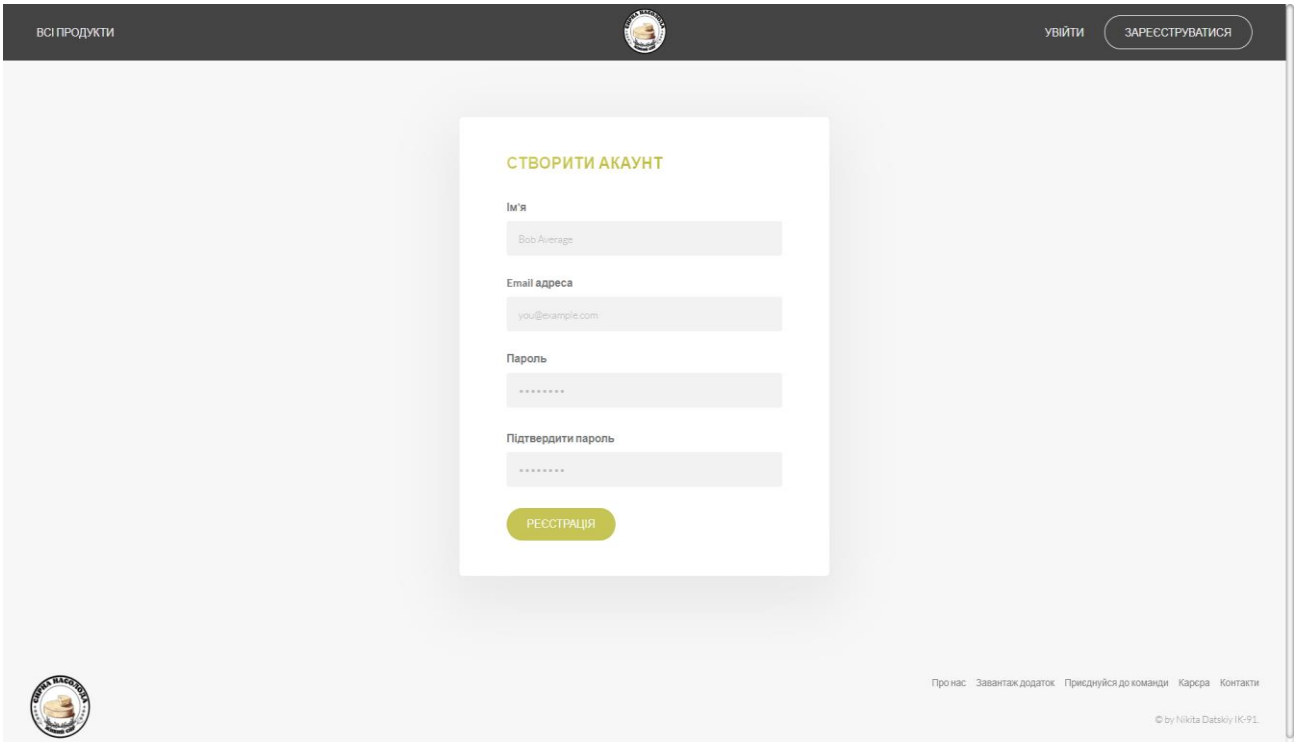

Рисунок 4.1 – Реєстрація користувача

Після того, як користувач ввів свої дані в форму логіну, ці дані були піддані процесу перевірки на відповідність збереженим даним користувачів у системі. В цьому процесі була проведена детальна перевірка правильності введеної електронної пошти користувача та відповідного пароля. Система перевірила, чи відповідають ці дані зареєстрованому користувачеві. В першу чергу, електронна пошта, введена користувачем, була перевірена.

Управління станом авторизації: Після успішної перевірки даних користувача, була створений JWT токен, що підтверджує авторизацію. Цей токен зберігався на стороні клієнта (у браузері), у відповідності до використаної архітектури. Застосовоні заходи безпеки для захисту маршрутів, які вимагають авторизації. До захищених маршрутів перевіряється наявність токену авторизації перед наданням доступу до захищених ресурсів або функцій.

Були враховані можливі помилки під час процесу авторизації, такі як неправильна електронна пошта або пароль, невалідний токен тощо. В таких випадках, користувачу відображаються відповідні повідомлення про помилку. Сторінка входу представлена на рисунок 4.2.

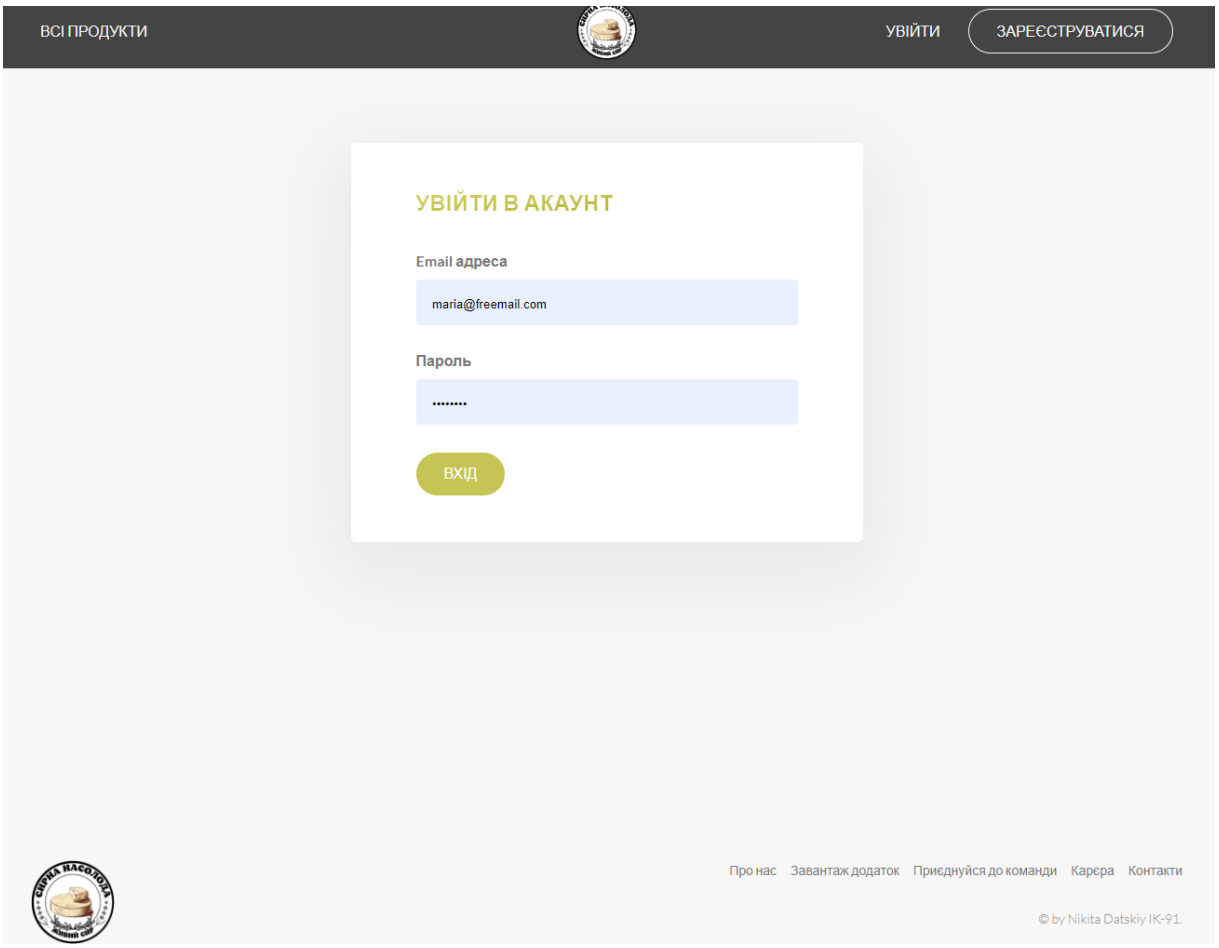

Рисунок 4.2 – Сторінка входу в додаток

Після успішного входу в додаток, користувачу відкривається доступ до кабінету, де він може здійснювати різні дії та отримувати персоналізовану інформацію.

Ось кілька основних функцій та можливостей, які доступні в кабінеті користувача.

Профіль користувача: В кабінеті користувач може переглядати та редагувати свій профіль. Це включає особисті дані, такі як ім'я, пошта, фото тощо. Користувач може оновлювати свої дані та зберігати їх для майбутнього використання. Замовлення: Користувач може переглядати свої замовлення, статус замовлення та історію покупок. Профіль користувача рисунок 4.3.

Відгуки та рейтинги: Користувач може переглядати свої відгуки які він залишив до товарів.

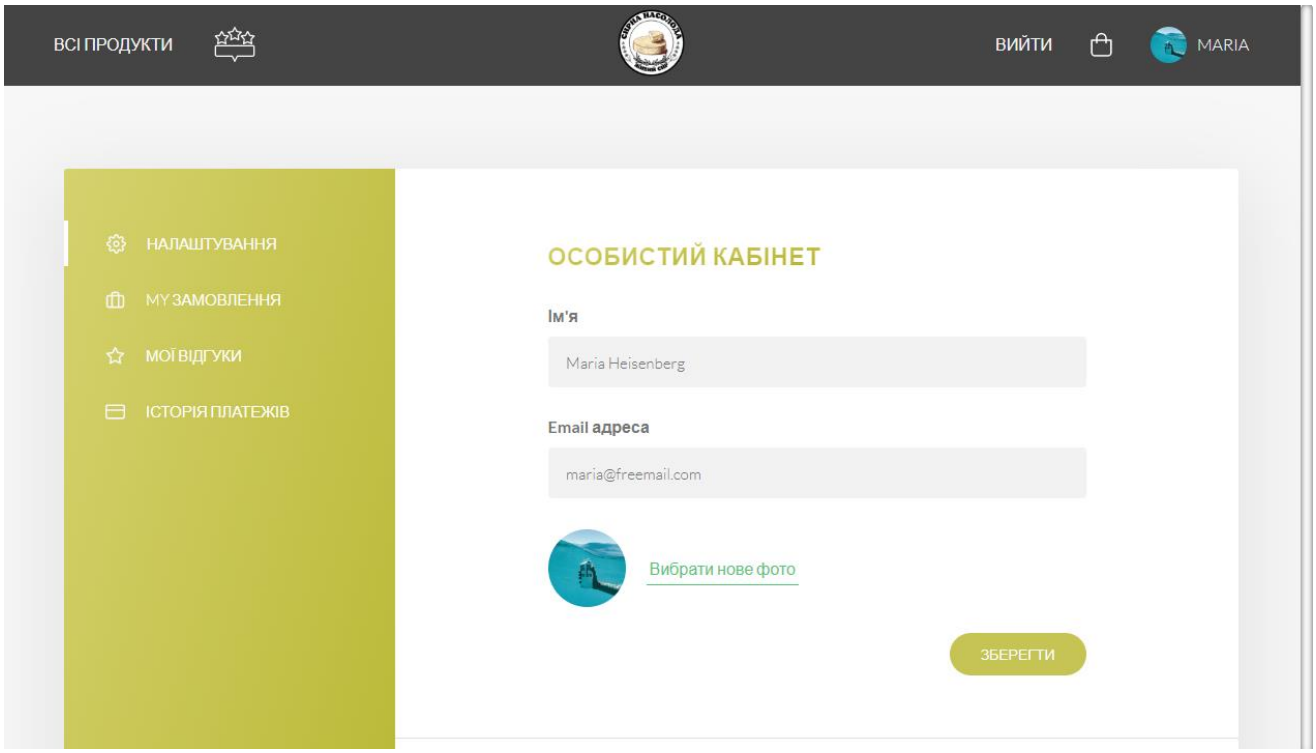

Рисунок 4.3 – Профіль користувача

Налаштування облікового запису: В кабінеті користувач може змінювати налаштування свого облікового запису, такі як пароль.

Це дозволяє користувачам контролювати свої особисті налаштування та забезпечувати безпеку свого облікового запису. Опція зміни пароля в особистому кабінеті користувача надає зручний інтерфейс для зміни паролю. Налаштування паролю в особистому кабінеті користувача додатку показано на рисунку 4.4.

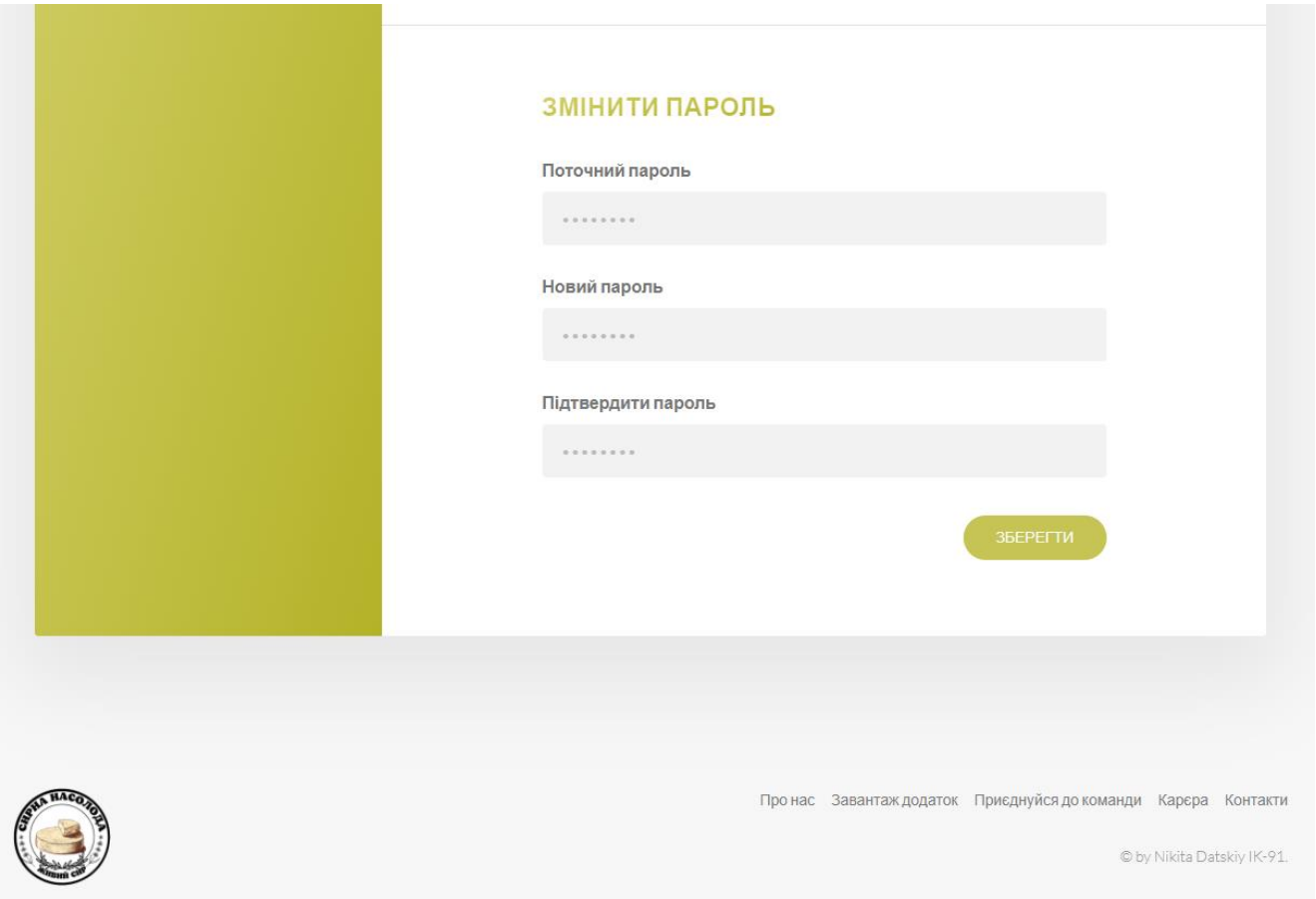

Рисунок 4.4 – Зміна Паролю на особистій сторінці

На головній сторінці показана візуалізація різних видів сирів, яка дозволяє швидко оглянути їх опис, характеристики та ціни. Зручний інтерфейс допоможе легко знайти сир, який відповідає смаку та вимогам. Це забезпечує зручний спосіб ознайомитись з асортиментом сирів та швидко знайти той, який відповідає смаку та вимогам клієнта. Крім того, головна сторінка надає вам доступ до особистого кабінету, де можна увійти до свого облікового запису.

В особистому кабінеті ж можливо переглядати та керувати своїми замовленнями, змінювати налаштування свого облікового запису та контролювати безпеку свого пароля. Головна сторінка на рисунку 4.5. Це місце, де клієнт може легко ознайомитись з продуктами, здійснити необхідні дії та знайти потрібну інформацію, забезпечує зручний та ефективний досвід для користувачів. Завдяки цій функціональності користувачі можуть впевнено керувати своїм обліковим записом та забезпечувати його безпеку.

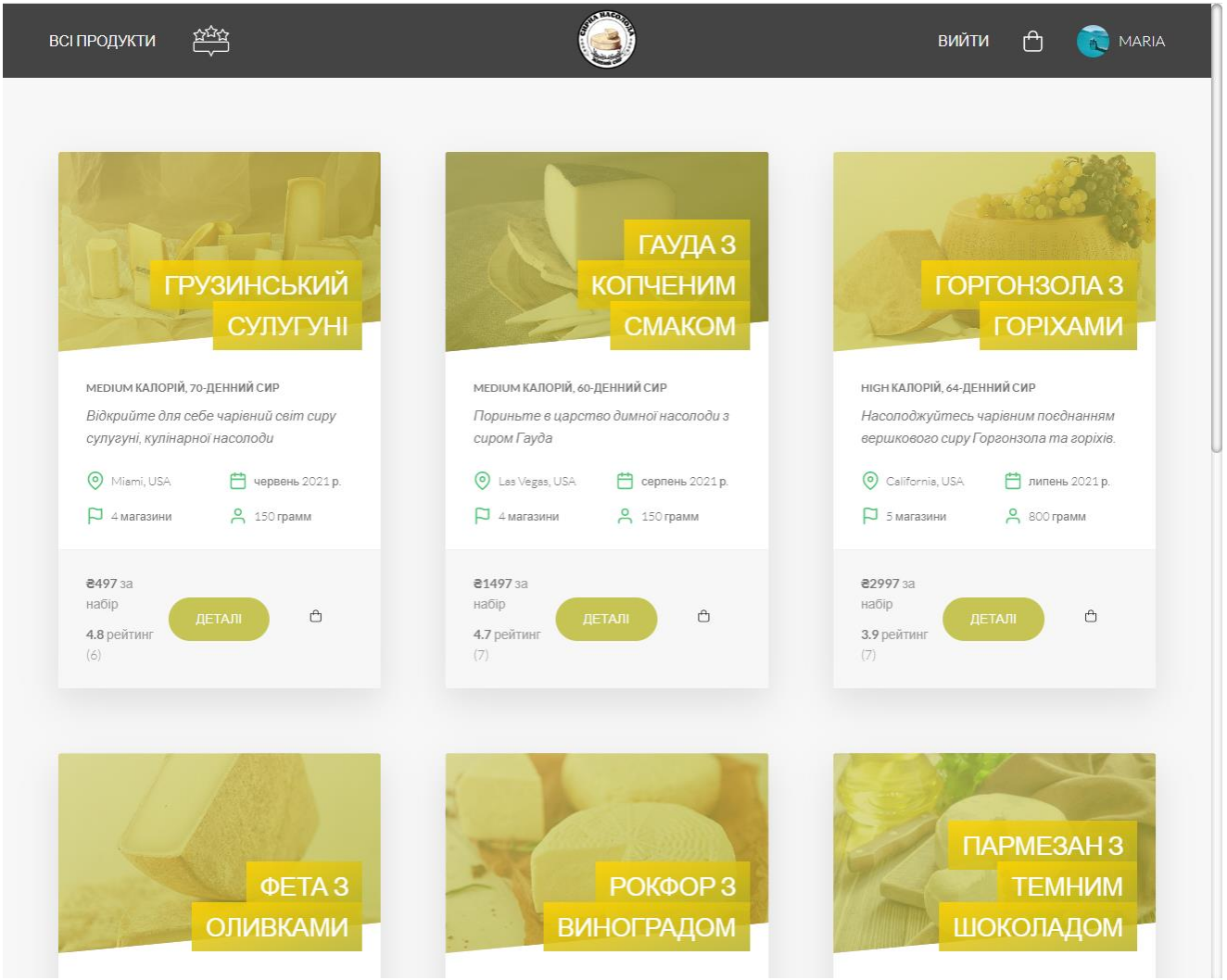

Рисунок 4.5 – Головна сторінка додатку

З головної сторінки є можливість перейти на сторінку конкретного товару, де буде надано детальний опис всіх характеристик та даних про цей товар. На цій сторінці ви зможете ознайомитись з усіма важливими відомостями про продукт.

З головної сторінки можна потрапити на сторінку конкретного товару, де буде детальний опис всіх характеристик, та даних. Рисунок 4.6. демонструє, що сторінка товару містить повну інформацію про обраний сир. Це дозволяє користувачам отримати всю необхідну інформацію перед придбанням товару, що сприяє прийняттю обґрунтованих рішень та задоволенню їх потреб. На сторінці конкретного товару також є можливість переглянути відгуки користувачів. Що надасть додаткову інформацію і думки інших людей, які можуть бути корисними при прийнятті рішення щодо покупки.

Рисунок 4.6 – Початок сторінки товару

На сторінці конкретного товару, як це представлено на рисунку 4.7, користувач має можливість ознайомитися з додатковими фотографіями товару, що дозволяє отримати більш повну картину щодо його зовнішнього вигляду та деталей.

Цей функціонал сприяє більш детальному розумінню характеристик товару та допомагає користувачам прийняти обґрунтовані рішення під час процесу покупки. Згідно потреб користувачів, сторінка товару також надає інтерактивну мапу, на якій відображені розташування магазинів, де можна придбати цей певний сир. Ця функціональність забезпечує зручну можливість знайти найближчі торгові точки та спрощує процес придбання продукту, дозволяючи користувачам здійснити покупку зручним та ефективним способом.

Рисунок 4.7 – Продовження сторінки товару

На нижній частині сторінки товару, як це проілюстровано на рисунку 4.8, розташовані відгуки від користувачів, що стосуються даного товару. Кожний відгук містить інформацію про автора, текст самого відгуку та рейтинг, на який користувач оцінив товар.

Додатково, на цій самій сторінці розташована кнопка "Придбати сир", яка надає можливість безпосередньо здійснити покупку обраного товару. Цей функціонал створює зручну та пряму можливість для користувачів здійснити покупку без необхідності переходити до інших сторінок або шукати додаткові опції.

Кнопка покупки спрощує процес оформлення замовлення та робить його доступним всього лише за кілька кліків, забезпечуючи зручність та ефективність для користувачів. Завдяки кнопці покупки користувачам не потрібно проходити довгий та складний процес оформлення замовлення.

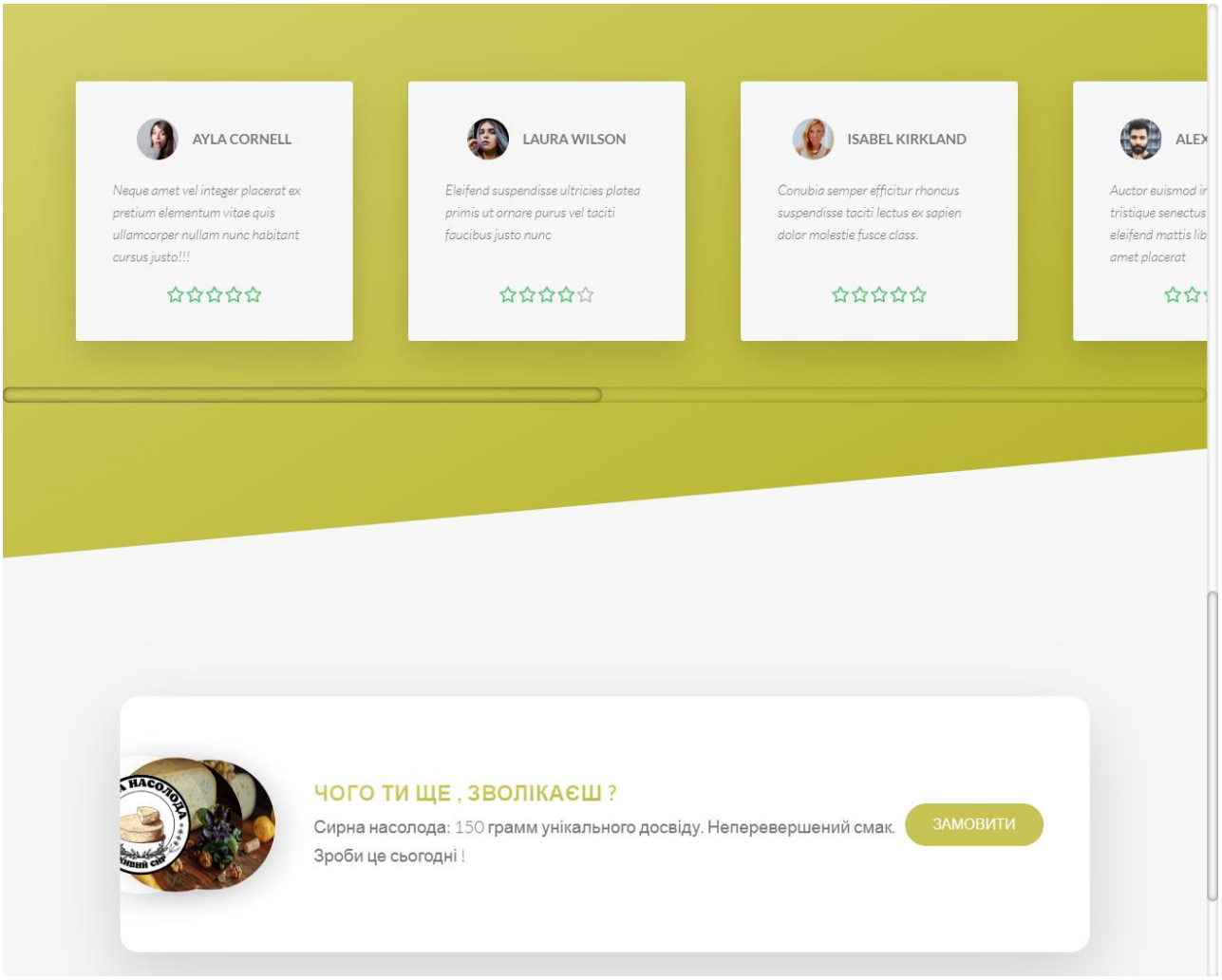

Рисунок 4.8 – Сторінка товару частина 3

Після натискання кнопки замовлення, користувач буде перенаправлений на інтегрований сервіс оплати Stripe, який гарантує високий рівень безпеки платежів за допомогою застосування сучасних технологій шифрування та захисту даних. Stripe відповідає найвищим стандартам безпеки в платіжній промисловості, забезпечуючи безпечні транзакції для користувачів. Цей сервіс підтримує різноманітні типи платежів, включаючи кредитні та дебетові картки, цифрові гаманці, мобільні платежі та інші. Використання Stripe дозволяє здійснювати платежі зручним і безпечним способом з використанням різних доступних опцій. На рисунку 4.8 показана сторінка оплати, яка візуально відображається під час проведення транзакції за допомогою Stripe. Цей момент є важливим, оскільки користувач може побачити та перевірити всі необхідні деталі свого платежу перед підтвердженням. Користувач може перевірити загальну суму, яку він повинен заплатити за продукт або послугу.

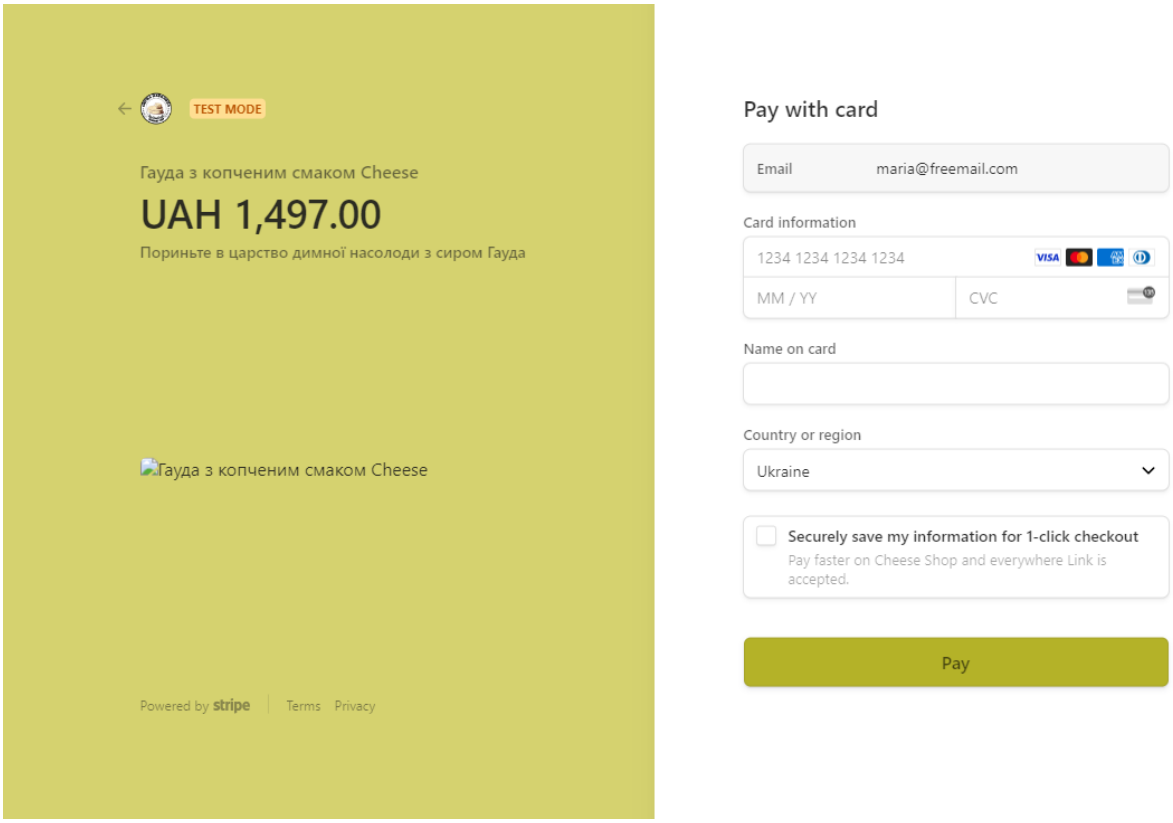

Рисунок 4.9 – Сторінка оплати Stripe

Після успішної оплати, замовлений сир буде автоматично відображений на вкладці "Мої товари". Це надає зручну можливість для користувачів відстежувати та переглядати свої замовлені товари безпосередньо після оплати. На вкладці "Історія платежів" буде відображатися список попередніх платежів, який містить різну додаткову інформацію, включаючи статус оплати. Вона надає зручний огляд попередніх транзакцій, що дозволяє користувачам відстежувати свою фінансову історію та переконатися в успішності оплати. Сторінка історія платежів на рисунок 4.10. Кожен запис в історії платежів буде містити такі дані:

1) ідентифікаційний номер транзакції;

2) ціна товару;

3) назва товару;

4) дату оплати;

5)статус;

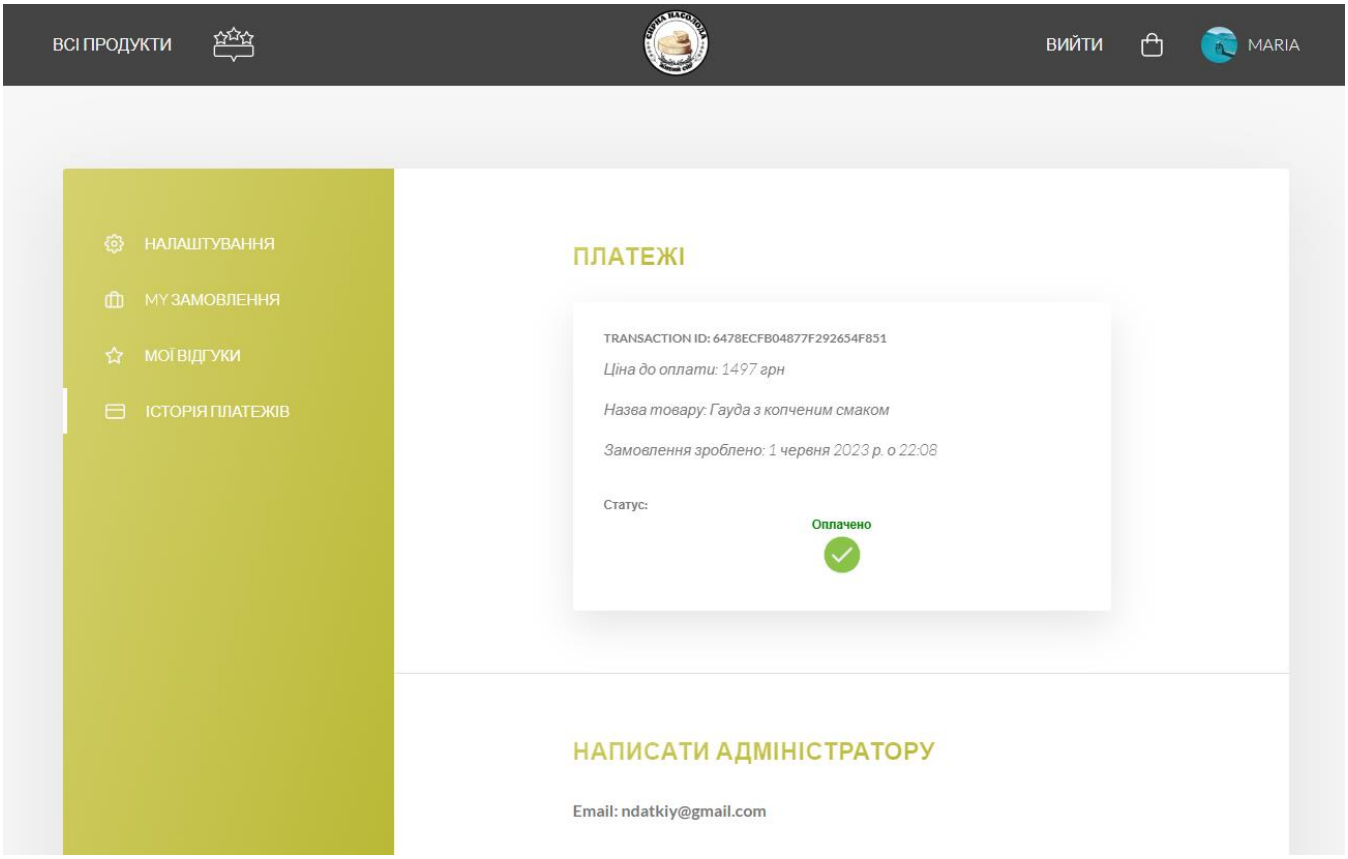

Рисунок 4.10 – Сторінка платежів

Сторінка “мої відгуки” містить всі відгуки користувача, які він робив до різних продуктів, зображено на рисунку 4.11.

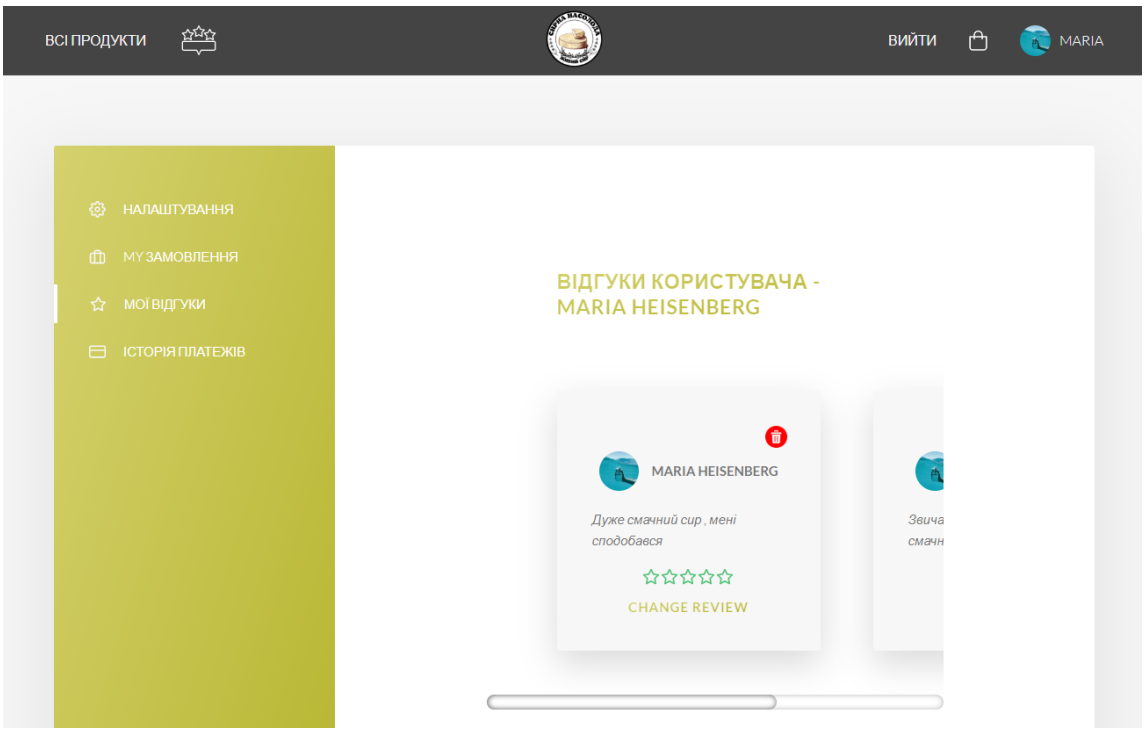

Рисунок 4.11 – Відгуки користувача

4.2 Тестування роботи веб-застосунку

Для перевірки роботи застосунку я використав Postman, для перевірки Email листів MailTrap сервіс.

Postman - це популярний інструмент для тестування API, який дозволяє легко взаємодіяти з веб-застосунками та перевіряти їх функціональність. Використовуючи Postman, можна створювати та відправляти HTTP-запити до сервера веб-застосунку, перевіряти відповіді сервера та перевіряти, чи працюють різні функції вашого додатку належним чином.

Крім того, для перевірки відправки та отримання електронних листів був використаний інструмент MailTrap. MailTrap – це сервіс, який дозволяє створювати віртуальну поштову скриньку для тестування електронних листів. Для проекту я налаштував ваш веб-застосунок таким чином, щоб він відправляв електронні листи на адреси, які створені в MailTrap. Це дозволяє перевірити, чи правильно відбувається відправка листів, переглядати їх вміст та переконуватися, що система електронної пошти функціонує належним чином.

Всього буде протестовано 6 папок з різними API:

1) Users;

2) Auth;

3) Products;

4) Reviews;

5) Product/Review;

6) Order.

Запити пов’язані з Користувачами.

Має в собі 7 різних HTTP запитів до сервера :

Тест 1: виконує GET запит до ресурсу “{{URL}}api/v1/users”, який повертає список всіх користувачів. Для успішного доступу до цього запиту необхідна роль “admin”, що підтверджується коректним токеном аутентифікації. Цей запит є корисним для адміністраторів системи, які можуть керувати списком користувачів. Результат при успішному запиті та при помилці таблиця 4.1.

Таблиця 4.1 – Результат тестування запиту 1

Статус Success	Статус Fail
Status code 200	Статус код 403
Отримання списку всіх користувачів	Ви не маєте дозволу на виконання цієї дії

Тест 2: GET запит “{{URL}}api/v1/users/(id користувача)” повертає інформацію про одного користувача , для доступу потрібна роль admin. Результат при успішному запиті та при помилці таблиця 4.2.

Таблиця 4.2 – Результат тестування запиту 2

Статус Success	Статус Fail
Status code 200	Статус код 403
Отримання даних про користувача	Ви не маєте дозволу на виконання цієї дії

Тест 3: GET запит “{{URL}}api/v1/users/me” призначений для отримання інформації про поточного авторизованого користувача. Цей запит вимагає, щоб користувач був увійшов в систему для доступу до своїх особистих даних. Запит дозволяє користувачеві отримати доступ до власних персональних даних та переглянути їх. Результат в таблиці 4.3.

Таблиця 4.3 – Результат тестування запиту 3

Статус Success	Статус Fail
Status code 200	Статус код 401
Отримання даних про користувача	Ви не авторизовані

Тест 4: PATCH запит “{{URL}}api/v1/users/(id користувача)” оновлює інформації користувача. В тілі запиту вказати які дані будуть змінені, для доступу потрібна роль admin. Результат при успішному запиті та при помилці таблиця 4.4.

Таблиця 4.4 – Результат тестування запиту 4

Статус Success	Статус Fail
Status code 200	Статус код 403
Дані успішно оновлені	Ви не маєте дозволу на виконання цієї дії

Тест 5: PATCH запит “{{URL}}api/v1/users/updateMe” призначений для оновлення даних поточного авторизованого користувача. Цей запит дозволяє змінювати конкретні дані користувача шляхом вказання в тілі запиту змінених полів. Для доступу до цього запиту необхідно авторизуватись, щоб переконатись, що тільки автентифікованим користувачам надається право змінювати свої дані. Результат при успішному запиті та при помилці таблиця 4.5.

Таблиця 4.5 – Результат тестування запиту 5

Статус Success	Статус Fail
Status code 200	Статус код 401
Дані успішно оновлені	Ви не авторизовані

Тест 6: DELETE запит “{{URL}}api/v1/users/(id користувача)” використовується для видалення конкретного користувача з системи. Цей запит дозволяє адміністраторам видаляти користувачів із зазначеним ідентифікатором. Результат при успішному запиті та при помилці таблиця 4.6.

Таблиця 4.6 – Результат тестування запиту 6

Статус Success	Статус Fail
Status code 204	Статус код 403
Видалено	Ви не маєте дозволу на виконання цієї дії

Тест 7: DELETE запит “{{URL}}api/v1/users/deleteMe” використовується для деактивації (відключення) облікового запису поточного користувача. Цей запит дозволяє користувачеві деактивувати свій власний обліковий запис. Для доступу до цього запиту необхідно бути авторизованим, тобто мати дійсний токен автентифікації, що підтверджує ідентифікацію користувача. Результат при успішному запиті та при помилці таблиця 4.7.

Таблиця 4.7 – Результат тестування запиту 7

Статус Success	Статус Fail
Status code 204	Статус код 403
Видалено	Ви не маєте дозволу на виконання цієї дії

Запити пов’язані з автентифікацією , всього мають 5 різних HTTP запитів до сервера:

Тест 8: POST запит “{{URL}}api/v1/users/signup” створення нового користувача, в тілі запиту вказати всі дані для реєстрації. Результат при успішному запиті та при помилці таблиця 4.8.

Таблиця 4.8 – Результат тестування запиту 8

Статус Success	Статус Fail
Status code 201	Статус код 500
Створено	Такий користувач вже існує

Для перевірки отримання листів , був використаний сервіс MailTrap.

Після реєстрації користувача з вказаною ним електронною поштою, він отримує повідомлення, яке можна переглянути на рисунку 4.11.

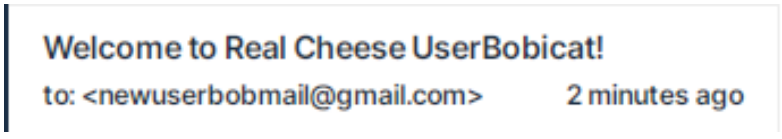

Рисунок 4.11. – Повідомлення для нового користувача

В тілі повідомлення запрошення до особистого кабінету, показано на рисунку 4.12.

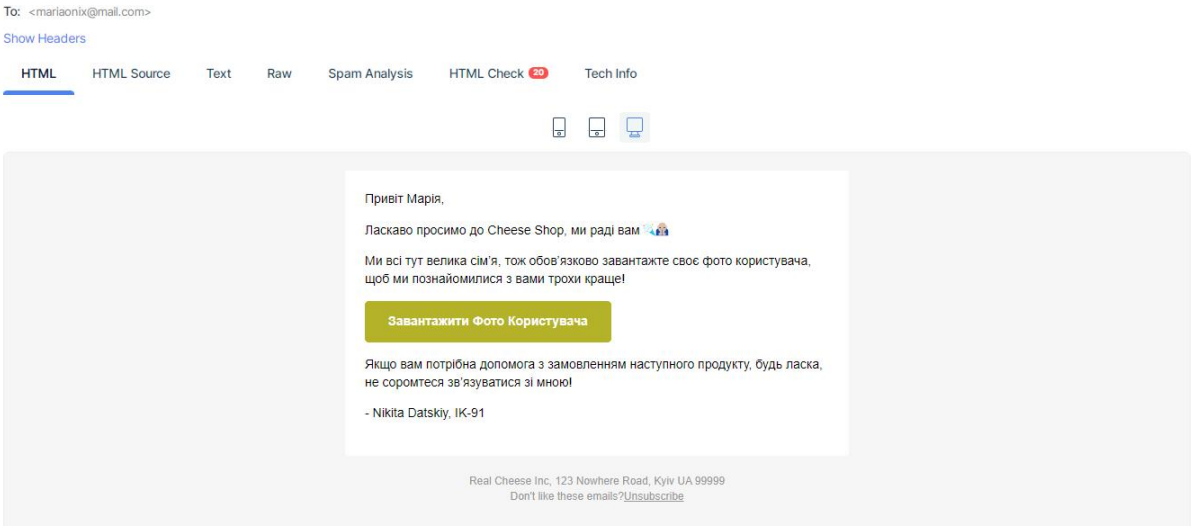

Рисунок 4.12 – запрошення до особистого кабінету

Тест 9: POST запит “{{URL}}api/v1/users/login” авторизація користувача, в тілі запиту вказати всі дані для входу. Результат при успішному запиті та при помилці таблиця 4.9.

Таблиця 4.9 – Результат тестування запиту 9

Статус Success	Статус Fail
Status code 200	Статус код 401
Користувач успішно авторизований	Неправильний пароль або пошта

Тест 10: PATCH запит “{{URL}}api/v1/users/ updateMyPassword”. Для успішного виконання запиту користувач повинен бути авторизованим. В тілі запиту передаються необхідні дані для зміни пароля. Результат при успішному запиті та при помилці таблиця 4.10.

Таблиця 4.10 – Результат тестування запиту 10

Статус Success	Статус Fail
Status code 200	Статус код 500
Пароль оновлено	Недійсний або підроблений токен

Тест 11: POST запит “{{URL}}api/v1/users/forgotPassword” запит на відновлення забутого паролю, при успішному запиті користувач отримує лист з токеном, що дозволяє оновити пароль. Помилки включають в себе випадки, коли користувач не знайдений, неактивований або вказана електронна пошта має неправильний формат. Результат при успішному запиті та при помилці таблиця 4.11.

Таблиця 4.11 – Результат тестування запиту 11

Статус Success	Статус Fail
Status code 200	Статус код 404
Токен успішно відправлено на пошту	Такої пошти не існує

За допомогою MailTrap перевіряємо лист який прийшов на пошту, рисунок 4.13. У повідомленні міститься посилання, яке веде на сторінку для відновлення паролю. Рисунок 4.14 відображає знімок екрана листа, отриманого через сервіс MailTrap. В тілі повідомлення буде присутнє посилання, з яким можна відновити пароль.

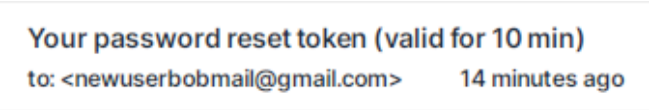

Рисунок 4.13 – Вигляд теми листа

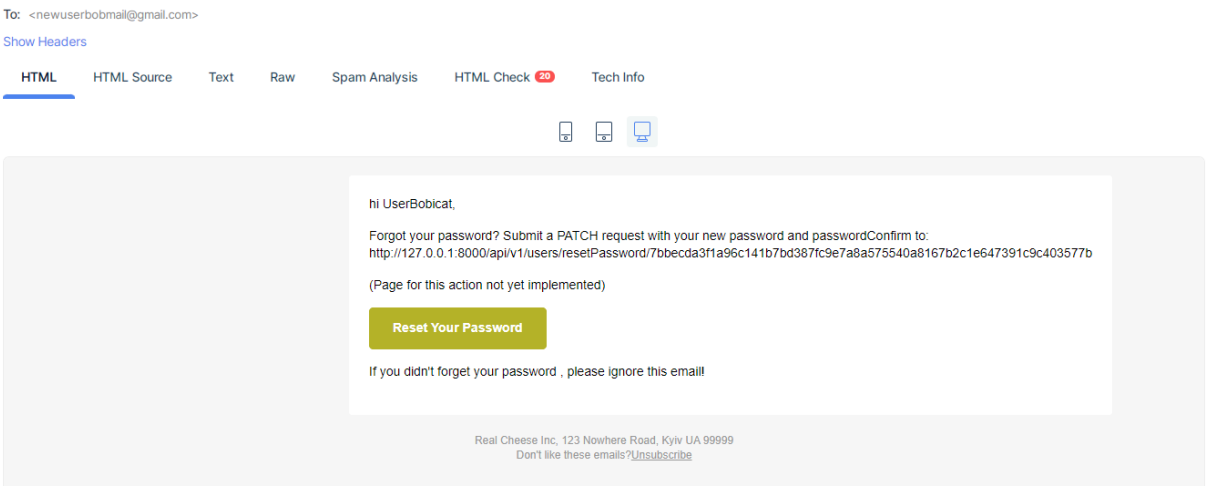

Рисунок 4.14 – Повідомлення про відновлення паролю

Тест 12: PATCH запит “{{URL}}api/v1/users/resetPassword/(Токен відновлення паролю)” використовується для відновлення забутого паролю. У тілі запиту необхідно вказати новий пароль та його підтвердження. При успішному запиті і вірному токені, система оновлює пароль користувача. Результати тестування цього запиту наведені в таблиці 4.12, яка відображає статус запиту і повідомлення про успішне або невдале відновлення паролю.

Таблиця 4.12 – Результат тестування запиту 12

Статус Success	Статус Fail
Status code 200	Статус код 400
Пароль змінено	Токен відновлення не валідний

Запити пов’язані з продуктами включають в собі 9 різних HTTP запитів до сервера:

Тест 13: GET запит “{{URL}}api/v1/products” використовується для отримання списку всіх продуктів. Запит також підтримує можливість фільтрування продуктів за різними параметрами. Наприклад, можна застосовувати фільтр за ціною (price), де значення менше (lt) 1000, і фільтр за середнім рейтингом (ratingsAverage), де значення більше або дорівнює (gte) 4.7. Крім того, можна налаштовувати вивід результатів по частинах, наприклад, по 3 товари на сторінку, загалом буде 3 сторінки з результатами. Результати тестування запиту наведені в таблиці 4.13, яка містить інформацію про статус запиту і повідомлення про успішне виконання запиту або помилку, якщо така виникла.

Таблиця 4.13 – Результат тестування запиту 13

Статус Success	Статус Fail
Status code 200	Статус код 400
Отримання списку всіх продуктів	Помилка серверу

Тест 14: POST запит “{{URL}}api/v1/products” використовується для створення нового продукту. Цей запит доступний для ролей "admin" і "chief-cheese-maker". У тілі запиту необхідно вказати всі необхідні дані для створення продукту.

Результати тестування запиту наведені в таблиці 4.14. Таблиця включає інформацію про статус запиту і повідомлення про успішне створення продукту або помилку, якщо така виникла.

Таблиця 4.14 – Результат тестування запиту 14

Статус Success	Статус Fail
Status code 200	Статус код 403
Продукт створено	Ви не маєте дозволу на виконання цієї дії

Також в тестуванні 15 були перевірені такі запити:

1) GET запит “{{URL}}api/v1/products/(Id продукту)” повертає результат, що містить детальну інформацію про продукт, якщо запит виконано успішно.

2) DELETE запит “{{URL}}api/v1/products/(Id продукту)”, якщо запит виконано успішно, то продукт буде видалений з системи.

3) PATCH запит “{{URL}}api/v1/products/(Id продукту)” використовується для оновлення конкретного продукту за його ідентифікатором (Id). Запит також надає можливість завантажувати фотографії продукту, які можуть бути змінені за розміром до потреб програми.

Також в тестуванні 16 перевірені запити:

GET {{URL}}api/v1/products/product-stats – повертає статистику по продуктам

GET{{URL}}api/v1/products/productswithin/(дальність)/center/(координати)/unit/(міра довжини) – Фільтрує продукти які продаються в певному радіусі від заданих координат в кілометрах або милях.

GET {{URL}}api/v1/products/distances/(координати)/unit/(міра довжини) – В

результаті запиту отримуємо відстань до місця де можна купити продукти від заданих координат , можливий розрахунок як в кілометрах так і в милях.

Запити пов’язані з відгуками складаються з 5 різних HTTP запитів до сервера :

1) GET запит “{{URL}}api/v1/reviews”, отримання списку всіх відгуків;

2) GET запит “{{URL}}api/v1/reviews/(id відгуку)”, отримання конкретного відгуку;

3) POST запит “{{URL}}api/v1/reviews/”, створення відгуку, потрібна роль user;

4) PATCH запит “{{URL}}api/v1/reviews/(id відгуку)”, використовується для оновлення існуючого відгуку в системі. Цей запит доступний як для користувачів з роллю "user", так і для адміністраторів з роллю "admin";

5) DELETE запит “{{URL}}api/v1/reviews/(id відгуку)”, використовується для видалення існуючого відгуку з системи. Цей запит доступний як для користувачів з роллю "user", так і для адміністраторів з роллю "admin".

Вкладені запити продукт/відгук включають 2 різних HTTP запитів до сервера:

1) GET запит “{{URL}}api/v1/products/(id продукту)/reviews”, отримання всіх відгуків до певного продукту;

2) POST запит “{{URL}}api/v1/products/(id продукту)/reviews”, залишити відгук до певного продукту.

Запити пов’язані з замовленням включають 6 різних HTTP запитів до сервера:

1) GET запит “{{URL}}api/v1/booking/checkout-session/(id товару)”, використовується для отримання сторінки оплати з використанням сервісу Stripe для конкретного товару за його ідентифікатором (id);

2) GET запит “{{URL}}api/v1/booking/”, використовується для отримання всіх замовлень з системи. Цей запит призначений для адміністратора системи і надає доступ до інформації про всі замовлення, сервер повертає список замовлень з системи, який містить детальну інформацію про кожне замовлення;

3) GET запит “{{URL}}api/v1/booking/(id замовлення)”, Запит на отримання конкретного замовлення;

4) POST запит “{{URL}}api/v1/booking/”, використовується для створення нового замовлення. Цей запит дозволяє адміністратору створити замовлення, наприклад якщо товар куплять за межами сервісу Stripe;

5) PATCH запит “{{URL}}api/v1/booking/(id замовлення)”, оновлення відгуку, роль admin;

6) DELETE запит “{{URL}}api/v1/booking/(id замовлення)”, використовується для видалення конкретного замовлення з системи. Цей запит доступний тільки для користувачів з роллю admin.

В результаті перевірки запитів до різних API-розділів (Users, Auth, Products, Reviews, Product/Review, Order) було підтверджено наступні висновки: запити до розділу Users, Auth, Products, Reviews, Order були успішно виконані.

Загалом, всі перевірені API-розділи працюють відповідно до очікувань, надаючи правильні дані та реагуючи на запити з відповідними повідомленнями про успіх або помилку.

Тестування веб-застосунку виявилося незамінним етапом, який дав змогу переконатися в належній роботі всіх його функцій. Цей процес не тільки сприяє виявленню можливих проблем, але й надає можливість їх виправити до впровадження програмного забезпечення в реальному середовищі. Значення тестування веб-застосунків неможливо недооцінити, оскільки воно забезпечує якість, надійність і відповідність запланованим функціональним вимогам.

Під час тестування було приділено особливу увагу виявленню можливих проблем та помилок, які були виправлені до фінального впровадження веб-застосунку. Цей підхід дозволяє підвищити якість та надійність програмного забезпечення, забезпечуючи користувачам стабільну та безперебійну роботу застосунку.

Висновки до розділу

У четвертому розділі дипломного проекту було розроблено інтерфейс користувача для веб-застосунку. Було використано шаблонізатор PUG та CSS для стилізації та оформлення веб-сторінок. Були реалізовані реєстрація та авторизація користувачів, а також створено особистий кабінет з можливістю зміни налаштувань облікового запису. Було створено сторінку з сиром, на якій була відображена додаткова інформація, фото, карта з магазинами, коментарі та можливість купити.

У розділі "Тестування роботи веб-застосунку" було успішно виконано перевірку різних API-розділів, зокрема Users, Auth, Products, Reviews, Product/Review та Order. Були перевірені різні типи запитів та їхні відповіді, а також обробка помилок. Тестування підтвердило правильну роботу функціональності веб-застосунку, а також виявило можливі проблеми та помилки, які були виправлені.

В цілому, розділ "Тестування роботи веб-застосунку" дозволив підтвердити правильну функціональність різних API-розділів та виявити та виправити можливі проблеми. Розділ "Впровадження інтерфейсу для користувача" дозволив розробити зручний та привабливий інтерфейс для користувачів, що задовольняє їхні потреби. Обидва розділи є важливими етапами в розробці та реалізації веб-застосунку інтернет-магазину сирів.

Крім того, розділи "Тестування роботи веб-застосунку" та "Впровадження інтерфейсу для користувача" взаємодоповнюють один одного, створюючи надійну та зручну платформу для користувачів. Тестування допомагає перевірити функціональність та надійність різних компонентів системи, тоді як розробка інтерфейсу забезпечує приємний та привабливий користувацький досвід. Разом ці розділи гармонійно працюють разом, сприяючи успішному впровадженню веб-застосунку інтернет магазину сирів та задоволенню потреб користувачів.

ВИСНОВКИ

Під час розробки дипломного проекту були здійснені різні етапи, такі як розробка інтерфейсу користувача, тестування продуктивності вебзастосунку, моделювання та проектування програмного забезпечення, аналіз та вибір засобів розробки. Кожен із цих етапів відіграв свою роль у створенні веб-додатку онлайн-магазину сиру.

Розділ «Впровадження інтерфейсу користувача» дозволив створити зручний та привабливий для користувачів інтерфейс. Приділялась увага до дизайну, навігації та функціональних можливостей, щоб забезпечити зручність використання та привабливість веб-додатку.

Розділ «Тестування продуктивності веб-додатків» підтвердив правильну роботу різних розділів API, а також виявив і усунув потенційні проблеми. Цей етап допоміг підтвердити ефективність та стабільність розробленої системи, а також виявити та усунути можливі недоліки.

Розділ «Моделювання та проектування програмного забезпечення» був спрямований на створення чіткої архітектури програми, що враховує функціональні вимоги та забезпечує зручну та ефективну роботу користувачів. Цей етап включав визначення структури бази даних, розробку моделей та компонентів системи, а також встановлення логічних зв'язків між ними.

Розділ «Аналіз та вибір засобів розробки» допоміг визначити оптимальні мови програмування, фреймворки та СУБД для проекту. Аналізувалися переваги та недоліки різних технологій, з метою вибору найбільш підходящих для веб-додатку сирів.

Разом ці етапи зачепили повний цикл розробки веб-додатків, починаючи з розробки інтерфейсу та функціональності, перевірки їх роботи, архітектури системи та вибору необхідних засобів розробки. Кожен із цих етапів був важливий для успішної реалізації веб-застосунку інтернет-магазин сирів та забезпечення заданих функцій і вимог.

ДЖЕРЕЛА

1. Node.js v20.2.0 documentation. URL: https://nodejs.org/docs/latest-v20.x/api/ (дата звернення: 05.06.2023).

2.Express.js documentation. Express. URL: https://expressjs.com/en/5x/api.html (дата звернення: 05.06.2023).

3.PUG documentation. URL: https://pugjs.org/api/getting-started.html (дата звернення: 05.06.2023).

4.MongoDB Node Driver. URL: https://www.mongodb.com/docs/drivers/node/current/ (дата звернення: 05.06.2023).

5.Mongoose documentation. URL: https://mongoosejs.com/docs/guide.html (дата звернення: 05.06.2023).

6.Axios documentation. URL: https://axios-http.com/docs/intro (дата звернення: 05.06.2023).

7.Postman documentation. URL: https://learning.postman.com/docs/introduction/overview/ (дата звернення: 05.06.2023).

8.JSON Web Tokens (JWT) documentation.URL: https://jwt.io/introduction/ (дата звернення: 05.06.2023).

9.User registration / Alex R. Young, Bradley Meck, Mike Cantelon, Tim Oxley, Marc Harter, Nathan Rajlich, T.J. Holowaychuk: Manning; 2nd edition (September 17, 2017) : Node.js in Action, p. 179-185

10.Bcrypt Hash. URL: Доступно: https://www.npmjs.com/package/bcrypt (дата звернення: 05.06.2023).

11. Cross-Origin Resource Sharing (CORS). URL: https://developer.mozilla.org/ru/docs/Web/HTTP/CORS (дата звернення: 06.06.2023).

12. Helmet.js documentation. URL: Доступно: https://helmetjs.github.io/ (дата звернення: 06.06.2023).

13. World Cross Site Scripting Examples. URL: https://websitesecuritystore.com/blog/real-world-cross-site-scripting-examples/ (дата звернення: 06.06.2023).
14. Prevent Parameter Pollution in Node.JS.URL: https://levelup.gitconnected.com/prevent-parameter-pollution-in-node-js-f0794b4650d2 (дата звернення: 06.06.2023).
15. Mark Richards, Neal Ford. 2020. Fundamentals of Software Architecture: An Engineering Approach.
16. Nikhil Marathe. An Introduction to libuv. 2015. p. 9
17. Common web application architectures. URL: https://learn.microsoft.com/en-us/dotnet/architecture/modern-web-apps-azure/common-web-application-architectures (дата звернення: 06.06.2023).
18. JavaScript Backend basics. URL: https://www.geeksforgeeks.org/javascript-backend-basics/ (дата звернення: 06.06.2023).
19. Relationship Between Node.Js and V8: A Complete Overview. URL: https://www.knowledgehut.com/blog/web-development/nodejs-and-v8 (дата звернення: 06.06.2023).