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

Факультет інформатики та обчислювальної техніки

Кафедра інформаційних систем та технологій

«На правах рукопису»
УДК 004

До захисту допущено:
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____2024 р.

Магістерська дисертація

**на здобуття ступеня магістра
за освітньо-професійною програмою «Інформаційні управляючі системи та технології»
зі спеціальності 126 «Інформаційні системи та технології»
на тему: «Веб-додаток для ведення ювелірного бізнесу»**

Виконав:
студент VI курсу, групи ІС-23мп
Мисак Олександр Кирилович _____

Керівник:
Доцент кафедри ІСТ, к.т.н., доц.
Ткач Михайло Мартинович _____

Рецензент:
Доцент кафедри ІПІ, к.т.н., доц.
Лісовиченко Олег Іванович _____

Засвідчую, що у цій магістерській дисертації немає запозичень з праць інших авторів без відповідних посилань.
Студент _____

Київ – 2024 року

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

Рівень вищої освіти – другий (магістерський)

Спеціальність – 126 «Інформаційні системи та технології»

Освітньо-професійна програма

«Інформаційні управляючі системи та технології»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____2024 р.

ЗАВДАННЯ
на магістерську дисертацію студенту

Мисаку Олександру Кириловичу

1. Тема дисертації «Веб-додаток для ведення ювелірного бізнесу», науковий керівник дисертації Ткач Михайло Мартинович, к.т.н., доцент, затверджені наказом по університету від «07» 11 2023 р. № 5168-с.

2. Термін подання студентом дисертації «08» 01 2024 р

3. Об’єкт дослідження побудова гнучкої системи керування ювелірним бізнесом з можливістю взаємодії з контентом та клієнтами.

4. Вихідні дані автоматизована система адміністрування та надання послуг в сфері ювелірного бізнесу.

5. Перелік завдань, які потрібно розробити дослідити предметну область та проаналізувати існуючі рішення, на основі дослідження сформувати вимоги до системи, обрати технології для розробки, розробити архітектуру системи, розробити структурну схему системи, розробити структуру для системи керування контентом, розробити інтерфейс користувача, розробити стартап-проект.

6. Орієнтовний перелік графічного (ілюстративного) матеріалу діаграма варіантів використання, блок-схема користувача, загальна структурна схема системи, структурна система серверної частини, інфологічна модель системи керування контентом, ER-діаграма.

7. Орієнтовний перелік публікацій

8. Дата видачі завдання 01.09.2023 р.

Календарний план

№ з/п	Назва етапів виконання магістерської дисертації	Термін виконання етапів магістерської дисертації	Примітка
1	Видача завдання	01.09.2023 р.	
2	Аналіз предметної області та формування вимог	21.09.2023 р.	
3	Розроблення діаграми використання	28.09.2023 р.	
4	Вибір технологій	04.10.2023 р.	
5	Розроблення структурної схеми системи	11.10.2023 р.	
6	Розроблення ER-діаграм	18.11.2023 р.	
7	Розроблення системи керування контентом	25.11.2023 р.	
8	Розроблення інтерфейсу користувача	09.11.2023 р.	
9	Оформлення дисертації	14.12.2023 р.	
10	Подання до захисту	18.12.2023 р.	

Студент Олександр МИСАК

Науковий керівник Михайло ТКАЧ

РЕФЕРАТ

Мисак О.К. Веб-додаток для ведення ювелірного бізнесу. «КПІ ім. Ігоря Сікорського», Київ, 2024

Дисертація складається з 112 сторінок, 45 рисунків, 52 таблиць, 19 літературних джерел та 9 додатків.

Метою даної магістерської дисертації є розробка інформаційної системи, яка дозволяє користувачу керувати всім змістом додатку, що забезпечує розширюваність. Додаток підвищить ефективність та легкість у налаштовуванні, зборі даних та підтримці зв'язку з клієнтами. Вона націлена на надання максимальної самостійності будь-якому ювелірному магазину, на підвищення гнучкості змісту сайту без зайвих складнощів, скорочення витрат бізнесу на підтримку програмного забезпечення, зробити такі задачі, як маркетинг, веб-дизайн та аналітика в одне ціле, готове до використання рішення.

Актуальність сфери розробки проєкту досі залишається однією з найбільш високих, зокрема через те, що світ все швидше переходить у цифрову еру, де більшість людей віддає перевагу здійсненням будь-яких справ в інтернеті.

Об'єктом дослідження є спрощення процесу керування контентом всередині веб-магазину ювелірного бізнесу, збільшення гнучкості у доступних модифікаціях, усунення необхідності побудови великої команди для роботи над різними частинами бізнесу.

В процесі виконання магістерської роботи було створено веб-систему для управління діяльністю ювелірного магазину. Здійснено докладний аналіз предметної області, розроблено користувацьку діаграму, на основі проведених досліджень розроблено архітектурний концепт та обрано технології для його втілення. Результатом роботи є структурна схема, схема системи управління контентом, інтерфейс програмного забезпечення та реалізована функціональна система.

Ключові слова: ювелірний магазин, веб-додаток, веб-магазин, E-Commerce, адміністрування, керування контентом, гнучкість, скорочення команди, Next.js, Sanity.io, менший бізнес, дешевий вихід на ринок.

ABSTRACT

Mysak O.K. Web application for running a jewelry business. "KPI named after Igor Sikorsky", Kyiv, 2024

The dissertation consists of 112 pages, 45 figures, 52 tables, 19 literary sources and 9 appendices.

The goal of this master's thesis is to develop an information system that allows the user to manage the entire content of the application, which ensures extensibility. The application will increase the efficiency and ease of configuration, data collection and customer communication. It is aimed at providing maximum independence to any jewelry store, increasing the flexibility of site content without unnecessary complications, reducing business costs for software support, and making such tasks as marketing, web design, and analytics into a single, ready-to-use solution.

The relevance of the field of project development still remains one of the highest, in particular due to the fact that the world is increasingly moving into the digital era, where most people prefer to do any business on the Internet.

The object of the study is to simplify the content management process inside the jewelry business web store, increase flexibility in available modifications, eliminate the need to build a large team to work on different parts of the business.

In the process of completing the master's work, a web system was created to manage the activities of a jewelry store. A detailed analysis of the subject area was carried out, a user diagram was developed, an architectural concept was developed based on the research and technologies were chosen for its implementation. The result of the work is a structural diagram, a diagram of the content management system, a software interface and an implemented functional system.

Keywords: jewelry store, web application, web store, E-Commerce, administration, content management, flexibility, downsizing, Next.js, Sanity.io, smaller business, low-cost entry.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface/програмований інтерфейс програми

BaaS – Backend-as-a-Service/бекенд-як-сервіс, платформа для виконання серверного коду, зазвичай надає зручний для використання SDK

CDN – Content Delivery Network/Мережа доправлення

CLI – Command Line Interface/Інтерфейс командного рядка

CMS – Content Management System або система управління змістом

CRUD – Create Read Update Delete/створення, читання, оновлення, видалення. Класичний набір операцій, що можна застосувати до даних.

CSS – Cascading Style Sheets/каскадні таблиці стилів

CRM – Customer Relationship Management або управління відносинами з клієнтами

CSR – Client-side Rendering/Візуалізація на стороні користувача

DB/БД – Database/база даних

DOM – Document Object Model/об’єктна модель документу

ER – Entity Relationship diagram/діаграма сутностей та їх зв’язків

FAQ – Frequently Asked Questions/найбільш поширені запитання

GROQ – Graph-Relational Object Queries/мова запитів до контенту в Sanity

HTML – Hyper Text Markup Language/мова гіпертекстової розмітки

JSON – JavaScript Object Notation/формат обміну даними характерний для вебу

MPA – Multi Page Applications/багатосторінкові застосунки

OS – операційна система

SEO – Search engine optimization/Пошукова оптимізація

SSG – Static Site Generation/Генерація статичних сторінок сайту

SQL – Structured Query Language/мова запитів до більшості баз даних

SPA – Single Page Applications/односторінкові застосунки

SSR – Server-side Rendering/Візуалізація на стороні серверу

UML – Unified Modeling Language або уніфікована мова моделювання

UI – User Interface/інтерфейс користувача

UX – User Experience/користувацький досвід, зручність у використанні

ВСТУП

В сучасному світі все стрімко переходить в цифровий формат і люди все більше схильні вирішувати будь-які свої проблеми в мережі Інтернет. У часи таких тенденцій важливість різноманітних платформ електронної комерції складно переоцінити. Ці платформи надають власникам бізнесів можливість швидко спробувати зайняти місце в онлайн-просторі у обраній сфері та налаштувати свій веб-ресурс так, щоб він містив актуальні дані та приваблював користувачів. Майже кожного дня з’являються нові ідеї і компанії в області онлайн-продаж, де ювелірний бізнес не є виключенням.

Щоб залишатися серед лідерів та займати високі позиції на ринку, будь-який бізнес повинен мати веб-сайт для взаємодії з користувачами та ведення продажів. Найбільш важливі фактори в таких веб-системах є гнучкість, налаштованість для пошукових систем, легкий у використанні інтерфейс, націлений на найшвидшу конвертацію клієнтів, а також високий рівень контролю над даними.

Тож завданням цієї роботи є розробка такої веб-система, яка надала б всі зазначені переваги для бізнесів будь-якого розміру. Гнучка та оптимізована система є ключовим інструментом для заохочення потенційних клієнтів та утримання даних під контролем, саме в такому форматі, як того прагне бізнес. Звісно, можливо досягти такої гнучкості, маючи велику команду розробників, які зроблять все, як попросить власник. Проте, не у всіх є ресурси, щоб наймати цілу команду.

Адміністратор такого сайту зможе легко створювати нові сторінки, контролювати їх зміст, оновлювати список товарів, підтримувати зв’язок з клієнтами, створювати поштові повідомлення, контролювати замовлення та дохід. Не менш важливим також є те, що такий адміністратор зможе аналізувати аналітичні дані, зібрані з дій користувачів, переглядати їх профілі, їх уподобання, та належним чином приймати рішення для подальшого розвитку бізнесу.

Користувач в свою чергу матиме змогу швидко та інтуїтивно переглядати товари, мати доступ до оптимізованого списку товарів, підібраних саме для нього, дивитися відгуки та обирати різноманітні характеристики товарів, які підійдуть саме йому, зможе створювати лист бажань, додавати товари до корзини і виконувати

моментальну оплату, сигналізуючи магазину, що йому необхідно організувати доставку. Йому також стануть доступні гнучкі фільтри, які адміністратор сам позначить, сортування, розумний пошук, можливість підписатися на розсилку для отримання свіжої інформації про нові товари. Користувач також отримає доступ до будь-яких сторінок, які створить адміністратор, наприклад «Про нас», «Умови використання» тощо.

Загалом система буде виконувати наступні функції:

- надасть можливість створення списку товарів, колекцій, фільтрів їх зв’язків, а також швидкий доступ до них для користувачей;
- швидке та наочне отримання аналітичних даних для проведення бізнес-аналізу;
- оплата онлайн та контроль замовлень від клієнтів;
- розсилка оновлень;
- створення нових спеціальних сторінок, відлагоджених за побажаннями адміністратора, без написання коду.

Дане рішення націлене на надання послуг магазинам ювелірних виробів будь-якого розміру. Воно допоможе оптимізувати необхідну кількість робочої сили, спростить ведення стосунків з клієнтами, дозволить створювати унікальні сторінки з обраним змістом для максимальної гнучкості.

Перед проєктом стоїть ціль вирішення наступних бізнес-задач:

- зменшення необхідного персоналу для ведення веб-порталу;
- надання наочної та детальної аналітики для розвитку бізнесу;
- створення та контроль замовлень, оплата;
- легкий та інтуїтивній процес конвертації клієнтів, спрощений доступ до товарів;
- адміністраційна панель для виконання дій, націлених на аналітичний аналіз, або приваблення клієнтів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ІСНУЮЧИХ РІШЕНЬ

В цифровому світі, де все розвивається з кожним днем все більш стрімко, кожний бізнес, який хоче мати більше клієнтів, має усвідомлювати, що статистично понад сорок відсотків людей віддають перевагу покупкам онлайн. А також, понад сімдесят відсотків з них використовують свій мобільний гаджет для отримання доступу до веб-ресурсу. Тому, попри те, що мати веб-сайт важливо, він також має бути швидкий та мати відповідно гарний вигляд на мобільних телефонах, бо вони, зрозуміло, менше за розміром та мають гірше з’єднання з Інтернетом.

Окрім цього, серед важливих факторів хотілось би відмітити інтуїтивність інтерфейсу для користувачів, щоб їм було якомога простіше додати товар до кошика, натиснути на корзину, ввести платіжні дані та натиснути на кнопку оплати. Саме цей шлях користувача має бути дуже простим та насиченим товарами, щоб отримати максимальний потенційний прибуток. Не менш важливим для тієї ж самої цілі є можливість створення контенту на сторінках, налаштування банерів, щоб першою чергою донести до користувача «гарячі» товари.

Тож, у першу чергу, у подальшому порівнянні буде розглянуто ці перелічені фактори.

1.1 Об’єкт та предмет дослідження

Насправді, більшість таких великих ювелірних веб-магазинів дуже схожі за собою. Якщо бренд великий, то він скоріш за все матиме купу фільтрів, колекцій, банерів, інформаційних сторінок та інших візуальних компонентів, які налаштовує команда розробників. Проте, це робочий час і гроші за працю. У першу чергу під час порівняння більшу частину уваги було приділено рішенням, які дозволяють модифікувати зміст найбільш гнучким чином, мають привабливий інтерфейс та швидкий час завантаження.

За змістом, такі магазини та сервіси можна поділити на дві частини:

- один або декілька продуктів для продажу та моментальної оплати;

- мережевий магазин з багатьма виробами, колекціями, сторінками, тощо.

До першого типу належать такі рішення як Shopify[1], Wix[2] та ін. Для них характерно мати один або декілька товарів, на які націлений весь веб-сайт. Користувач заходить і одразу розуміє, хоче він цю річ, або ні. Швидко та без проблем проводить оплату та залишається задоволеним.

Другі – це мастодонти сфери. Такі компанії, як Pandora[3], Bulgari, Cartier та інші, їх веб-сайти мають різноманітні колекції, сторінки та індивідуальні ювелірні вироби. Над такими веб-сайтами працюють команди розробників та підтримки для оновлення контенту а також створення унікальних сторінок.

Тож метою дослідження буде розкрити недоліки обидвох та запропонувати можливі рішення виявлених проблем. Зазначені вище платформи примушують робити вибір. Після виконання проєкту, що є метою розробки робити його буде не потрібно.

1.2 Огляд існуючих рішень

За результатом аналізу існуючих рішень можна буде більш чітко визначити вимоги до майбутньої роботи, врахувавши всі фактори та весь функціонал, який стане вирішальним для покращення вже відомих UX-патернів. Під час пошуку подібних продуктів, пропонується до розгляду декілька систем (або продуктів) з раніше описаних категорій.

1.2.1 Shopify

Shopify – це e-commerce платформа для створення веб-сайтів переважно орієнтованих на продукт. Платформа має багато можливостей і додаткових плагінів, що дозволяє модифікувати контент найбільш різноманітними способами. Це – ,безумовно, один з найбільших конкурентів для продукту, який будується в рамках цієї дисертації. Проблема такого рішення полягає в тому, що користувач може

створювати весь контент лише в рамках продукту. Унікальні сторінки та шаблони не доступні.

На лівій частині екрану можна побачити редактор контенту (рис. 1.1). Він досить багатий на вибір та дозволяє швидко зробити сторінку мрії для свого магазину.

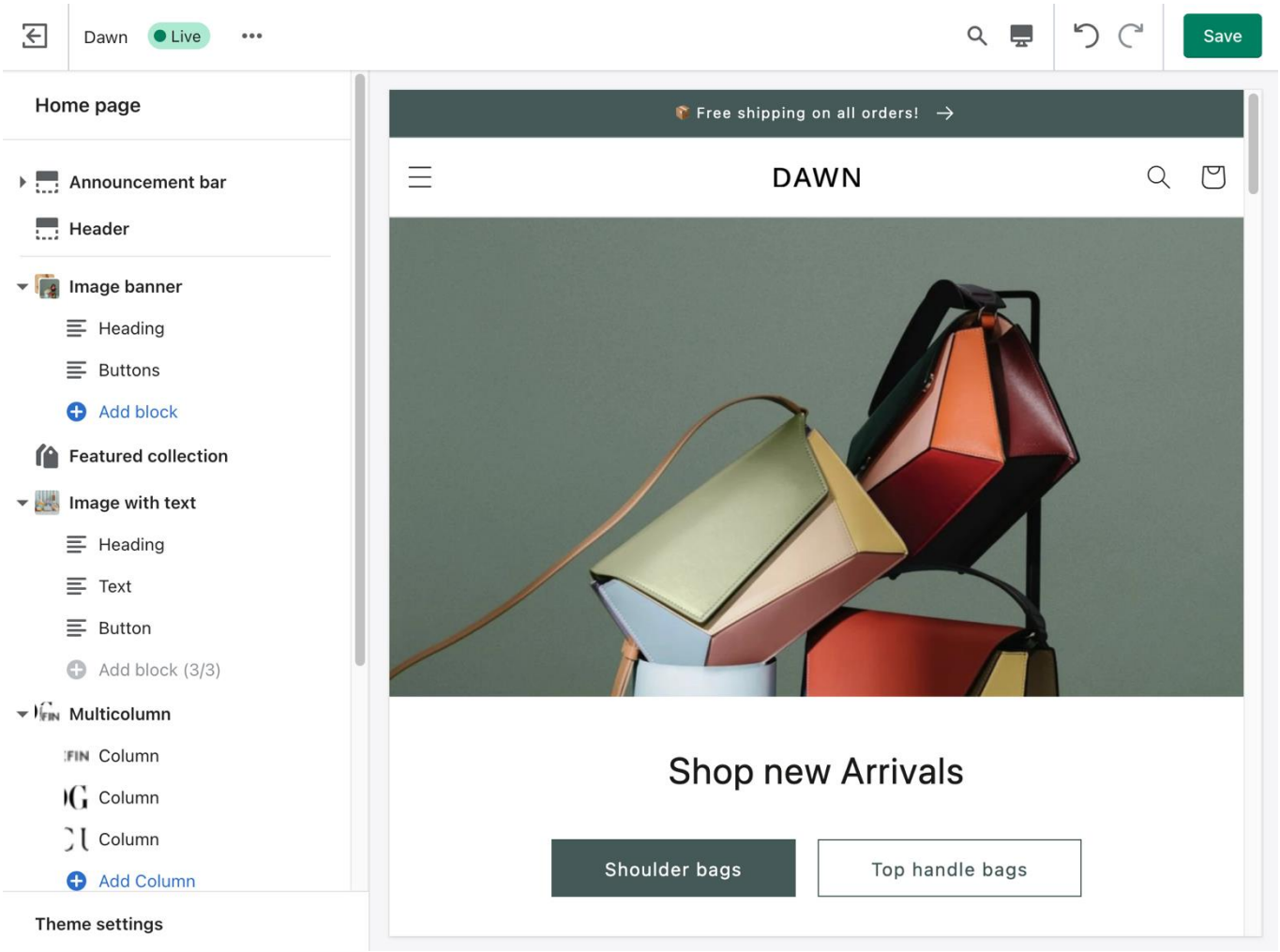

Рисунок 1.1 – Редактор змісту Shopify

З доступних функцій:

- редактор сторінок веб-сайту;
- аналітика даних;
- редактор доступний без коду;
- підтримка.

Даний сайт має сучасний та стильний дизайн, вичерпний функціонал і гарні стандартні шаблони. Проблема полягає в лімітованості шаблонів саме сторінок, а також досить високих цінових планах, коли діло доходить до мінімального індивідуального налаштування. Наприклад, тема може коштувати понад триста

доларів для одного сайту (рис. 1.2). Також, не вдасться зробити зміни без розробника, який знайомий з платформою, якщо такі зміни не передбачені Spotify.

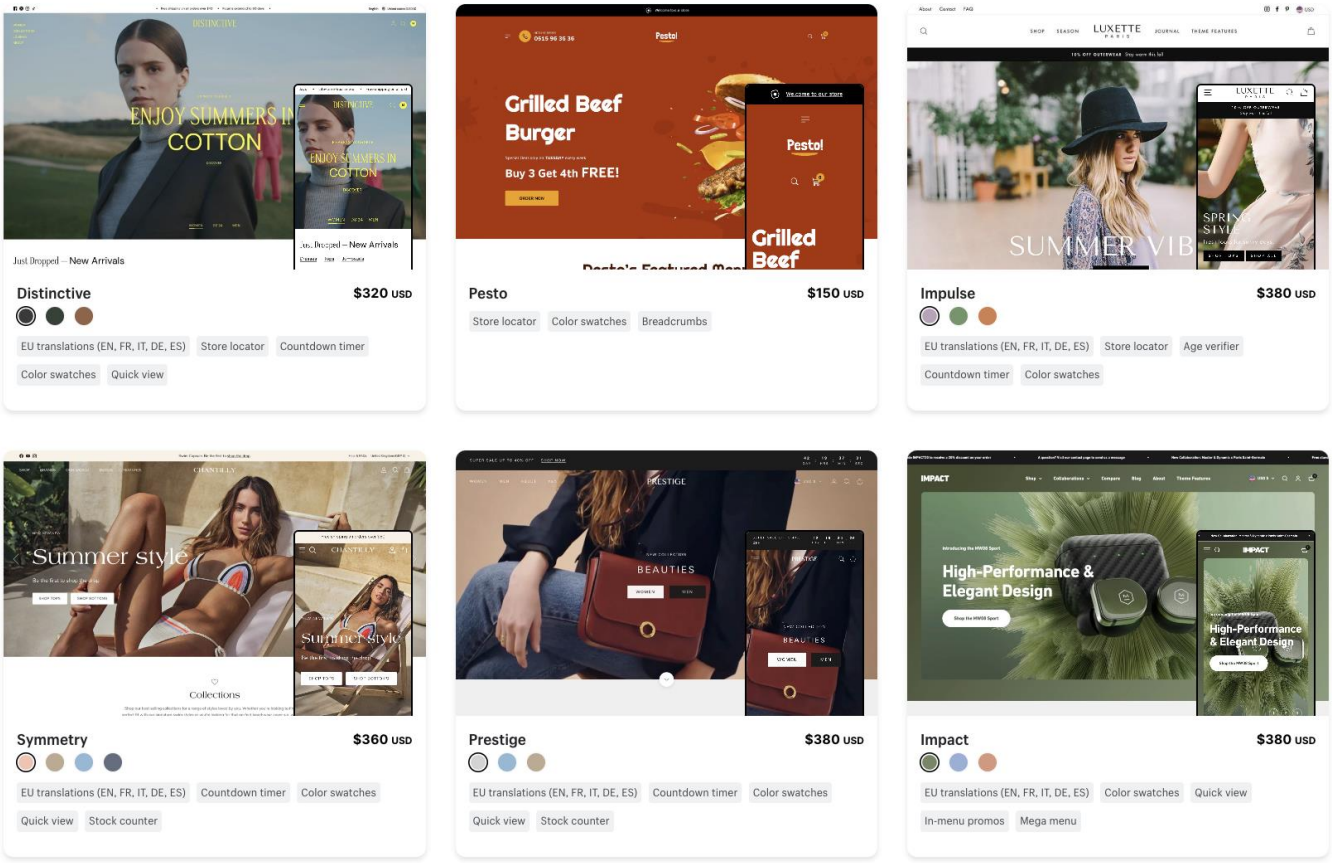

Рисунок 1.2 – Ціни на теми Shopify [4]

Загалом, одне з найкращих рішень для досягнення цілі, яку ставить перед собою дана магістерська дисертація. Однак, підтримка такого сайту для великих магазинів буде коштувати набагато більше, ніж тоді, коли магазин знаходиться на старті.

Також існує певна проблема так званого «lock-in'у», що означає, що коли проєкт почне розвиватися, змінити технологію буде дуже важко, і треба буде до кінця його існування залишатися з Shopify. А також платити кожен місяць.

Підсумовуючи, можна виділити наступні плюси даного програмного комплексу:

- швидкий та простий у використанні;
- підтримка і допомога;
- адаптованість для мобільних девайсів;
- оптимізованість сайту для пошуку.

Серед недоліків можна виділити:

- недостатнє налаштування, якщо бренд буде рости;
- відсутність спілкування з клієнтами;
- неможливість створювати свої власні елементи для відображення на сторінці і розширяти можливості додатку.

1.2.2 WooCommerce

WooCommerce – платформа для побудови веб-сайтів для продажу товарів на базі WordPress. Це, зокрема, і є однією з найбільших проблем цієї системи: застарілість технології, на якій вона побудована. Ця платформа додає складності в розробку, не має досить професійної бази розробників, послугами яких можна скористатися і отримати підтримку і той результат, якого прагне власник бізнесу. Розширюваність дуже проблематична через деталі імплементації, в які дуже важко внести будь-які зміни.

WordPress також приносить з собою деякі проблеми безпеки [5]: такі веб-сайти можуть зіткнутися з більшими ризиками безпеки, якщо їх не обслуговувати та оновлювати належним чином. Це може викликати серйозне занепокоєння для ювелірних компаній, оскільки порушення безпеки можуть завдати шкоди репутації та призвести до фінансових втрат.

Застарілість інтерфейсу і складність у розумінні та налаштуванні через відсутність логічних стандартних налаштувань дає про себе знати вже з перших днів розробки сайту (рис. 1.3).

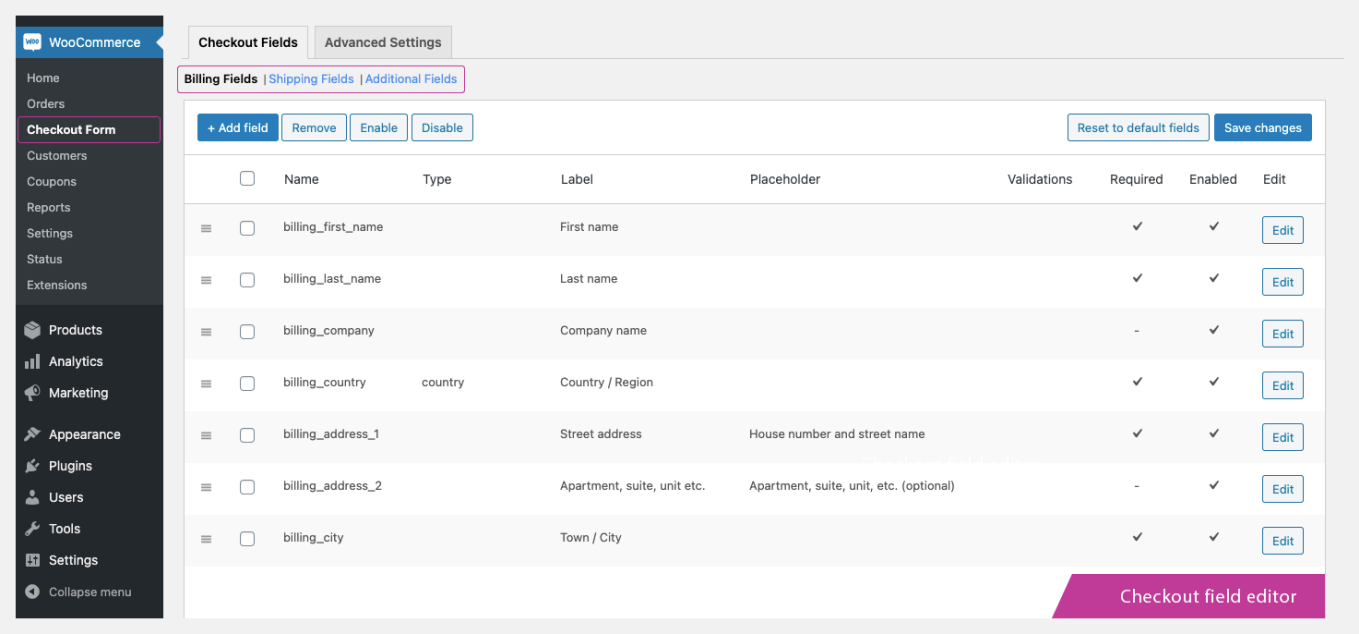

Рисунок 1.3 – Панель керування WooCommerce [6]

Для роботи сайту WordPress із WooCommerce може знадобитися вищий рівень технічної експертизи, особливо для ювелірних компаній, що мають обмежені технічні ресурси, це може бути значною перешкодою для досягнення швидкого розгортання та підтримки продукту.

Управління веб-сайтом WordPress із численними плагінами, темами та оновленнями найбільш вірогідно викликатиме складності. Для власників ювелірних магазинів простота й ефективність є вирішальними, і ці складності можуть перешкодити їхній здатності підтримувати простий і ефективний онлайн-магазин.

З переваг варто зазначити:

- недостатнє налаштування, якщо бренд буде рости;
- легко інтегрується з WordPress, широко використовуваною та універсальною системою керування вмістом. Ця інтеграція пропонує гнучку та звичну платформу для демонстрації своїх продуктів;
- пропонує велику кількість тем і плагінів, що дає власникам ювелірних магазинів свободу налаштовувати та масштабувати свої онлайн-магазини в міру розвитку свого бізнесу.

З недоліків:

- застарілість технологій;
- обмежена розширюваність без технічних знань;

- неможливість розширюватися через лімітованість WordPress та специфічної екосистеми плагінів.

1.2.3 Pandora

Pandora – патентований продукт, не є платформою для розробки, а є веб-ресурсом однієї з широко відомих ювелірних компаній.

Основним недоліком даної веб-системи є неможливість модифікації змісту сторінок без участі команди розробників, адже єдине, що там може додавати не розробник – це товари. Ще однією проблемою є швидкість завантаження. Браузер отримує пакет розміром біля 10 мегабайт на головній сторінці, що каже про погану оптимізацію фінального пакету програмного забезпечення, неякісною роботою з оптимізацією зображень та відсутності SSR.

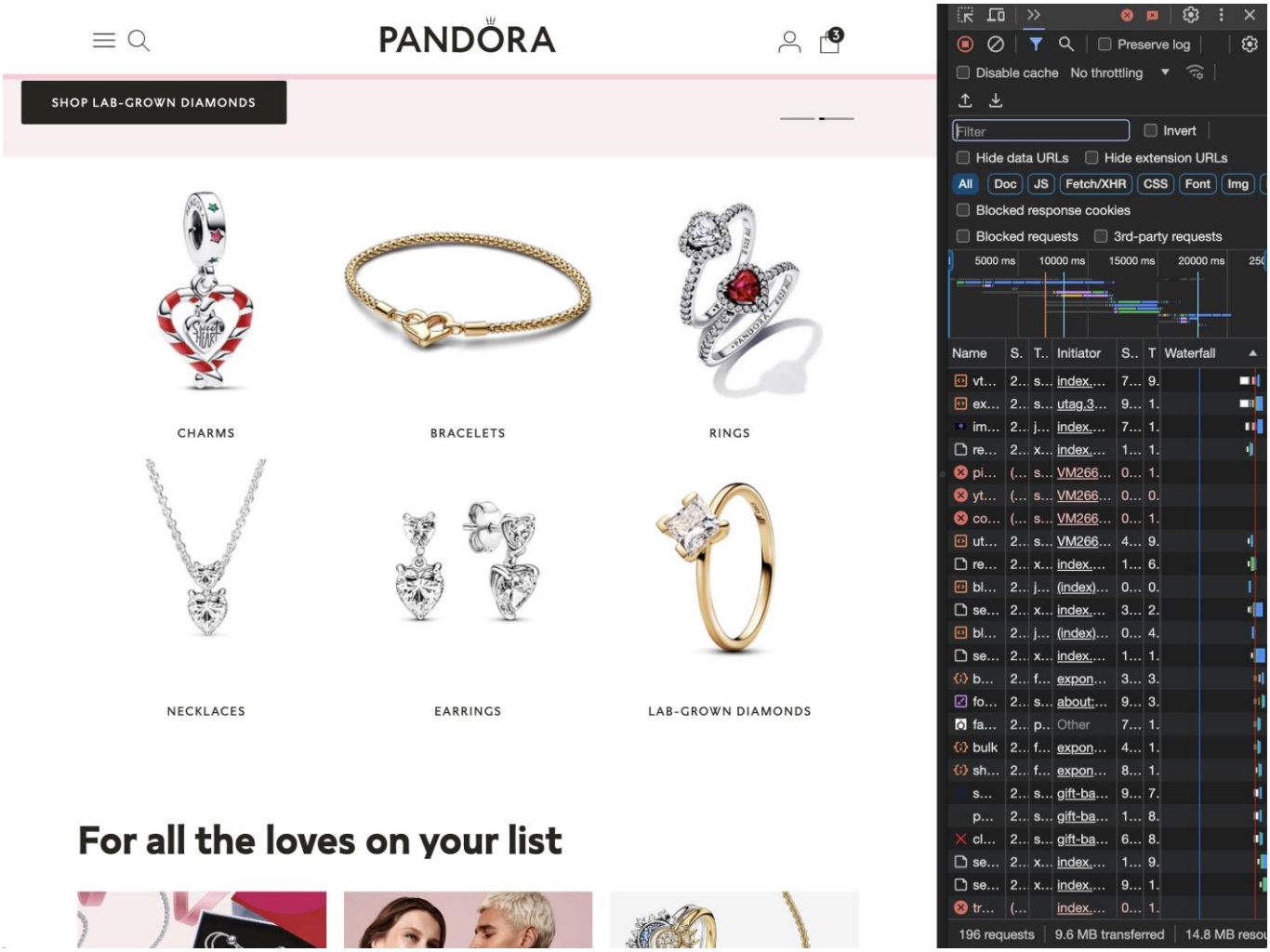

Рисунок 1.4 – Головна сторінка Pandora

Для завантаження повного пакету і відображення змісту мені знадобилося більше 20 секунд, що є неприйнятним для продукту, який хоче конвертувати найбільшу кількість клієнтів.

З переваг ресурсу зокремо хотілося б зазначити різноманітність фільтрів та якість пошукового поля. Ці функції було запозичено у фінальному продукті, бо вони дійсно зроблені на гарний лад і налаштовані на високу якість користувацького досвіду. А також:

- швидкість конвертації через меню кошика;
- лаконічний дизайн, що приваблює користувачів;

Із основних недоліків:

- відсутність можливості підтримувати веб-систему автономно без участі розробників;
- надто велика кількість посилань, важко знайти те, що шукаєш без використання пошуку.

Вданому розділі було проаналізовано об’єкт та предмет дослідження, визначено актуальність розробки, та проведено розширений аналіз альтернативних рішень для ведення ювелірного бізнесу. Результат цього аналізу дозволив визначити сильні та слабкі сторони інших платформ та брендів, якими можна скористатися при визначенні вимог до результуючого продукту. Було проаналізовано функціонал даних систем та визначено основні складові частини, які стали основою для проведення даного аналізу. Після цього, здійснено зважження всіх переваг та недоліків кожної з альтернатив.

Результатами виконання аналізу є чітко визначений стан ринку подібних рішень, а також чітко зазначені об’єкт та предмет дослідження. Результат аналізу показує, що кожне з застосувань не в повній мірі, або зовсім не виконує потребу в гнучкості і автономності. Розроблювана система націлена на покриття цих недоліків. Тепер сформуємо вимоги до розроблюваного продукту.

2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

В цьому розділі пропонується визначити вимоги до результуючого продукту. Створення списку таких вимог є суттєвою складовою успіху в розробці будь-якої системи. В попередньому розділі було охарактеризовано альтернативні рішення до продукту магістерської дисертації, завдяки чому тепер можна чітко розуміти чого не вистачало існуючим платформам а також запозичити деякі з їх істотних функцій для максимізації якості кінцевого додатку.

Спочатку необхідно визначити сторони, які прийматимуть участь в системі, тобто гравців, для яких необхідно обумовити комфортний досвід використання. Додаток матиме лише три такі сторони:

- користувач – потенційний клієнт магазину;
- володар аккаунту – авторизований користувач;
- власник магазину або адміністратор.

Вимоги можуть бути двох типів: функціональні та нефункціональні[7]. Тож у наступних двох підрозділах визначимо їх.

2.1 Функціональні вимоги

Функціональні вимоги формулюють необхідну поведінку системи. Вимоги такого типу визначають безпосередньо програмні функції системи, її наповненість корисними шляхами для користувачів, щоб задовольнити їх очікування та потреби. Тобто, це, як правило ,запрограмовані можливості системи, до яких користувач має прямий доступ і може ними скористатися в разі необхідності.

Система, яка є кінцевим продуктом даної роботи повинна виконувати такі функціональні потреба користувачів

- авторизація для доступу в особистий кабінет і написання відгуків. З точки зору бізнесу, будуть деякі функції, що лімітовані для авторизованих користувачів для покращення якісного збору даних;

- можливість адміністраторів створювати унікальні сторінки для користувацького інтерфейсу;
- розумні стандартні налаштування та можливість їх редагування для зменшення необхідної роботи, щоб почати працювати;
- створення відгуків до продукції;
- розумні секції, що шукатимуть продукти особисто для користувача і заохочуватимуть більше конвертацій;
- створення продуктів та колекцій;
- підтримка контакту з користувачами;
- створення рекламних поштових листувань;
- створення шаблонів для листування;
- надання користувачам можливості ефективно та поглиблено фільтрувати, сортувати товари для швидкого доступу до набору товарів, що йому цікавий;
- перегляданння історії замовлень для користувача (якщо він зареєстрований) та для адміністратора;
- можливість створювати замовлення;
- можливість здійснювати оплату;
- можливість створення індивідуальної сторінки для колекції;
- ведення аналітики для розуміння тенденцій та улюблених товарів користувачів, шляхів для покращення бізнесу;
- перегляд зібраної аналітики;
- розумний пошук серед всього змісту веб-сайту;
- доступ до навігації у будь-якому місці в додатку;
- додання та видалення продукції з кошику;
- додання та видалення продукції з списку бажаного;
- панель перегляду інформації про зареєстрованих користувачів;
- надання адміністраторам можливості створення банерів та слайдерів довільного формату;
- надання адміністраторам можливості наповнення сторінки частих питань новими вхідними даними;

- створення зручної та багатофункціональної панелі для маніпуляції даними для адміністраторів;
- створення довільних фільтрів для продукції;
- система повинна ідеально виконувати користувацькі історії подані в додатку Б. Далі зазначимо перелік нефункціональних вимог.

2.2 Нефункціональні вимоги

Нефункціональні вимоги характеризують систему без чіткої прив’язки до функцій як таких. Тобто, до системи, що розроблюється, застосовують вимоги, які загалом пояснюють сенс її існування, а також її можливості та обмеження. Іншими словами, вони зазначають яким саме продукт має бути у результаті побудови.

Беручи до уваги попередньо визначені переваги та недоліки розглянутих систем, було побудовано наступний перелік:

- система повинна забезпечувати легкий доступ до модифікації сторінок та даних без потреби писати щось, окрім HTML/CSS та тексту. Всі інші зв’язки можуть бути побудовані на рівні відносин між об’єктами та реалізовані внутрішньою логікою;
- система повинна забезпечувати інтуїтивну навігацію, щоб користувач завжди мав доступ до потенційно корисних для нього сторінок;
- система має бути завжди оптимізованою для мобільних пристроїв;
- система має бути завжди оптимізованою для пошукових систем;
- система повинна збирати аналітику для подальшого аналізу та розвитку бренду;
- система має зберігати продуктивність виконання та бути максимально швидкодоступною при умові зберігання понад 2000 товарів;
- система має шляхом листування заохочувати конвертацію потенційних клієнтів;
- система має шляхом додаткових розумних секцій заохочувати конвертацію потенційних клієнтів;
- система має зробити процес конвертації максимально інтуїтивним та швидким;

- система має надавати прозору інформацію щодо замовлень для адміністраторів для найшвидшої обробки;
- система не повинна використовувати нетривіальні діалекти та приховувати системний код;
- система повинна використовувати сучасні технології для потенційної модифікації логіки і розширення з часом;
- система не повинна прив’язувати користувача до єдиного способу її використання, в будь-який момент програмний код може бути модифікований для адаптації до специфічних вимог бізнесу;
- система повинна забезпечувати доступ до ресурсу в найкоротший час шляхом використання найсучасніших технік оптимізації контенту.

3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Формування вимог для веб-додатку поклало базу для розробки UML-діаграми, яка описуватиме різні сценарії експлуатації системи в деталях та пояснить виконання визначених вимог.

Кожний варіант використання системи є систематичною послідовністю дій, які система спроможна виконати. Такий аналіз забезпечує концептуальне пояснення різних шляхів, якими може пройти той чи інший користувач системи за допомогою графічної репрезентації, яку спроможний зрозуміти будь-хто. Розробка такої блок-схеми допомагає розробнику проєкту в деталях описати всі можливі шляхи, які будуть доступні користувачу системи та наочно показати їх результат.

У попередніх розділах було визначено вимоги та основних акторів, що прийматимуть участь у тих чи інших шляхах взаємодії з продуктом. А саме такі типи ролей:

- «Користувач» – потенційний клієнт;
- «Володар аккаунту» – користувач, що пройшов авторизацію;
- «Адміністратор» – авторизований адміністратор бренду.

Діаграма варіантів використання системи наведена в додатку А. У підрозділах 3.1-3.3 проведено ретельний аналіз прецедентів для кожного актора.

3.1 Опис спільних прецедентів

Система, що розроблюється, не так сильно залежить від авторизації користувача, основний функціонал все одно є доступним навіть без проходження процедури авторизації. Проте лише авторизовані користувачі можуть оставляти відгуки. Саме через цю особливість, користувач у більшості випадків має сприйматися як будь-який відвідувач веб-сервісу.

Таблиця 3.1 – Прецедент перегляду товарів

Назва	Переглянути список товарів
Актори	Користувач
Опис	Користувач переглядає список товарів та обирає бажану покупку
Передумова	Наявність заповнених товарів адміністратором
Успішний сценарій	1. Відвідати бажану категорію товарів з меню навігації 2. Система виконує запит до системи керування змістом 3. Система відмальовує отримані дані
Результат	Користувач відкрив сторінку з товарами та переглядає їх
Виключення	Система не знайшла жодного товару 1. Система відображає текстовий напис повідомляючий користувачеві про відсутність даних

Таблиця 3.2 – Прецедент фільтрації товарів

Назва	Переглянути відфільтровані товари
Актори	Користувач
Опис	Користувач фільтрує список товарів
Передумова	Наявність заповнених товарів адміністратором
Успішний сценарій	1. Відвідати бажану категорію товарів з меню навігації 2. Обрати бажані фільтри, щоб звузити коло товарів 3. Система виконує запит з фільтрами 4. Система відмальовує отримані дані

Продовження таблиці 3.2

Результат	Користувач відкрив сторінку з товарами, застосував бажані фільтри та переглядає товари
Виключення	Система не знайшла жодного товару 1. Система відображає текстовий напис повідомляючий користувачеві про відсутність даних

Таблиця 3.3 – Прецедент перегляду інформації про товар

Назва	Переглянути інформацію про товар
Актори	Користувач
Опис	Користувач переглядає інформацію про товар
Передумова	Товар існує в системі
Успішний сценарій	1. Відвідувач повторює сценарій «Переглянути товари» 2. Відвідувач обирає бажаний товар 3. Система відображає сторінку, що містить інформацію про товар
Результат	Відвідувач відкрив сторінку з інформацією про товар
Виключення	Якщо товар зазначений в запиті не існує, користувач буде направлений на сторінку з відображенням тексту помилки 404

Таблиця 3.4 – Прецедент перегляду колекції

Назва	Переглянути інформацію про колекцію
Актори	Користувач
Опис	Користувач переглядає інформацію про колекцію

Продовження таблиці 3.4

Передумова	Колекція існує в системі
Успішний сценарій	1. Відвідувач відкриває сторінку колекції через меню або через будь-яке інше посилання 3. Система відображає сторінку з інформацією про колекцію
Результат	Відвідувач відкрив сторінку з інформацією про колекцію
Виключення	Якщо колекція зазначена в запиті не існує, користувач буде направлений на сторінку з відображенням тексту помилки 404

Таблиця 3.5 – Прецедент авторизації

Назва	Авторизація
Актори	Користувач
Опис	Користувач входить в систему
Передумова	Конфігурація для сервісу авторизації існує
Успішний сценарій	1. Користувач натискає на кнопку входу 2. Користувач обирає бажаний спосіб авторизації 3. Зовнішній провайдер виконує авторизацію та перенаправляє користувача до додатку
Результат	Відвідувач увійшов в систему
Виключення	Не існує, бо без конфігурації авторизації система не буде запущена

Таблиця 3.6 – Прецедент підписки на листування

Назва	Підписка на листування
Актори	Користувач
Опис	Користувач підписується на листування
Передумова	Конфігурація для сервісу листування існує
Успішний	1. Користувач натискає на кнопку підписки 2. Зовнішній провайдер авторизації додає прапорець підписки до мета-даних користувача
Результат	Відвідувач увійшов в систему
Виключення	Ключа для сервісу листування не існує, або він не налаштований належним чином 1. Відобразити помилку для користувача

Таблиця 3.7 – Прецедент оформлення замовлення

Назва	Оформлення замовлення
Актори	Користувач
Опис	Користувач оформлює замовлення
Передумова	Конфігурація для сервісу оплати існує

Продовження таблиці 3.7

Успішний	1. Користувач додає товари до кошику 2. Користувач заповнює платіжні дані і натискає на кнопку оплати 3. Провайдер оплати здійснює операцію оплати 4. Система створює запис типу «Замовлення» 5. Користувач перенаправлений на сторінку успішного замовлення
Результат	Відвідувач увійшов в систему
Виключення	Оплату було відхилено 1. Користувачу показано відповідне повідомлення

Таблиця 3.8 – Прецедент додання до листу бажаного

Назва	Додання до листу бажаного
Актори	Користувач
Опис	Користувач додає товар до листу бажаного
Передумова	Відсутня
Успішний	1. Користувач виконує сценарій «Переглянути товар» 2. Користувач натискає на кнопку «Додати в список бажаного»
Результат	Список бажаного оновлюється і користувач бачить нову кількість товарів у списку
Виключення	Не існує

Таблиця 3.9 – Прецедент додання до кошику

Назва	Додання до кошику
Актори	Користувач

Продовження таблиці 3.9

Опис	Користувач додає товар до кошику
Передумова	Відсутня
Успішний	1. Користувач виконує сценарій «Переглянути товар» 2. Користувач обирає необхідні параметри товару 3. Користувач натискає на кнопку «Додати в кошик»
Результат	Список бажаного оновлюється і користувач бачить нову кількість товарів у кошику
Виключення	Не існує через присутність валідації на користувацькій частині

3.2 Опис прецедентів для адміністратора

В даному підрозділі розглянуто основні варіанти використання системи для адміністратора.

Таблиця 3.10 – Прецедент створення секції

Назва	Створення секції
Актори	Адміністратор
Опис	Адміністратор створює секцію, яку потім зможе використати для побудови унікальних сторінок або додання до сторінки колекції
Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор заходить в адмін-панель 2. Адміністратор створює запис типу «Секція» 3. Адміністратор обирає тип секції та заповнює необхідні поля 4. Адміністратор натискає кнопку «Опублікувати»

Продовження таблиці 3.10

Результат	Створено нову секцію
Виключення	При виникненні помилки валідації, адміністратору буде повідомлено що саме не так і секцію не буде створено

Таблиця 3.11 – Прецедент створення сторінки

Назва	Створення сторінки
Актори	Адміністратор
Опис	Адміністратор створює сторінку, яка буде моментально доступна користувачам системи
Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор заходить в адмін-панель 2. Адміністратор створює запис типу «Сторінка» 3. Адміністратор заповнює необхідні поля, зокрема секції 4. Адміністратор натискає кнопку «Опублікувати»
Результат	Створено нову сторінку
Виключення	При виникненні помилки валідації, адміністратору буде повідомлено що саме не так і сторінку не буде створено

Таблиця 3.12 – Прецедент створення фільтру

Назва	Створення фільтру (тегу)
Актори	Адміністратор
Опис	Адміністратор створює фільтр

Продовження таблиці 3.12

Передумова	Адміністратор авторизований в системі
Успішний сценарій	1. Адміністратор заходить в адмін-панель 2. Адміністратор створює запис типу «Тег» 3. Адміністратор заповнює необхідні поля 4. Адміністратор натискає кнопку «Опублікувати»
Результат	Створено новий продукт
Виключення	При виникненні помилки валідації, адміністратору буде повідомлено що саме не так і фільтр не буде створено

3.3 Опис прецедентів для володаря аккаунта

У авторизованого користувача (за стандартними налаштуваннями) є лише один прецедент, який не доступний користувачеві:

Таблиця 3.13 – Прецедент створення відгуку

Назва	Створення відгуку
Актори	Володар аккаунту
Опис	Володар аккаунту залишає відгук
Передумова	Володар аккаунту авторизований в системі
Успішний сценарій	1. Володар аккаунту виконує сценарій «Переглянути товар» 2. Володар аккаунту натискає «Залишити відгук 3. Володар аккаунту обирає рейтинг та пише текст відгуку 4. Володар аккаунту натискає «Відправити»
Результат	Створено новий відгук до товару

Продовження таблиці 3.13

Виключення	При виникненні помилки валідації, користувачеві буде повідомлено що саме не так і відгук не буде створено

У цьому розділі роботи основна увага була приділена вивченню альтернативних шляхів користування додатком, на основі чого були виокремлені ключові групи користувачів, з якими працюватиме розроблювана система. Визначено спектр дій, доступних для користувачів, та умови їх застосування. Була складена деталізована UML-діаграма, яка наочно демонструє всі можливості системи і сприяє глибшому розумінню запропонованих їй функцій. Також було розроблено докладний опис типових сценаріїв для користувачів, узгоджено їх з вимогами, описаними у попередньому розділі.

Завершення цього розділу дало змогу чітко окреслити функціональні характеристики системи та уточнити перелік її основних користувачів. Наступний крок передбачає вибір інструментів та технологій для практичного втілення системи.

4 ВИБІР ТА ОБҐРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Вибір технологій є важливим кроком в процесі розробки програмного забезпечення, адже від нього буде залежати оптимальність розробленої системи, здатність до розширення, швидкість та вартість розробки.

Базуючись на розроблених функціональних та нефункціональних вимогах необхідно визначити стек технологій та архітектуру системи. Стеком технологій називають набір засобів, інструментів, мов програмування та фреймворків, що використовуються розробниками для написання веб та мобільних додатків.

Підбір технологій є ключовим етапом у створенні будь-якого цифрового продукту, оскільки він є одним з основних факторів, що впливають на ефективність, масштабованість, швидкість реалізації та витрати на розробку системи.

Беручи за основу заздалегідь визначені функціональні та нефункціональні вимоги, настав час обирати інструменти для реалізації ідеї проекту. Необхідно обрати такий набір технологій (технологічній стек) та побудувати таке архітектурне рішення, щоб всі (або більшість) з поставлених цілей було досягнути найпростіше. Технологічний стек включає в себе різноманітні інструменти, мови програмування та фреймворки, які програмісти використовують для створення додатків в тій чи іншій сфері розробки.

4.1 Тип та архітектура додатку

Для створення цільового продукту було прийнято рішення створити веб-додаток. По-перше тому що в цій області екосистема зростає дуже швидко і дозволяє з мінімальними зусиллями будувати дуже потужні системи. Веб-додатками дуже просто керувати, тому що доступ до них може бути отриманий всього лише за допомогою переходу за посиланням. Не є секретом, що більшість користувачів Інтернету віддає перевагу саме веб-додаткам через те, як просто ними користуватися.

В попередніх розділах вже було прийняте рішення дивитися більше у бік веб-платформ, тому що це є стандартом для будь-якого онлайн-бізнесу. Це – перше, що

власник захоче мати в наші часи, якщо він бажає залишатися на ринку і тримати планку. Окрім простоти доступу можна зазначити ще деякі переваги, про які піде мова далі.

Незалежність від платформи – веб-додаток є доступним з будь-якого пристрою, що є дуже важливим. Потенційний клієнт бізнесу може віддавати перевагу будь-якому зі своїх гаджетів, отже, важливо забезпечити підтримку всіх можливих варіантів.

Технології і їх зрілість – у світі веб-розробки є провідні технології і фреймворки, що дозволяють реалізовувати ідеї у дуже швидкому темпі, завдяки тренду мета-фреймворків, зокрема таких як Next.js[8], SvelteKit[9], Nuxt.js[10], тощо. Ці фреймворки є рішеннями для написання як користувацької частини додатку, так і серверної. Саме цим розроблюваний додаток може скористатися. І далі стане зрозуміло чому саме.

Даний тип додатків також є дуже швидким, якщо розроблений правильно, а також знаходиться в тому самому середовищі, де знаходяться всі зовнішні провайдери, які було б непогано використати в такому проєкті.

Коли мова йде про зовнішні провайдери, мається на увазі пошук провайдерів, які дадуть відповіді наприклад на такі запитання:

- Де зберігати дані?
- Як реалізувати авторизацію?

Звісно, є варіант створити власну базу даних і керуючись нею реалізувати свою власну аутентифікацію, як і озеро контенту. Але у випадку з нашим додатком, це не зовсім те, що продукту треба. Підтримувати власну базу означає створювати інтерфейс для адміністраторів вручну, створювати власний шар валідації даних, або використовувати ORM (Object Relational Mapper), а також хостити її у якомусь хмарному сховищі.

Для того, щоб позбутися всіх цих проблем було прийнято рішення не використовувати сервер, а використати безсерверну (Serverless) архітектуру. Це рішення допоможе збутися непотрібних накладних витрат на хостинг серверу, і є цілком виправданим рішенням, тому що в такому додатку, як розробляється,

розгортати цілий сервер з базою даних не має жодного сенсу. Набагато легше використати готове рішення для зберігання даних, таке як BaaS (Backend-as-a-Service) або CMS (Content Management System). Такі інструменти, як правило, є цілими платформами і дають змогу використовувати централізоване сховище на їх стороні, а також надають змогу використовувати CDN (Content Delivery Network), якщо мають її для забезпечення швидкої доставки контенту.

Тож було прийняте рішення позбавитися непотрібних складнощів за рахунок наступних рішень:

Авторизація за допомогою зовнішнього провайдера. Це допоможе позбутися необхідності будувати її власноруч, коли це є однією з останніх речей, які мають бути в додатку, зокрема через те, що авторизований користувач має лише одну додаткову функцію над звичайним користувачем. Також, зовнішні рішення допоможуть мати легкий доступ до статистики і аналітики, а також зберігати мета-дані за потреби.

Система керування змістом (CMS) для зберігання даних і їх редагування. Централізоване джерело даних з доступним CDN і вбудованим механізмом кешу буде ідеальним варіантом. Нема необхідності мати окремий сервер і розповсюджувати API інтерфейс, тому що його просто не буде. Всі дані можуть бути отримані напряму з CMS.

Для обробки коду на сервері можна скористатися вбудованим функціоналом раніше зазначених мета-фреймворків, чого має бути достатньо для додатку, що розробляється.

Проблема архітектури серверної частини вирішена просто так, без особливих складнощів.

Проте не варто забувати про користувацьку частину. Загалом, якщо не заглиблюватися в термінологію, веб-застосунки бувають:

- MPA – багатосторінкові додатки;
- SPA – односторінкові додатки.

MPA – це старий спосіб роботи з веб-платформою, який виник у часи популярності мови PHP, де при кожному обміні даними з клієнтом, сторінка повністю

перезавантажує зміст заново і відмальовує новий файл (сторінку). , тобто сервер щоразу надсилає нову сторінку.

SPA – підхід, який виник після MPA, тому що SPA – це швидше і більш практичний підхід до роботи з роутингом. Даний підхід передбачає завантаження користувацькою частиною маленького HTML файлу зі слотом для рендерингу іншого змісту, який відбувається за допомогою використання JavaScript.

Схематично різниця між даними видами обміну даних подана на рисунку 4.1.

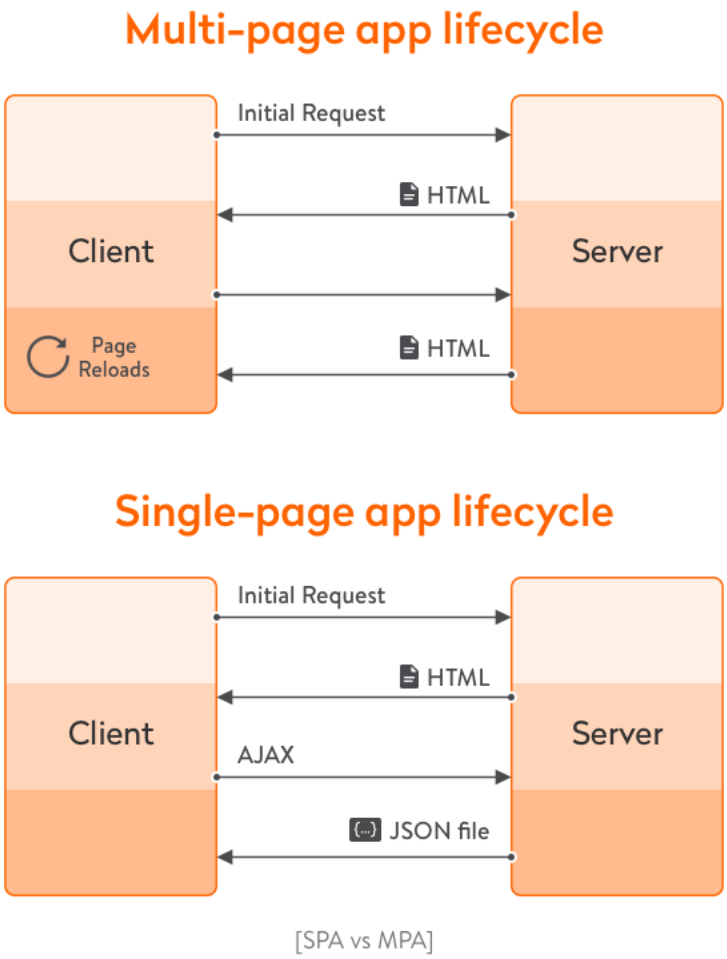

Рисунок 4.1 – Різниця між SPA та MPA[11]

Проте, як виявилося, жоден з цих підходів не є ідеальним. MPA робить занадто багато для того, що має бути зроблено, а SPA ускладнює працю з деякими критично важливими концептами, як наприклад індексація пошуковими системами, або скорочення розміру завантажуваного JavaScript коду. Тому, зараз, мета-фреймворки, такі як зокрема Next.js намагаються поєднувати краще з обох світів використовуючи Progressive Enhancement і Partial Hydration, два складні для пояснення терміни, які

мають на увазі часткову відправку контенту користувачеві для моментального доступу до частини даних, а потім відбувається «гідрація»[12], процес, коли JavaScript код приходить в дію, і всім кнопкам і іншим інтерактивним елементам надаються слухачі подій, що роблять їх інтерактивними. Коли якась частина змінюється такі фреймворки можуть точково оновлювати інтерфейс, використовуючи Partial Hydration і HTML Streaming. Тож, насправді, такий підхід на поточний день є напевне найкращим.

4.2 Платформа та мова програмування

На даних момент існує багато різноманітних мов програмування для різних сфер розробки, проте у веб-розробці все залишається незмінним. JavaScript продовжує домінувати, маючи величезну екосистему бібліотек та TypeScript у себе за спиною. Насправді, тут даже нема про що думати, але все ж таки є сенс провести порівняльний аналіз найпопулярніших мов програмування для веб-платформи, таких як:

- Go;
- Python;
- JavaScript.

Go, часто відома як Golang, є відкритою мовою програмування, розробленою Google у 2007 році. Її головними перевагами є простота, швидкість та ефективність, особливо в мережевому програмуванні та масштабованих системах. Go відома своєю здатністю до паралельного виконання завдань завдяки вбудованим можливостям корутин, званих "горутинами". Це дозволяє легко створювати багатопоточні програми. Крім того, Go має строгу типізацію та автоматичне управління пам'яттю, що забезпечує високу продуктивність та безпеку.

Go стала популярною у веб-розробці для створення надійних та ефективних веб-серверів та мікросервісів. Її компактний синтаксис та вбудована підтримка одночасної обробки допомагають розробникам створювати високопродуктивні веб-

додатки з меншим обсягом коду. Також, завдяки зростаючій підтримці спільноти, є багато бібліотек та фреймворків, спрямованих на веб-розробку.

Python є однією з найпопулярніших мов програмування у світі завдяки своїй простоті та читабельності. Вона ідеально підходить для початківців та має широкий спектр застосувань, від веб-розробки до наукових досліджень. Python підтримує кілька парадигм програмування, включаючи об'єктно-орієнтовану, процедурну та функціональну, що надає гнучкість розробникам. Його велика стандартна бібліотека та велика кількість сторонніх пакетів роблять Python могутнім інструментом для вирішення широкого спектру завдань.

У веб-розробці Python використовується для створення веб-серверів, API та бекендів для веб-додатків. Він популярний через свої потужні веб-фреймворки, такі як Django, FastAPI та Flask, які полегшують створення масштабованих та безпечних веб-додатків. Крім того, Python часто використовується для роботи з базами даних, виконання серверного скриптингу, та інтеграції з іншими мовами та сервісами.

JavaScript є мовою програмування, яка стала стандартом для веб-розробки, використовуючись на клієнтській (браузерній) стороні для створення інтерактивних веб-сторінок. Завдяки своїй універсальності та підтримці всіма основними браузерами, JavaScript відіграє ключову роль у створенні динамічних та залучаючих користувацьких інтерфейсів. Ця мова підтримує об'єктно-орієнтоване, імперативне та функціональне програмування, дозволяючи розробникам використовувати різноманітні підходи для розв'язання проблем. З розвитком Node.js, JavaScript також став популярним для серверного програмування, що розширило його можливості за межі веб-браузерів.

Одним із сучасних розширень JavaScript є TypeScript. TypeScript є надмножиною JavaScript, яка додає строгу типізацію та інші функції, які допомагають управляти великими кодовими базами та забезпечують більшу безпеку розробки. Використання TypeScript допомагає виявляти та виправляти помилки на ранніх етапах розробки, що є важливим для складних або масштабованих проектів. Його популярність зростає завдяки підтримці великих проектів та спільноти, яка активно розвивається.

Вибір JavaScript та TypeScript для веб-розробки є виправданим з огляду на їхню універсальність, широку підтримку та здатність до адаптації до різних потреб веб-проектів. JavaScript є відмінним вибором для створення інтерактивних веб-інтерфейсів, тоді як TypeScript пропонує додаткові переваги у вигляді типізації та масштабованості для більш складних проектів. Через вибір TypeScript в якості мови програмування для розроблюваного додатку можна використовувати одну і ту саму мову програмування для двох частин додатку, що є великим плюсом в рамках DX (Developer Experience).

Також, найпотужніші та найзручніші у використанні фреймворки, всі написані для TypeScript, що робить його найбільш привабливим варіантом. У наступному підрозділі буде обрано фреймворк для використання з цією мовою програмування.

4.3 Вибір фреймворку

Фреймворк – це набір заздалегідь написаних програмних компонентів, що значно пришвидшують та покращують процес розробки.

Для побудови додатку, що розробляється в рамках цієї роботи, було прийнято рішення використати вбудовану самостійно розміщувану систему керування змістом (CMS) Sanity. Цю систему можна цілком використовувати як свого роду базу даних, а також вона має гарні стандартні налаштування, які якісно працюють разом з трендами в світі мета-фреймворків, такими як, наприклад, кешування і CDN (Content Delivery Network), що дозволяють розмістити додаток на Edge (безсерверна платформа) і використовуючи CDN бути впевненим, що дані йдуть в додаток з найближчого доступного джерела.

Тож, у комбінації з цією системою було обрано фреймворк Next.js. Вибір стояв між SvelteKit та Next.js, але все ж таки було прийнято рішення зробити ставку на більш зріле рішення з величезною екосистемою React, яка має готове рішення у вигляді бібліотеки або функціоналу, щоб відповісти на будь-яке запитання, яке може постати під час сучасної розробки.

Next.js є відкритим фреймворком для веб-розробки, побудованим на базі React, що дозволяє створювати статичні та серверно-генеровані веб-сайти та додатки. Запущений у 2016 році, Next.js швидко набув популярності завдяки своїй гнучкості та продуктивності, пропонуючи потужні функції для сучасної веб-розробки.

Одна з ключових особливостей Next.js - це його підтримка Server-Side Rendering (SSR)[13] та Static Site Generation (SSG)[14]. SSR дозволяє веб-сторінкам завантажуватися швидше, оскільки більша частина контенту генерується на сервері перед відправкою до клієнта. Це покращує час завантаження сторінки та сприяє SEO-оптимізації, що є абсолютно необхідною умовою для росту популярності будь-якого онлайн-бізнесу. З іншого боку, SSG дозволяє генерувати сторінки під час збірки додатку, що забезпечує швидке завантаження та високу продуктивність статичного контенту.

Next.js також підтримує автоматичне розбиття коду, що означає, що кожна сторінка завантажує лише ті ресурси, які їй потрібні, замість завантаження великого єдиного пакету коду. Це значно зменшує час завантаження та покращує загальну продуктивність веб-додатку. Крім того, Next.js має вбудовану підтримку міжнародної локалізації (i18n), що дозволяє легко створювати багатомовні веб-сайти.

Ще однією перевагою Next.js є його тісна інтеграція з React, що робить його ідеальним вибором для розробників, які вже знайомі з React. Він дозволяє розробникам використовувати всі звичні можливості React, такі як компонентний підхід та робота зі станами, одночасно надаючи додаткові функції для покращення продуктивності та оптимізації.

У сучасному світі веб-розробки, Next.js вважається одним з найкращих виборів для створення React-додатків з кількох причин. По-перше, він пропонує оптимізоване рішення для SSR та SSG, що є важливим для SEO та швидкості завантаження. По-друге, підтримка автоматичного розбиття коду та оптимізованого завантаження ресурсів у Next.js забезпечує вищу продуктивність веб-додатків, особливо коли мова йде про великі та складні проекти. Це дозволяє розробникам створювати додатки, які завантажуються швидше, пропонуючи кращий досвід користувачам.

Третім важливим аспектом є легкість розгортання та масштабування. Next.js має вбудовані можливості для оптимізації додатків та їх розгортання на різних платформах хостингу, таких як Vercel, що значно спрощує процес розгортання та управління веб-додатками. Крім того, Next.js сприяє масштабованості, дозволяючи легко додавати нові функції та удосконалювати існуючі.

Нарешті, інтеграція з React в Next.js є значним плюсом. Розробники, які вже працюють з React, знайдуть у Next.js знайомий інтерфейс із додатковими перевагами. Вони можуть використовувати всі знайомі патерни проектування React, але з додатковими можливостями, які надає Next.js, включаючи покращену обробку маршрутів, оптимізацію зображень та підтримку API-маршрутів.

У підсумку, провівши детальний аналіз існуючих фреймворків, Next.js в будь-якому разі залишається відмінним вибором для розробників, які бажають створювати сучасні, швидкі та оптимізовані веб-додатки на базі React. Його унікальний набір функцій, легкість використання та висока продуктивність роблять його ідеальним рішенням для розробки веб-додатків в сучасному швидкоплинному цифровому світі.

4.4 Вибір середовища розробки

Neovim був обраний як інтегроване середовище розробки (IDE) для цього проекту з декількох причин. По-перше, Neovim відрізняється високою налаштовуваністю та гнучкістю, що дозволяє адаптувати середовище під конкретні потреби проекту та персональні переваги розробника. Це включає можливість інтеграції різноманітних плагінів, які забезпечують додаткову функціональність, від доповнення коду до управління проектами. По-друге, Neovim пропонує високу швидкість роботи та ефективність, особливо при роботі з великими кодовими базами, завдяки своєму легковазі та оптимізованому інтерфейсу.

Крім того, Neovim підтримує розширений набір скриптових мов, що дозволяє глибоко налаштувати середовище розробки. Це підвищує продуктивність розробника та дозволяє зосередитися на важливих аспектах розробки. Така гнучкість та

масштабованість робить Neovim відмінним вибором для комплексних проектів веб-розробки.

Завершуючи цей розділ, було прийнято рішення створити інформаційну систему у форматі веб-додатку, що забезпечить її доступність з будь-якого пристрою, підключеного до Інтернету, незалежно від операційної системи. Аналізуючи архітектурні підходи для таких додатків, було обрано клієнт-серверну архітектуру, яка реалізується в односторінковому додатку (SPA). Така архітектура підвищить загальну продуктивність системи та зробить її використання більш зручним для користувачів.

Для реалізації проекту було обрано TypeScript як основну мову програмування, а також Next.js у якості фреймворка. Базу даних замінить Sanity CMS. У якості сервісу аутентифікації виступить Clerk.

5 ЗАГАЛЬНА СТРУКТУРНА СХЕМА СИСТЕМИ

Створення структурної схеми[15] є ключовим моментом у процесі створення програмного продукту. Цей етап надає вичерпний огляд щодо глобальної архітектури системи, включаючи її складові частини, взаємозв'язки між ними та їхні функції. Структурна схема також сприяє детальнішому розумінню вибору технологічного стеку та його інтеграції.

Враховуючи технології, що були вибрані для реалізації та поставлені вимоги до системи, було складено структурну схему системи, представлену у Додатку В.

З першого погляду на структурну схему можна побачити, що на ній відсутній окремий блок для серверу. Це зумовлено тим, що у випадку з розробленим додатком Next.js виконує обидві функції. Завдяки серверним компонентам та «server actions», які вийшли в стабільну версію в останній мажорній версії Next.js, тепер працювати з серверним кодом стало настільки просто, що достатньо просто написати функцію, вказати директиву, яка скаже компілятору, що код має виконуватися на сервері, і функція буде виконана на сервері, не розкриваючи нічого, що відбувається всередині на стороні користувача.

Окрім цього, як вже зазначалося раніше, замість бази даних використовується Sanity CMS. Цієї системи управління контентом цілком і повністю вистачає для досягнення поставлених перед проєктом цілей.

Клієнтська частина логіки додатку виконує дві основні функції:

- рендеринг компонентів з директивою, яка каже про рендеринг на стороні клієнта;
- гідрація HTML’у, що прийшов з серверу.

Зокрема, на схемі було прийняте рішення зобразити адміністраторський інтерфейс окремо від користувацького, бо найчастіше вони дійсно використовуватимуться зовсім окремо. Адміністратор навряд чи буде проходити користувацькими шляхами і навпаки.

Також, окрім Sanity, на схемі можна побачити інших провайдерів, які використовуються для авторизації, оплати та відправки листів відповідно.

У якості платіжної систему було обрано Stripe через те, що це рішення має вбудовані примітиви для React та дуже просте і зрозуміле у використанні. На даний момент це ведуча платформа для прийому платежів.

Resend для листування було обрано, тому що автори цього рішення також є авторами чудової бібліотеки react-email, що дозволяє відправляти листи, немовби то це звичайні React компоненти. Інтеграція надто зручна у використанні, щоб нехтувати такою пропозицією.

Clerk є одним з найкращих рішень для авторизації користувачів з Next.js. Через те, що розроблений додаток не використовує власну базу даних, як таку, а Sanity не найкраще місце, щоб зберігати користувацьку базу, було прийнято рішення обрати централізоване рішення, яке забирає всі потенційні питання з приводу безпеки на себе. Також, ця інтеграція дозволяє зручним чином зберігати мета-дані користувачів в одному місці. Аналізувати користувацьку базу дуже просто, бо Clerk має вбудовану панель керування, яка має всі необхідні функції, щоб легко побачити що відбувається з користувачами в системі.

Для спілкування з базою даних використовується мова запитів GROQ (Graph-Relational Object Queries). Своєю природою досвід праці з Sanity та GROQ нагадує NoSQL бази даних, що є досить зручним рішенням для проблем, які стоять перед продуктом. Швидкі та гнучкі запити, знижена строгість – все це допомагає розвивати платформу швидко і будувати більш ефективні запити. Деякі функції відсутні, наприклад відношення між документами в дві сторони, але це не так критично для такого додатку, який будується в рамках роботи.

Для хостингу і безперервної інтеграції (CI) використано платформу Vercel. Опис процесу розгортання можна побачити в додатку Г.

6 СИСТЕМА КЕРУВАННЯ ЗМІСТОМ

CMS, або система керування вмістом, – це програмний додаток, який дозволяє користувачам створювати, керувати та змінювати вміст на веб-сайті без спеціальних технічних знань. По суті, він забезпечує зручний інтерфейс для керування вмістом веб-сайту.

Беручи до уваги основні цілі поставлені перед продуктом, що розробляється в рамках даної роботи, CMS гарно підійде в якості заміни для бази даних. Проте, це не означає, що це щось зовсім відмінне від СУБД. Навпаки, це програмне забезпечення, яке за своєю суттю дуже схоже на NoSQL бази даних. Sanity використовує GROQ в якості мови запитів, якою досить зручно користуватися.

Орієнтовний формат даних можна побачити на ER-діаграмі, зображеній на додатку Д.

6.1 Інфологічне проектування

Перед створенням структури бази даних краще за все створити перший варіант концептуальної розробки. Цей процес, як правило, має на увазі формування початкового варіанту логічної структури БД. Ця структура використовується для опису концепцій бази даних через інтегровані блоки.

Так як в даному проекті використовується нетрадиційна база даних, то деякі конструкції, характерні такому проектуванню будуть опускатися. Зокрема, FK (Foreign key), PK (Primary key) тощо. Тож, пропонується побудувати загальну орієнтовну структуру даних.

В процесі розробки даної моделі будемо оперувати наступними трьома термінами:

- сутність – структурний елемент, деяка абстракція об'єкту реального світу, явища або процесу, що містить атрибути;
- атрибут – характеристика сутності;
- зв'язок – елемент взаємодії сутностей.

На основі встановлених критеріїв та аналізу сценаріїв використання системи було визначено набір ключових елементів та їх характеристик, які необхідні для ефективної роботи системи (див. таблицю 6.1). Крім того, була розроблена ER-діаграма бази даних, деталі якої представлені у Додатку Д.

Таблиця 6.1 – Перелік основних сутностей та їх атрибутів

Сутності	Атрибути
Продукти	Назва, Опис, Ціна, Скорочена назва, Фото, Знижка, Ціна Ціна зі знижкою (computed), Обраний, Посилання, Категорія
Категорії	Назва, Фото, Скорочена назва
Секції	Тип, Продукти, Текст, Фото, Фото (масив), Стиль тексту
Колекції	Назва, Опис, Скорочена назва, Продукти, Секції
Сторінки	Секції, Скорочена назва
Теги	Назва, Обов’язковий, Значення, Бедж
Запити	Назва, Email, Тема, Повідомлення, Телефон, Відреаговано
Навігація	Заголовок, Посилання, Піднавігація
Email шаблони	Тема, Контент, Посилання
Баннери	Посилання, позиція, фото, текст
Відгуки	Нікнейм, Email, Рейтинг, Контент
Замовлення	Контент, Email, Адреса, Заплачено, Статус
FAQ	Назва, Контент
Соціальні мережі	Провайдер, Посилання

Продовження таблиці 6.1

FAQGroup	Ім’я, Фото, Дочірні елементии

6.2 Опис схеми бази даних

Після розробки ER-діаграми здійснено фізичне моделювання БД. У таблицях 6.2-6.22 зазначено всі сутності, які існують в системі з їх зв’язками. Будемо використовувати наступні позначки:

- R – required (обов’язковий)
- DEF – default (базове значення)

Таблиця 6.2 – Продукт

Назва поля	Опис	Тип даних	Позначки
_id	Унікальний ідентифікатор	CUID	R
name	Назва	string	R
description	Опис	string	
price	Ціна	number	R
discount	Ім’я	number	0 <= x <= 100
slug	Скорочена назва	string	R
featured	Обраний	boolean	NN, UN
category	Категорія	reference	to: category, R
views	Перегляди	number	DEF(0)
images	Фото	Image[]	R
tags	Теги	reference	to: tag

Таблиця 6.3 – Категорія

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
name	Назва	string	R
image	Фото	Image	
slug	Скорочена назва	string	R

Таблиця 6.4 – Секція

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
type	Тип	string	R
products	Продукти	reference	to: items
text	Текст	string	
image	Фото	string	
images	Фото[]	[string]	
text	Текст	string	

Таблиця 6.5 – Колекція

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
name	Назва	string	R
description	Опис	string	
slug	Скорочена назва	string	R

Продовження таблиці 6.5

items	Продукти	reference	
sections	Секції	reference	

Таблиця 6.6 – Сторінка

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
sections	Секції	reference	R
slug	Скорочена назва	string	R

Таблиця 6.7 – Тег

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
value	Значення	string	R
tagGroup	Група	reference	R

Таблиця 6.8 – Група тегів

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
name	Назва	string	R
required	Обов’язковість	boolean	R, DEF(false)

Таблиця 6.9 – Запит

Назва поля	Опис	Тип даних	Обмеження

Продовження таблиці 6.9

_id	Унікальний ідентифікатор	CUID	R
topic	Тема	string	R
email	Електронна пошта	string	R
message	Повідомлення	string	R
phone	Продукти	string	
actioned	Відреаговано	boolean	R, DEF(false)

Таблиця 6.10 – Навігація

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
title	Заголовок	string	R
link	Посилання	string	R
subnav	Під-навігація	object	R

Таблиця 6.11 – Email шаблон

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
subject	Тема	string	R
content	Контент	markdown	R
link	Посилання	string	

Таблиця 6.12 – Банер

Назва поля	Опис	Тип даних	Обмеження

Продовженяя таблиці 6.12

_id	Унікальний ідентифікатор	CUID	R
link	Посилання	string	
position	Позиція	string	R
photo	Фото	string	R
text	Текст	string	

Таблиця 6.13 – Відгук

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
username	Нікнейм	string	
email	Електронна пошта	string	R
rating	Рейтинг	string	R
content	Контент	string	R

Таблиця 6.14 – Замовлення

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
username	Нікнейм	string	
email	Електронна пошта	string	R
rating	Рейтинг	string	R
content	Контент	string	R

Таблиця 6.15 – FAQ

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
title	Заголовок	string	R
content	Контент	markdown	R

Таблиця 6.16 – група FAQ

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
name	Назва	string	R
image	Фото	string	R
children	Дитячі елементи	array	R

Таблиця 6.17 – Соціальне посилання

Назва поля	Опис	Тип даних	Обмеження
_id	Унікальний ідентифікатор	CUID	R
provider	Мережа	string	R
url	Посилання	string	R

В даному розділі було розглянуто структуру системи керування даними для розробленого продукту, націлену на роботу з ювелірним магазином з можливістю підвищенної модифікації контенту.

7 РОЗРОБКА ІНТЕРФЕЙСУ СИСТЕМИ

В сфері бізнесу, для якої розробляється проект, однією з найголовніших складових є якість розробленого інтерфейсу користувача. Цей фактор напряму впливає на можливість клієнта ефективно орієнтуватися в просторі продукції та, як наслідок, на прибуток для бізнесу, який він може принести. Без якісного інтерфейсу дуже складно конкурувати з іншими учасниками ринку, тому що вони мали набагато більше часу для того, щоб їх розробники попрацювали над зовнішнім виглядом веб-ресурсу.

Важливо також сказати про те, що дуже велика кількість користувачів користується Інтернетом з мобільних пристроїв, тому окрім гарних кольорів і розмітки, правильно виконаний інтерфейс має також бути адаптованим для телефонів, планшетів, тощо. Перше враження користувача сильно повпливає на його вибір щодо використання системи, тим паче придбання продукції. Крім того, персоналізовані рекомендації та можливість легко порівнювати продукти можуть покращити враження від покупок, що призведе до збільшення продажів. Вони мають бути доступні для кожного.

Інтерфейс – це уособлення статусу компанії. Він має бути досконалим. Для досягнення такої якості, треба взяти до уваги такі складові:

- привабливість та вишуканість (UI);
- зручність у використанні (UX).

UI (User Interface) має на увазі сучасний дизайн, обмірковане поєднання кольорів, вибір шрифту, налагодженість візуальних компонентів на сторінках, тощо. Це та складова, яка дуже швидко визначить чи захоче користувач продовжувати використовувати систему. Також, ця складова визначає наскільки легко та інтуїтивно користувач зможе виконати ті дії, які йому потрібно, не вдаючись до зв’язку з підтримкою.

В свою чергу UX (User Experience) – це складова, якість якої реалізується за допомогою функціональної наповненості, зручності у використанні, швидкості відповідей на запити, та ін. Загалом, за відсутності якісного UX користувачеві буде

складно орієнтуватися у системі та складно розуміти як саме або чи взагалі йому вдалося виконати дію, яку він хоче.

Отже, базуючись на технічних особливостях продукту, а також беручи до уваги сферу, де продукт буде використовуватися, необхідно розробити дизайн для частини сайту, що буде відкрита до клієнтів, а також частини для адміністраторів. Версія для клієнтського використання має бути доступна для всіх потенційних користувачів, а адміністративна частина має бути легкою у використанні в першу чергу з десктопних систем.

Враховуючи раніше визначені вимоги, проведено розробку графічних інтерфейсів з дотриманням високої якості обох вище зазначених складових

7.1 Клієнтська частина

У розробці інтерфейсу клієнтського сайту, особливу увагу було приділено заохоченню уваги користувача при першому відвідуванні. Після використання багатого спектру навігаційних можливостей, основною метою було забезпечити фільтрування даних та легкість подальшого конвертування клієнта. Було приділено особливу увагу реалізації сценаріїв, поданих на користувацькій діаграмі діяльності, яку можна побачити в додатку Е.

7.1.1 Маркетингова сторінка

Першим етапом була розробка дизайну головної сторінки сайту (рисунок 7.1). Ця сторінка має на меті познайомити користувача з продуктом, надати опції для навігації, які визначив адміністратор, та дозволити подальшу роботу з сайтом.

На головній сторінці одразу стає доступним контент, який адміністратор ресурсу додав власноруч. Починаючи від банеру та закінчуючи навігацією, абсолютно все медіа наповнення повністю контролюється адміністратором.

Мобільний вигляд цієї сторінки запропонований на рисунку 7.2.

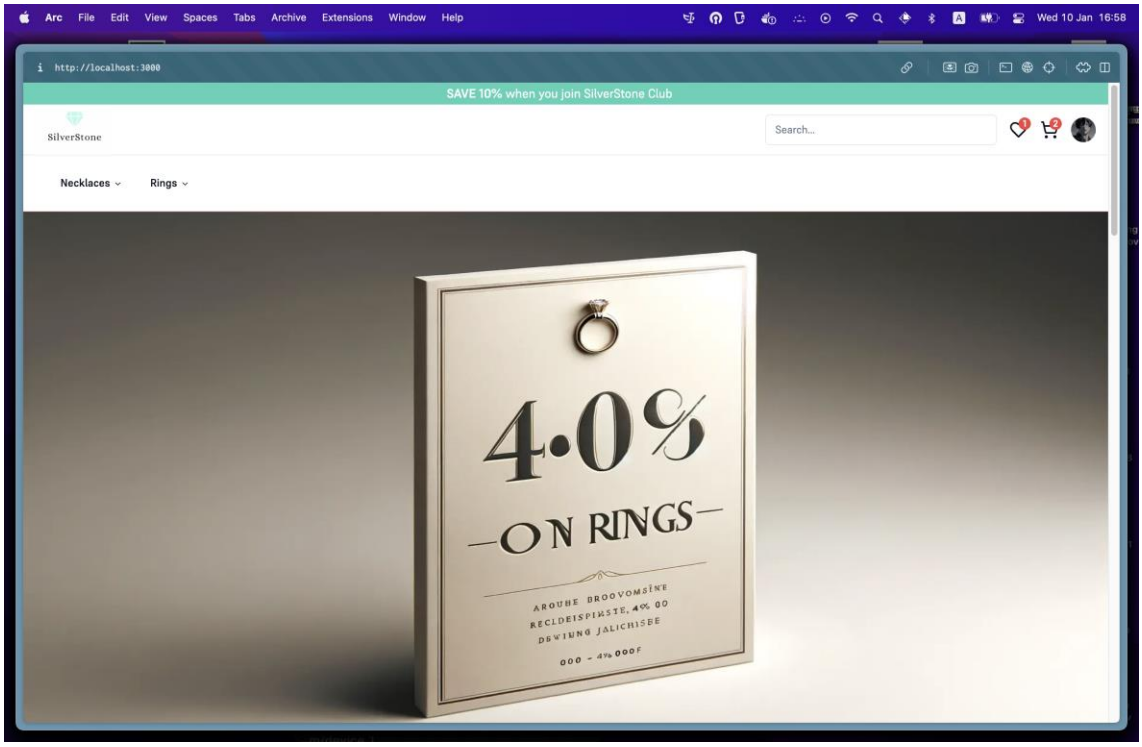

Рисунок 7.1 – Маркетингова сторінка

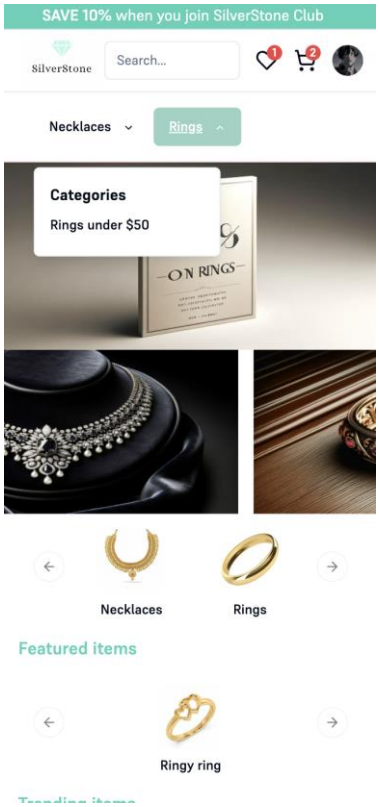

Рисунок 7.2 – Вигляд головної сторінки на мобільному пристрої

Одразу ж, з першого контакту з веб-системою, користувач (клієнт) має можливість всілякої навігації, що була визначена адміністратором, а також йому представлено медіа-контент, який адміністратор вважає доцільним. Всі ці банери і

слайдери у додатку є можливість відключити, про що поговоримо трохи пізніше. На цій сторінці користувач отримує доступ до:

- банери та слайдери з промо-акціями;
- навігація зверху екране, що визначена адміністратором;
- навігація за категоріями;
- обрані для відображення адміністратором особливі товари;
- популярні товари;
- нещодавно переглянуті.

Таким чином, забезпечується направлення користувача до контенту, який адміністратор вважає потрібним, а також надається можливість всіма можливими способами потрапити до головного наповнення ресурсу – товарів.

Знизу, користувачеві запропоновані різноманітні корисні посилання, які також визначає адміністратор та може сам їх створити. Також, з використанням Metals API представлено ціни на різні коштовні матеріали. Ці дані знаходяться в кеші, тому доступ до них є без будь-яких обмежень. Візуалізація цієї секції сторінки представлена на рисунку 7.3.

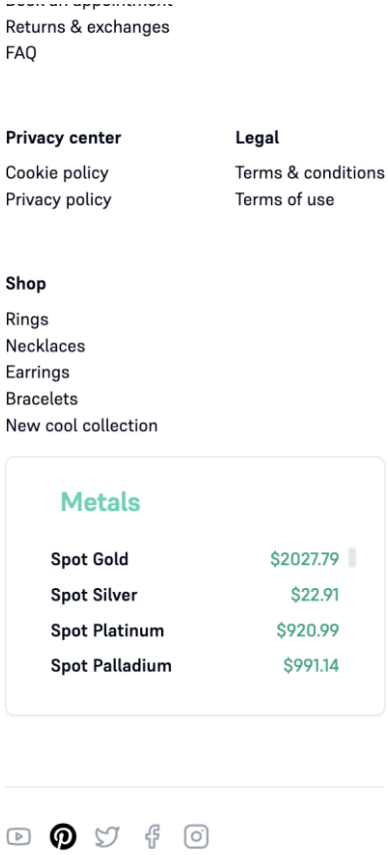

Рисунок 7.3 – Нижня частина головного макету (footer)

7.1.2 Список товарів

При переході до будь-якої категорії товарів з використанням раніше зазначених методів навігації, користувачеві представлено список товарів з визначеним адміністратором набором фільтрів (рис. 7.4, рис. 7.5). Є підтримка сортування та відображення окремих сторінок з товарами (pagination).

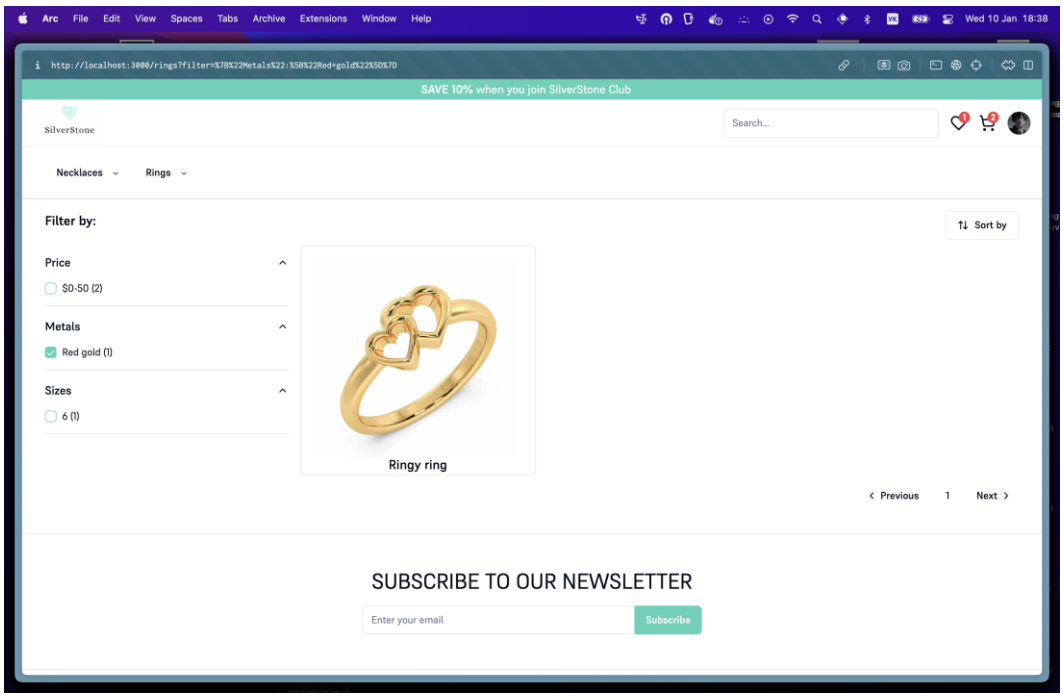

Рисунок 7.4 – Сторінка категорії товарів

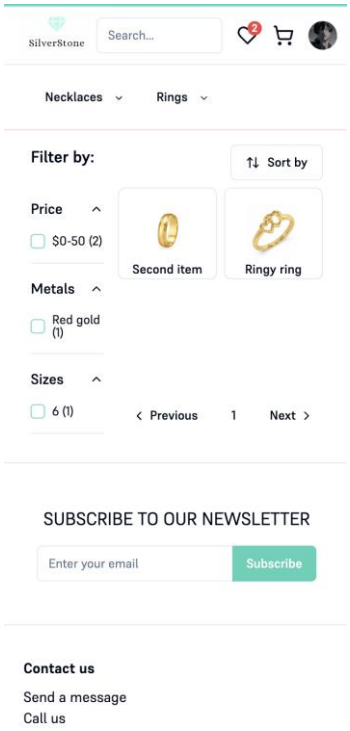

Рисунок 7.5 – Сторінка категорії товарів адаптована для мобільних

7.1.3 Сторінка товару

Обираючи товар зі списку, користувач потрапляє на сторінку продукту. Тут він може обрати який саме продукт йому потрібен, обравши зі списку тегів, які знову ж визначає адміністратор. На прикладі з рисунку 7.6 продемонстровано варіант, коли зазначені теги потребують від користувача обрати метал та розмір. Так як було визначено лише по одному тегу з кожної категорії, відображається лише один варіант.

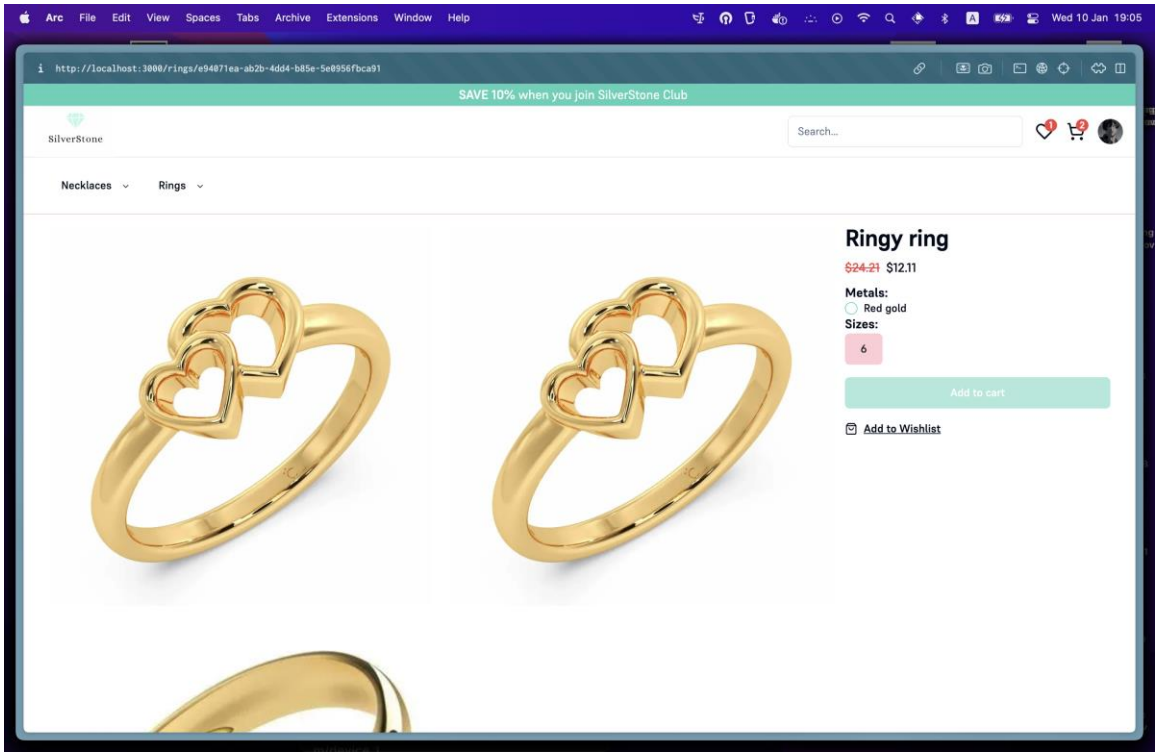

Рисунок 7.5 – Сторінка товару

На цьому ж екрані у користувача є можливість додати товару до списку бажаного (wishlist).

Такий список товарів містить корисну інформацію не лише для користувача, але й для системи. Завдяки тому, що ця логіка існує, в систему є можливість додати більш інформативну аналітику, у майбутньому ввести можливість налаштування листування, коли один з товарів зі списку стає доступним за знижкою.

На рисунку 7.7 продемонстровано варіант даної сторінки для мобільних пристроїв.

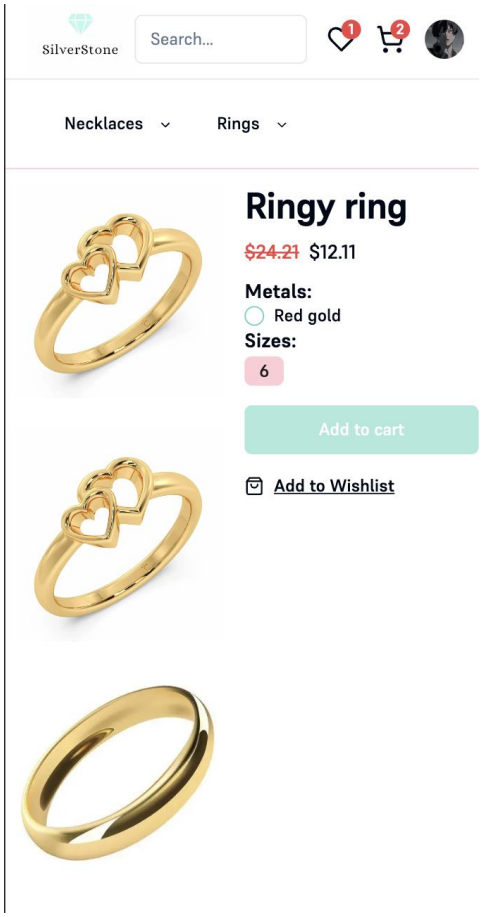

Рисунок 7.7 – Сторінка товару адаптована для мобільних

На тій самій сторінці є можливість залишити відгук, як показано на рис. 7.8.

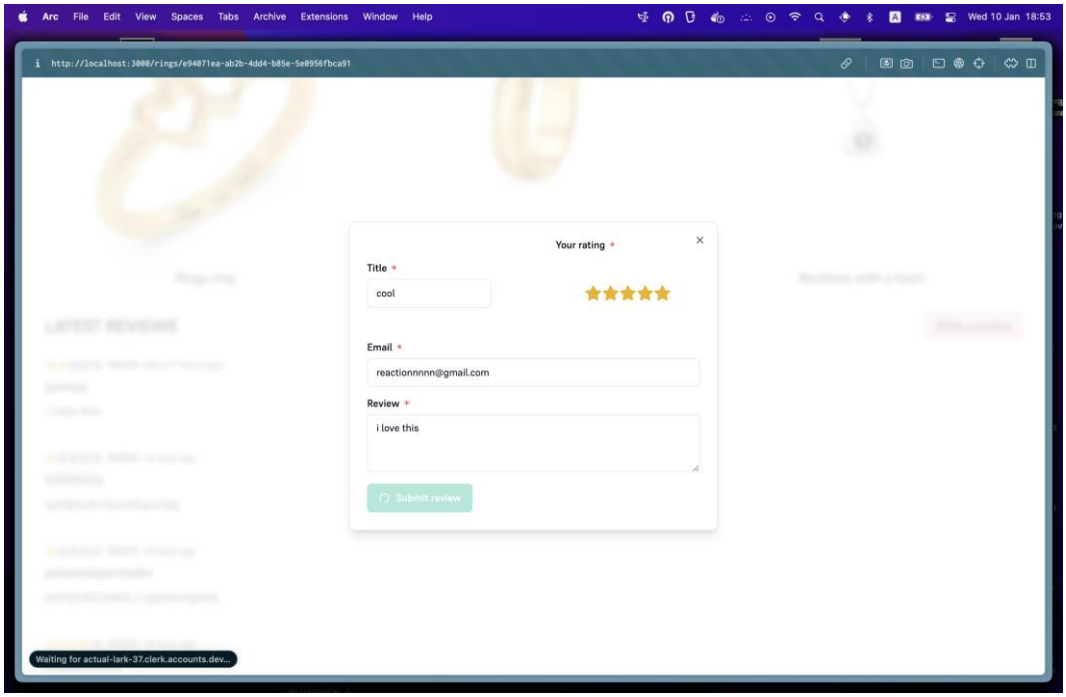

Рисунок 7.8 – Функціонал відгуку

Завдяки вдалому використанню кешу Next.js, відгук одразу потрапляє на сторінку, і має вигляд як продемонстровано на рис. 7.9.

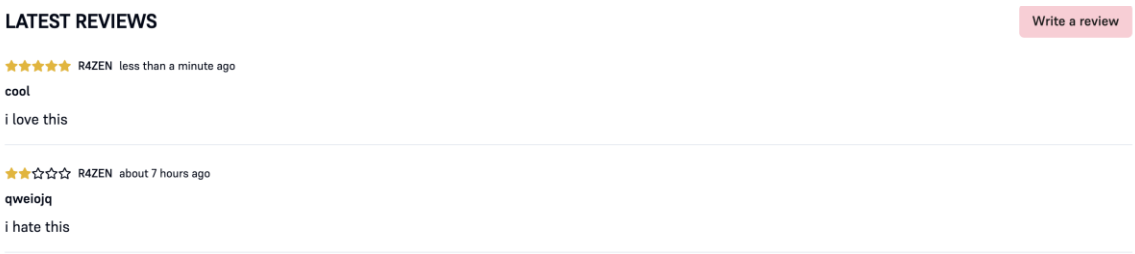

Рисунок 7.9 – Список відгуків

7.1.4 Сторінка кошику

Після додавання товару до кошику, логічним наступним кроком було б зробити замовлення, тому далі розглянемо саме цю сторінку, візуальна репрезентація якої подана на рисунку 7.10. Цей етап разом з попереднім реалізовує діаграму станів, подану у додатку Ж.

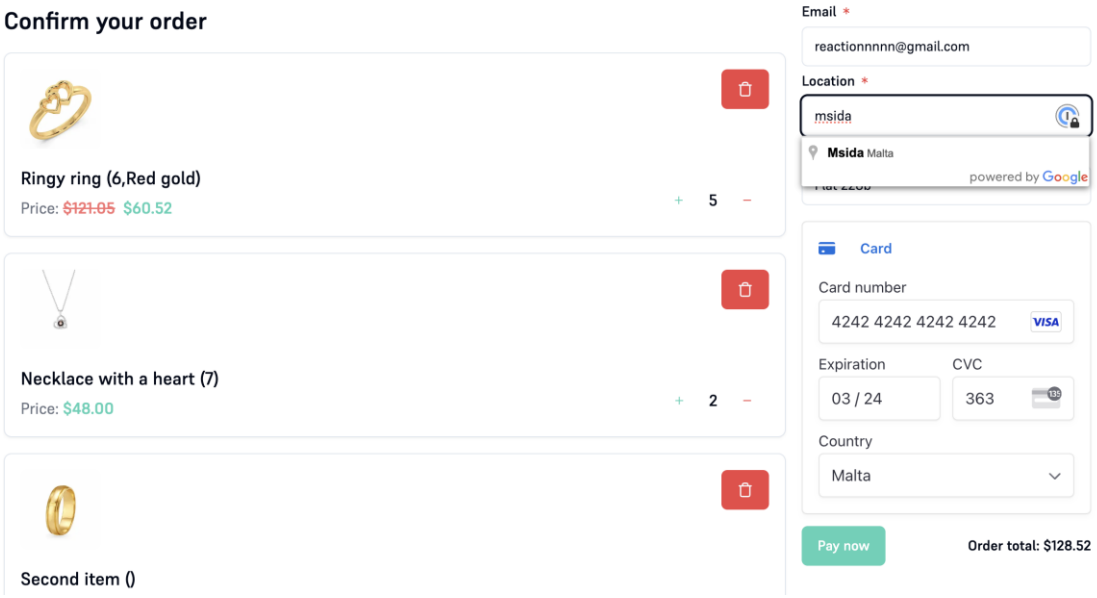

Рисунок 7.10 – Сторінка кошику

Поля, які пропонується заповнити користувачу такі:

- поштова скринька, куди прийде підтвердження;
- місто;
- повна адреса;
- платіжні дані.

Варто зазначити використання Google API для простішого набору міста замовника, що впливає на загальну якість UX.

На рисунку 7.11 подано вигляд даної сторінки для мобільних пристроїв.

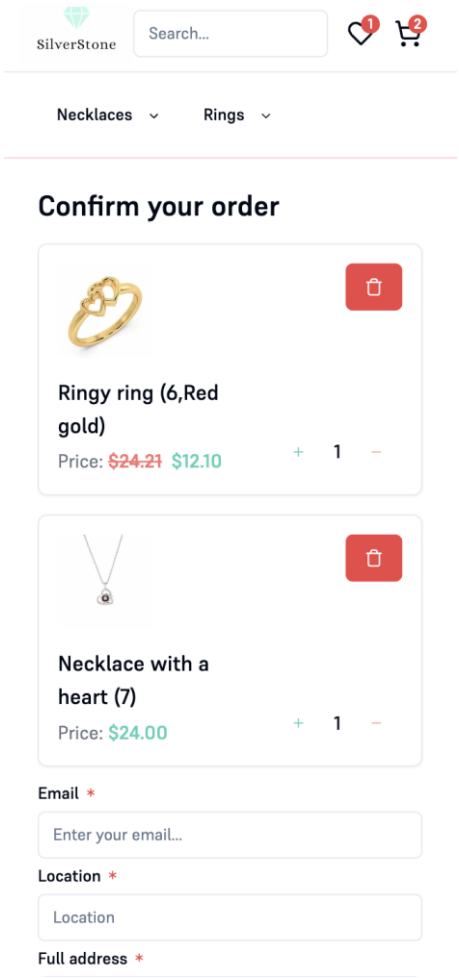

Рисунок 7.11 – Сторінка кошику на мобільних пристроях

Поля, які пропонується заповнити користувачу такі:

- поштова скринька, куди прийде підтвердження;
- місто;
- повна адреса;
- платіжні дані.

Варто зазначити використання Google API для простішого набору міста замовника, що впливає на загальну якість UX.

Після підтвердження замовлення, на зазначену поштову скриньку надходить лист, схожий на той, що зображено на рисунку 7.12. Такий самий лист надходить на адміністраторську поштову скриньку.

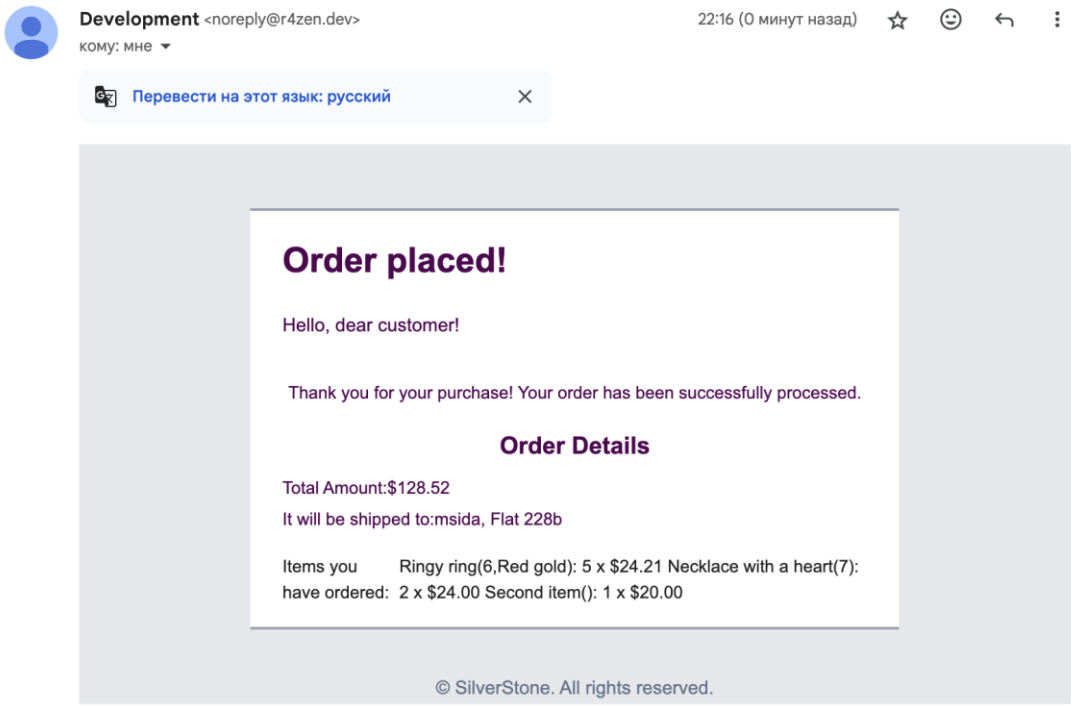

Рисунок 7.12 – Лист, що свідчить про підтвердження замовлення

7.1.5 Налаштовувані розділи

Користувацький інтерфейс також має деякі особливі секції, які адміністратор може або ввімкнути, або вимкнути у налаштуваннях, які містять різноманітні набори продуктів, згруповані за певними характеристиками. Їх візуальна репрезентація подана на рисунках 7.13, 7.14, 7.15, 7.16.

Рисунок 7.13 – Секція «Вам також може сподобатися» або «Товари в тренді»

Цей набор товарів відображає товари, які збирають найбільше переглядів та додавань до кошику.

Рисунок 7.14 – Секція «Обрані товари»

Цей набор товарів адміністратор може налаштовувати сам, обираючи які саме товари будуть обрані для відображення в ньому шляхом ввімкнення відповідного перемикача в адміністраторській панелі, про яку трохи нижче.

Рисунок 7.15 – Секція «Переглянуті товари»

Цей розділ також може бути ввімкнений або ні через налаштування. Він відображає для користувача товари, які він щойно переглядав.

7.1.6 Інші деталі основних сторінок

Щоб не залишити без уваги деякий важливий функціонал, який є дещо менш специфічним, бо доступ до нього є з будь-якої сторінки, пропонується також розглянути такі функції, як маркетингова розсилка (рис. 7.16) та пошук (рис. 7.17).

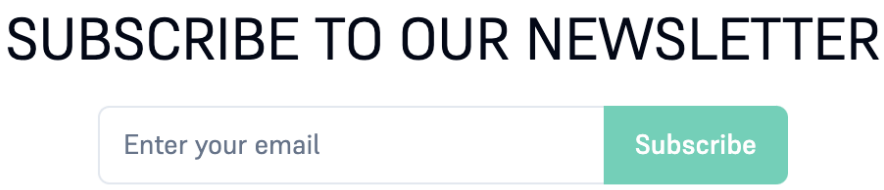

Рисунок 7.16 – Підписка на розсилку

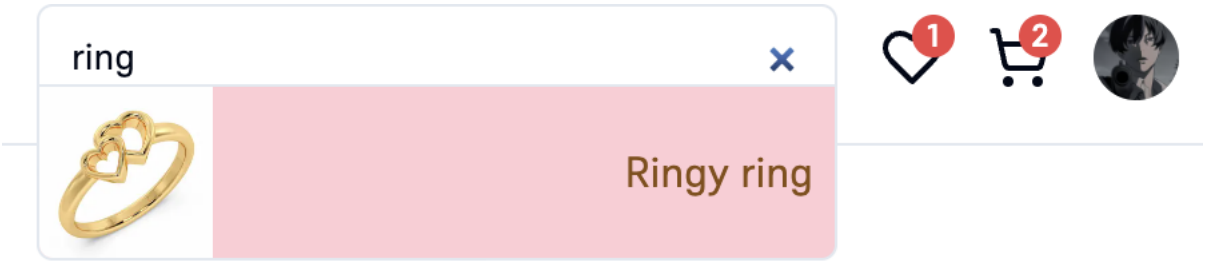

Рисунок 7.17 – Пошук товарів та сторінок

7.1.7 Спеціальні створювані сторінки

Як було зазначено у попередніх розділах, одна з провідних особливостей системи є можливість створювати маркетингові або інформаційні сторінки буквально з нуля, які будуть якісно індексовані пошуковими системами. Приклад такої сторінки та її вигляд на мобільному пристрої подано на рисунку 7.18.

Рисунок 7.18 – Спеціальна створена сторінка

7.1.8 Форма зворотнього зв’язку

Користувач також може звернутися до адміністрації з питанням або відгуком за допомогою форми зворотнього зв’язку (рис. 7.19). Після заповнення всіх необхідних полів і відправки форми, поштовий лист надійде на адміністраторську поштову скриньку, де він зможе зконтактувати з клієнтом (рис. 7.20).

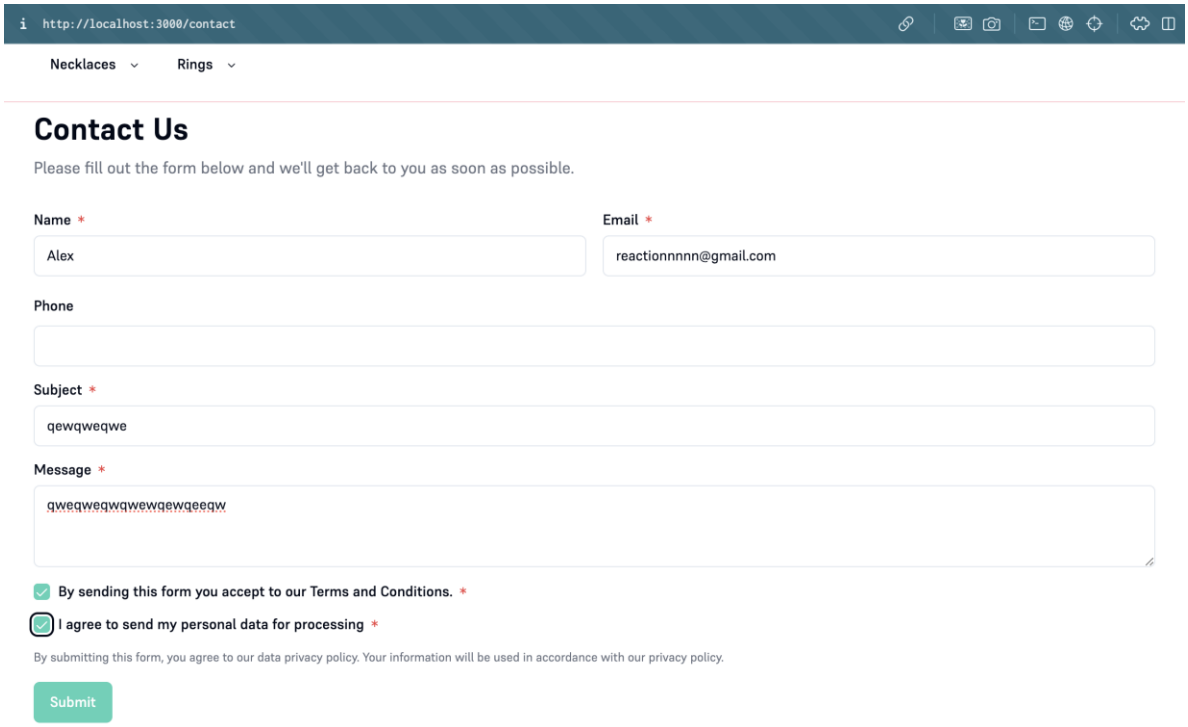

Рисунок 7.19 – Форма зворотнього зв’язку

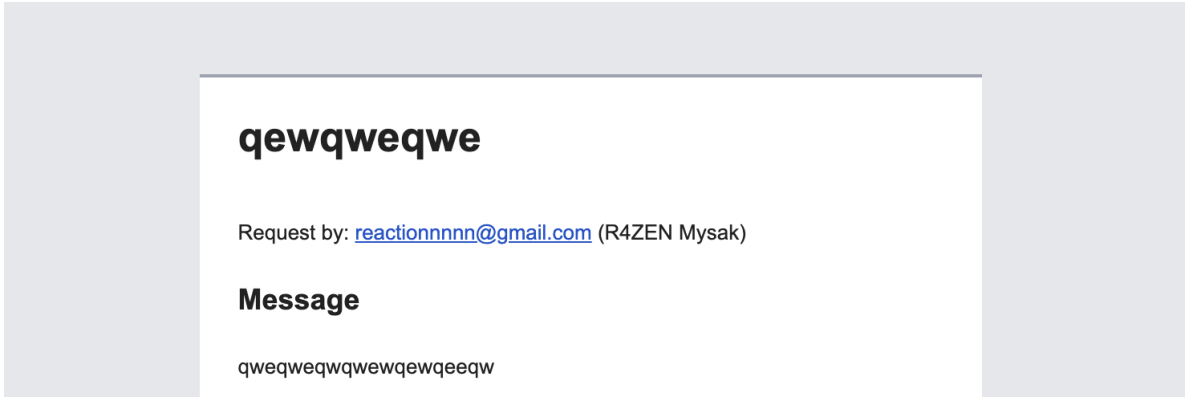

Рисунок 7.20 – Приклад поштового листа від клієнта

7.2 Адміністраторська панель

Для роботи зі сторони власника веб-системи було розроблено адміністраторську панель. Вона знаходиться дуже близько до користувацького інтерфейсу, а саме розміщена на одному і тому ж самому домені. Це надає можливість швидко працювати з даними та сторінками привілейованим користувачам. В цьому розділі розглянемо деякі з основних частин контенту, які може редагувати адміністратор.

7.2.1 Створення товару

Створювати товари в додатку неймовірно просто. Для цього існує вкладка «Items», де подано наступний функціонал:

- створення нового товару;
- видалення товарів;
- оновлення існуючих товарів.

Вигляд даної вкладки подано на рисунку 7.21.

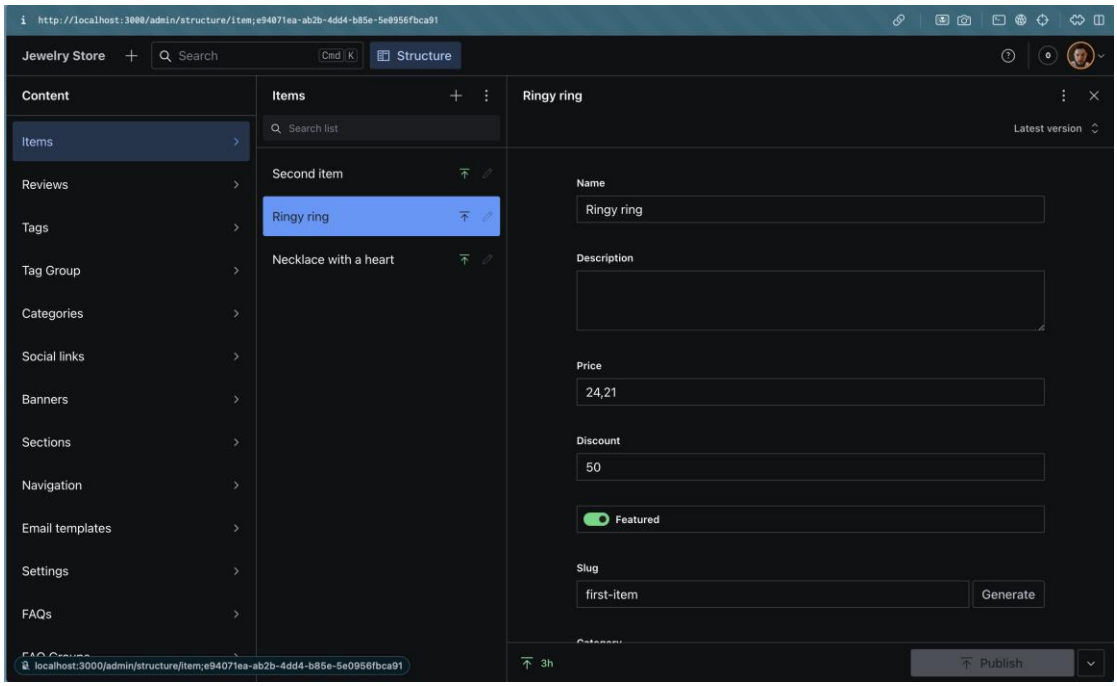

Рисунок 7.21 – Вкладка товару

7.2.2 Створення спеціальної сторінки

Для створення сторінки з нуля адміністратору запропоновано підхід з використанням секцій, які можуть бути різних типів і, відповідно, відображати різний контент. Вкладка сторінок виглядає дуже просто (рис. 7.22) і дає можливість додати ці секції, як показано на рисунку 7.23.

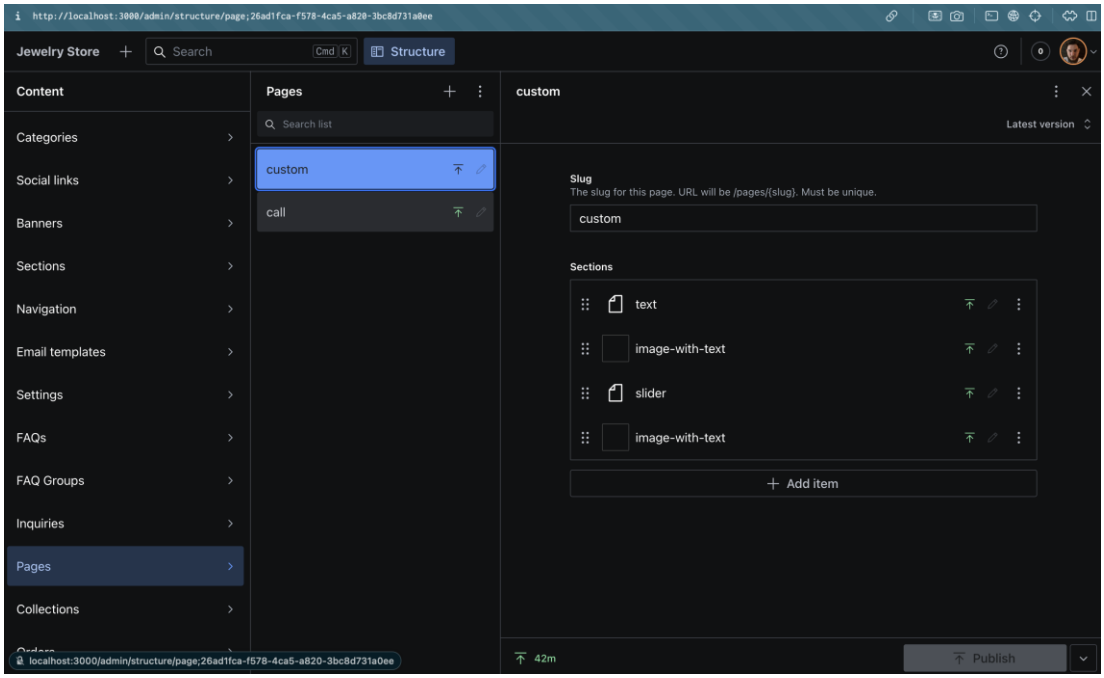

Рисунок 7.22 – Вкладка спеціальних сторінок

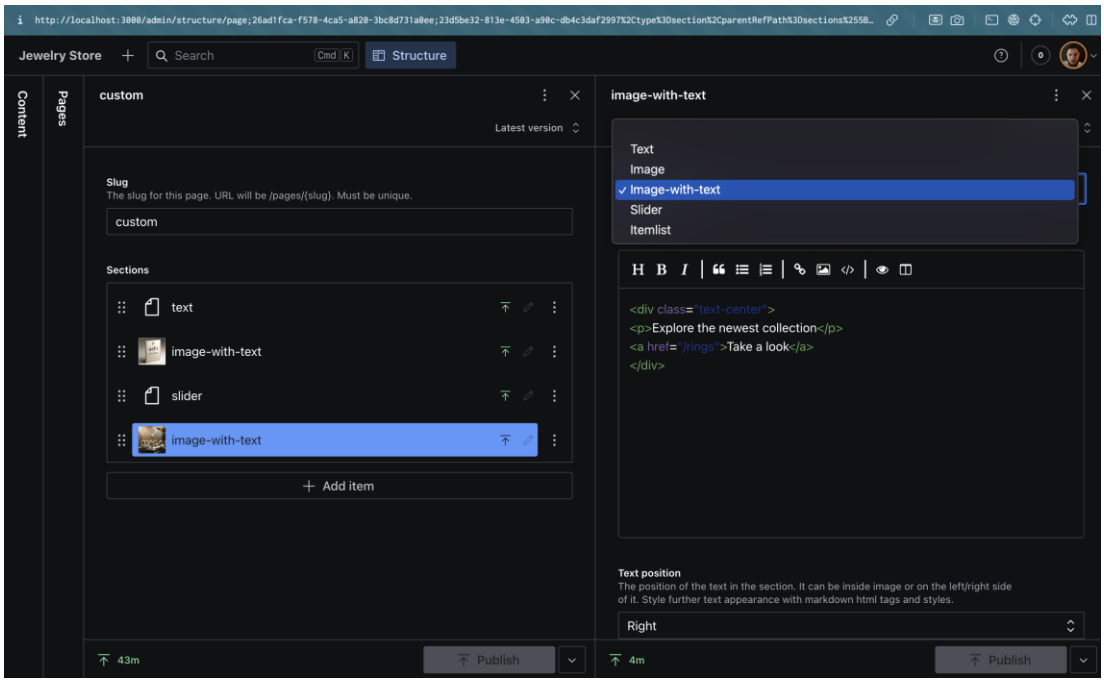

Рисунок 7.23 – Створення секції

Варто зазначити, що одна з особливостей системи полягає у можливості використовувати HTML разом з Tailwind CSS для накладання стилів і досягання високого рівня гнучкості налаштування. Звісно, це не є обов'язковим, тому навіть якщо просто написати текст, він відобразиться досить гарно для очей користувача. Проте, така інтеграція дає змогу просунути різноманітність контенту ще на один рівень вище.

7.2.3 Налаштування

На рисунку 7.24 подано налаштування системи, де адміністратор може обрати які зі стандартних налаштувань (секцій, банерів, тощо.) він хоче мати на своєму веб-сайті.

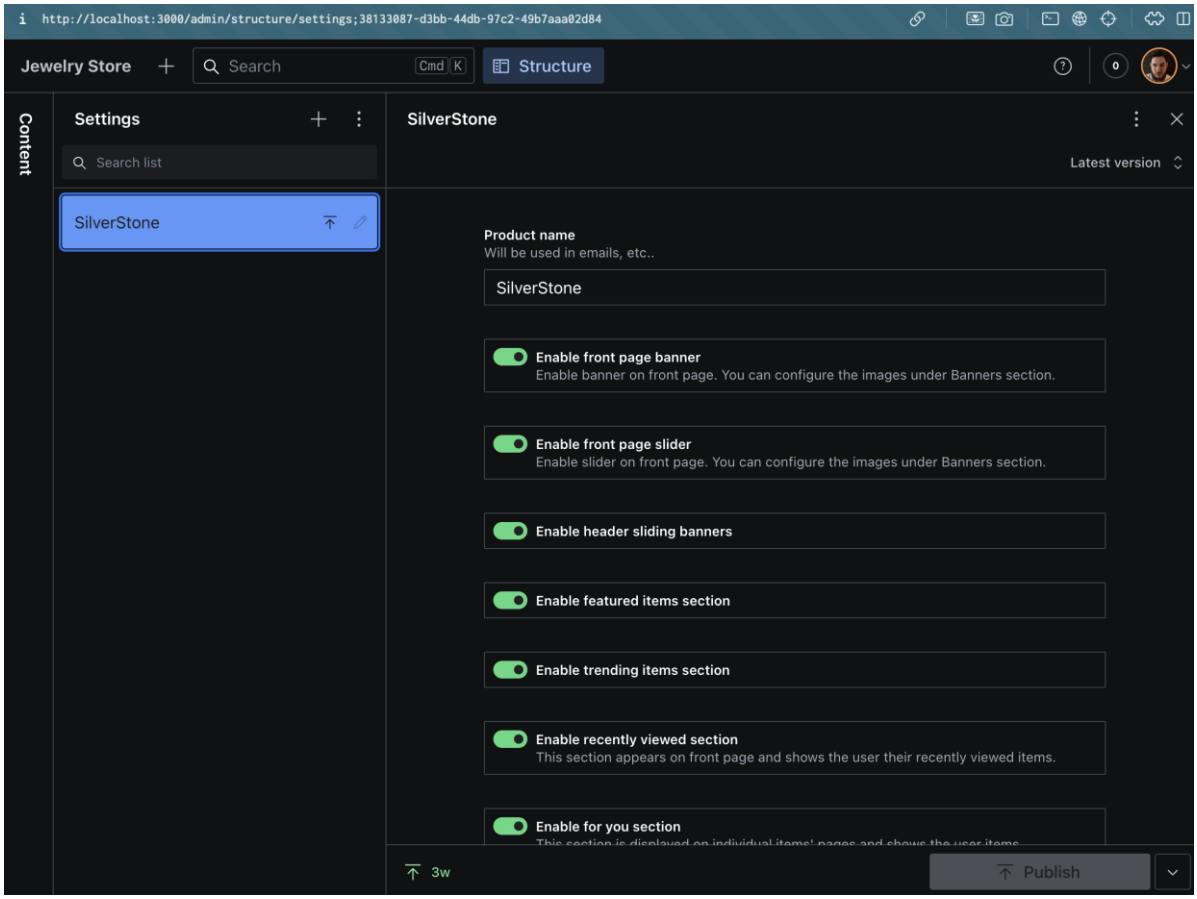

Рисунок 7.24 – Налаштування системи

7.2.4 Листування

Для забезпечення маркетингового листування адміністратору надається можливість створювати такі листи власноруч за допомогою Markdown. На рисунку 7.25 показано як може виглядати приклад такого листа. Контент, що задається адміністратором використовується в заздалегідь визначеному шаблоні і надсилається всім, хто підписався на листування. Для відправлення такої розсилки є можливість використання функції листування в адміністраторській приладовій дошці, як показано на рисунку 7.26.

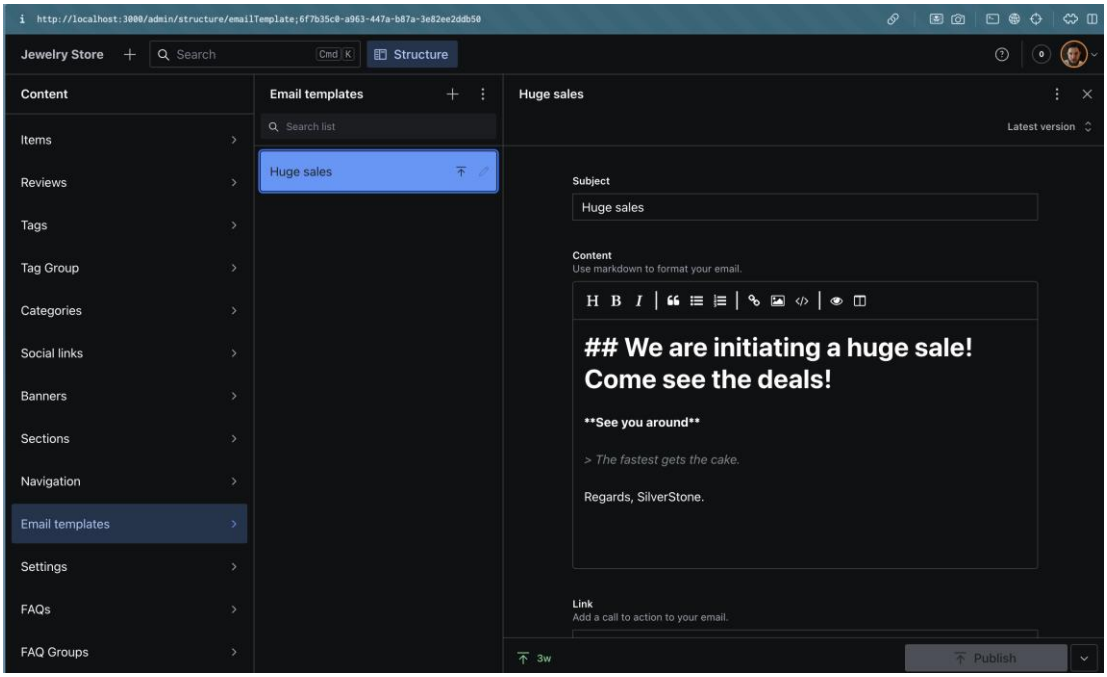

Рисунок 7.25 – Створення листа

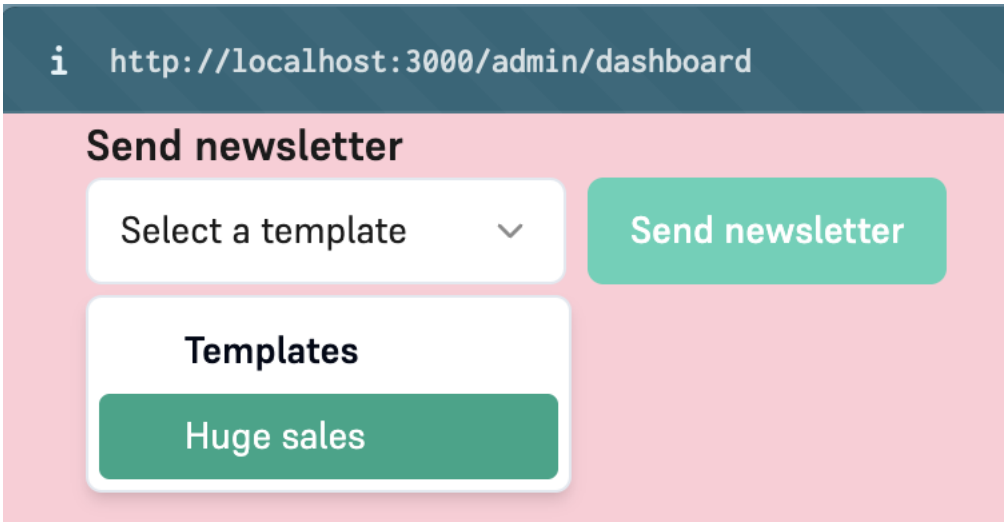

Рисунок 7.26 – Створення розсилки

7.2.5 Авторизація та користувачі

У веб-системі, що розробляється, авторизація є необов’язковою. Вона може бути потрібна лише для того, щоб залишити відгук або рецензію. Проте така опція все одно надається. У панелі користувача він може побачити всі деталі, що стосуються його аккаунту (рис. 7.27).

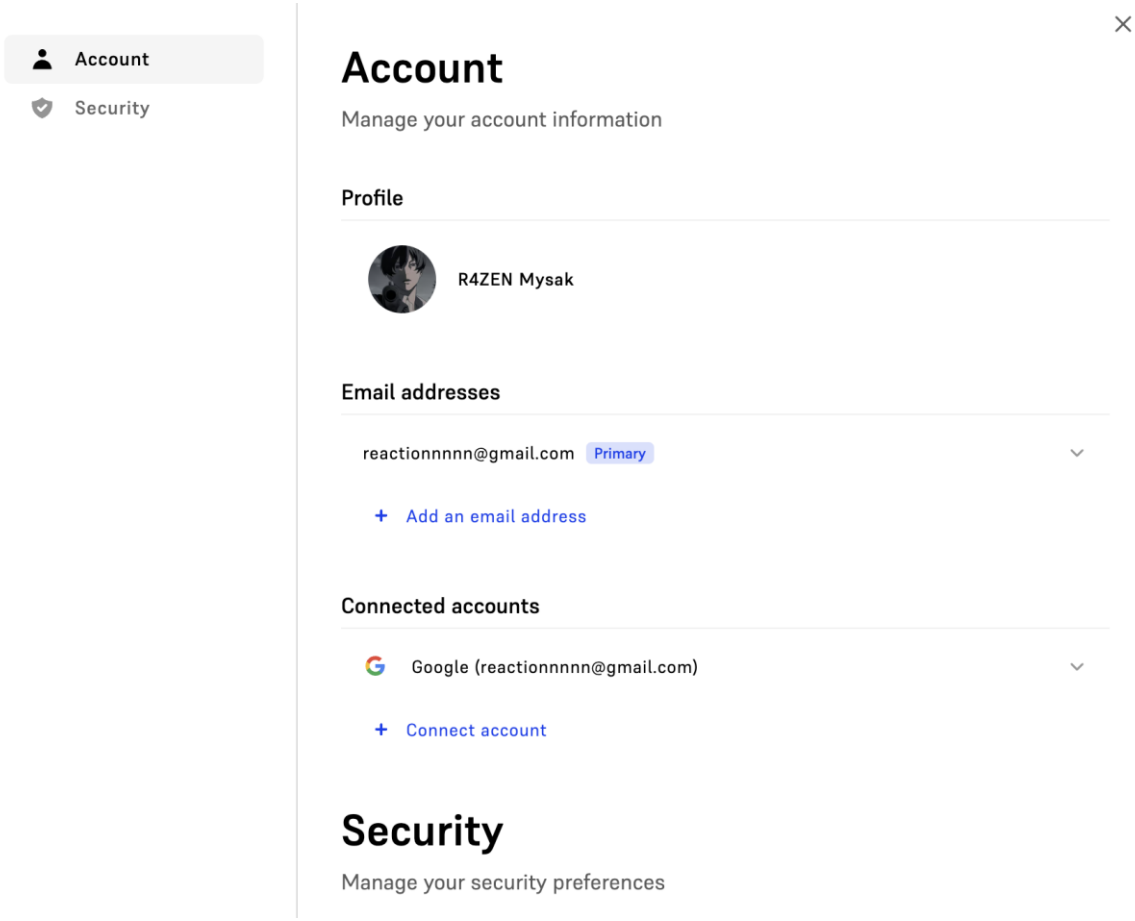

Рисунок 7.27 – Панель користувача

З іншої сторони, для адміністраторів доступна панель керування користувачами (рис. 7.28), яка також використовується для перегляду інформації щодо тих, хто підписаний на розсилку, адже для них створюється користувач в будь-якому випадку. Вони не можуть увійти на сайт використовуючи створеного користувача, проте цей запис все одно існує для відображення підписників.

Доступ до цієї панелі можна отримати перейшовши до облікового запису свого проекту у системі авторизації Clerk.

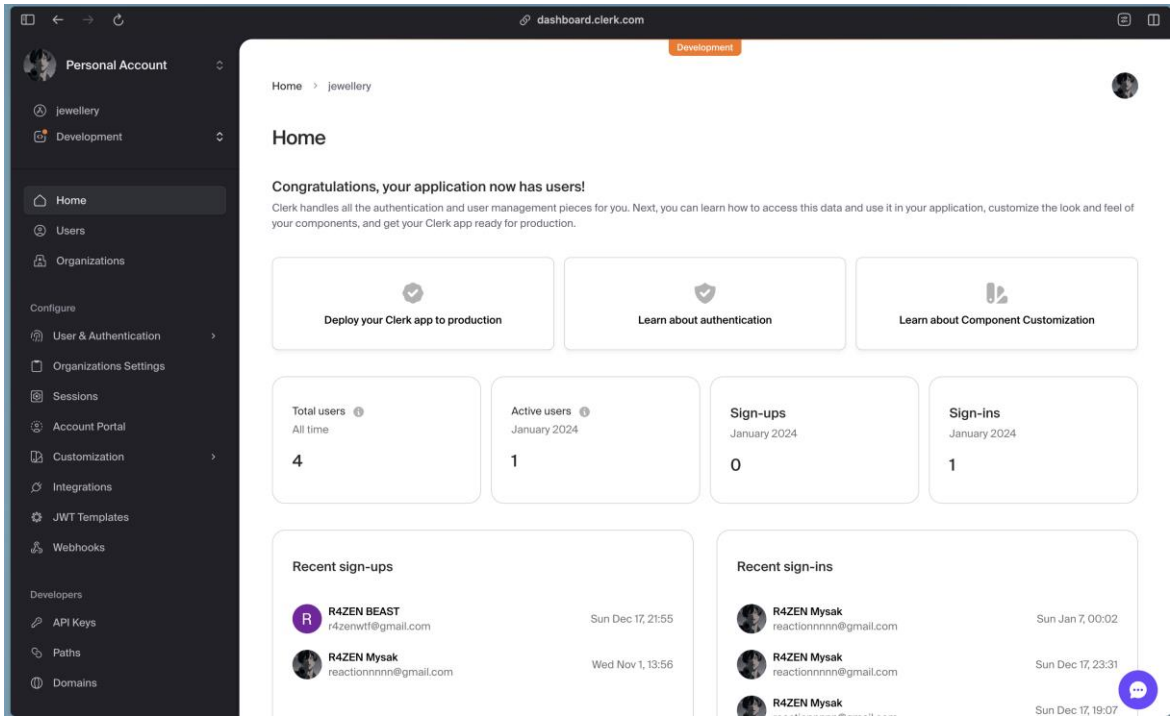

Рисунок 7.28 – Адмін-панель користувачів

7.2.6 Аналітика

У веб-системі існує базова інтеграція аналітики (рис. 7.29), яка є, мабуть, єдиною частиною додатку, яку адміністратор не може налаштувати без втручання в вихідний код програми, бо вона потребує визначення яка саме дія користувача буде надсилати аналітику про ту чи іншу подію. На щастя, така можливість існує, бо доступ до продукту надається у вигляді цілісного коду, тож будь-які модифікації можуть бути виконані при потребі, з належними навичками розуміння коду.

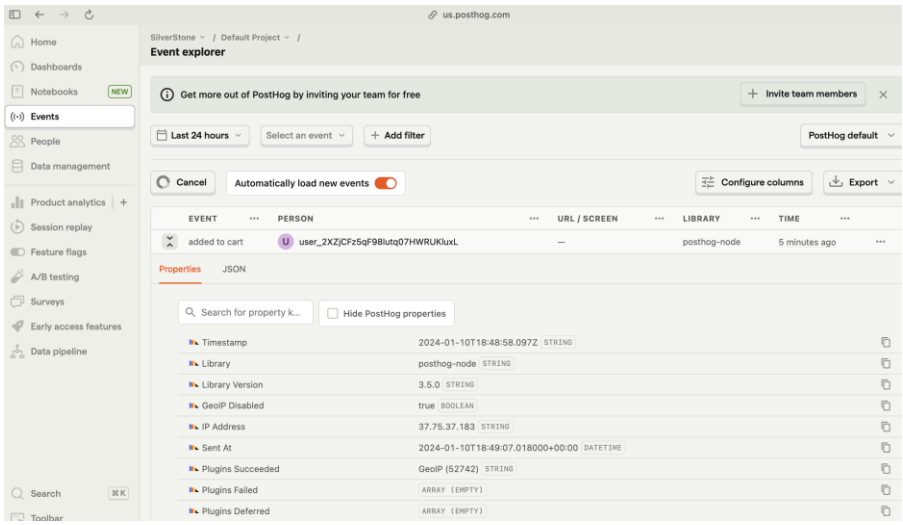

Рисунок 7.29 – Панель аналітики

Доступ до панелі аналітики здійснюється через перехід на ресурс PostHog, де користувач може створити аккаунт перед використанням продукту. Варто зазначити, що аналітика є опціональною.

В даному розділі було розроблено два різновиди інтерфейсу: клієнтський та адміністраторський. Обидва інтерфейси мають підтримку комп’ютерного та мобільного вигляду для забезпечення доступності будь-якому клієнтові. Розроблені інтерфейси відповідають потребам відповідних груп користувачів, які були описані в додатках Е та И, забезпечують високі стандарти UI/UX дизайну та повністю надають функціонал, визначений вимогами.

8 ПРОГРАМНА РЕАЛІЗАЦІЯ

8.1 Загальні принципи та архітектура

Як вже було зазначено в попередніх розділах, для розробки системи було обрано Next.js в якості клієнтської та серверної частини, тому архітектура додатку вийшла досить простою.

Розрізняють два основні типи архітектури: монолітну та мікросервісну.

Мікросервісна архітектура має на увазі розділення серверної частини (іноді також клієнтської) на окремі сервіси, кожний з яких, як правило, відповідає за відведений для нього функціонал. Така архітектура у корпоративному світі надає можливість назначити відповідальними за кожний з них певну команду розробників, які спеціалізуються у роботі з технологіями, що необхідні для забезпечення головної функції цих сервісів. Також, за умови якісного проектування системи, можна досягти незалежність сервісів один від одного (low coupling), що може дозволити всій системі залишатися у функціональному стані, навіть якщо щось пішло не так у одному з сервісів і він пішов в оф-лайн. Ця архітектура дозволяє ефективно масштабувати систему та обирати різні технології для вирішення різних конкретних проблем, що знижує складність планування та надає більшу гнучкість системі.

У свою чергу, монолітна архітектура зазвичай побудована таким чином, що весь додаток працює як одне ціле з усіма його частинами. Іноді вони також можуть жити окремо одне від одного, проте таке їх існування не має майже ніякого сенсу.

У випадку з системою, що розробляється в контексті даної роботи, поведінка, яку нав’язує моноліт цілком влаштовує всі потреби системи. Навіть з урахуванням всіх плюсів мікросервісної архітектури, такий вид систем має тенденцію швидко набувати складності і зазвичай має на увазі нижчу швидкість розробки для однієї людини. Тому для розроблюваної системи було обрано саме монолітну архітектуру.

Під час розробки було приділено увагу загальним принципам програмування, що забезпечують чистоту, читабельність та здатність коду до масштабування. Серед них:

- модульність коду;
- DRY;

- мінімальна зв’язність коду (low coupling);
- логічне групування коду (high cohesion);
- мінімальне повторювання коду.

За дотримання таких простих принципів досягається висока якість коду та, як наслідок, висока якість системи і її можливостей. Вони також впливають на кількість часу, що необхідна для розробки такої системи і спрощують багато задач до мінімальних необхідних змін.

8.2 Серверна частина (Back-end)

У серверній частині системи було реалізовано роботу з зовнішніми сервісами та сховищами, що необхідні для реалізації поставлених вимог. Серед них:

- Sanity CMS для роботи з даними;
- Clerk для авторизації;
- PostHog для аналітики;
- Resend для листування;
- Stripe для здійснення оплати.

Розроблена система гарно підходить для розташування у безсерверному середовищі, а використання зазначених провайдерів робить наявність серверу не обов’язковою, бо тримає дані в централізованих сховищах. Далі розглянемо кожен з них, які саме проблеми вони вирішують і як реалізовано роботу з ними.

8.2.1 Sanity CMS

У якості системи керування змістом, що також виступає в ролі бази даних для деяких операцій всередині системи, було обрано Sanity CMS. Ця система використовує мову побудови запитів GROQ, що дозволяє ефективно працювати з даними та будувати гнучкі запити, які не завжди вдасться легко побудувати при використанні SQL.

За допомогою цього сховища (або правильніше, коли кажуть про системи керування змістом, – озера контенту) реалізовано роботу майже з усіма даними в системі, надано можливість легкого наповнення, а також маніпуляції цими даними,

через вбудований інтерфейс адміністративної панелі. Це сховище є однією з найсучасніших платформ, яка також має вбудовані інтеграції для провідних бібліотек, що дозволяє дуже швидко адаптувати цю технологію у сучасний проект.

На рисунку 8.1 зображено приклад побудови запиту з використанням мови запитів GROQ. Як можна бачити, чимось це нагадує SQL, проте легший для розуміння і більш адаптований саме під пошук даних, про що кажуть такі оператори як “|” (pipe) і функції, які можна використовувати з ними (наприклад “order”, “count”, тощо.)

```
export async function getProductReviews(id: string): Promise<Review[]> {
  try {
    return await sanityFetch({
      query: groq`*[_type == "review" && item->_id == $id]{
        _id,
        _createdAt,
        title,
        content,
        rating,
        username,
        userId,
      } | order(_createdAt desc)`,
      params: { id },
      tags: [GET_REVIEWS_TAG],
    });
  } catch (err) {
    logger.error({ err }, `Error fetching reviews for item with id ${id}`);
    return [];
  }
}
```

Рисунок 8.1 – Приклад запиту GROQ

Щоб продемонструвати контраст з SQL, пропонується поглянути на рисунок 8.2 і аналог JOIN, що виглядає як children[]-> і дозволяє розгорнути пов’язані об’єкти повністю, або частково.

Також є можливість діставати поля з обробляючогося запису, подібно AS в SQL, але набагато простіше: “image”: image.asset->url. Цей запис каже: «дістань url з поля asset об’єкту з поля image на записі, що обробляється».

```
export async function getFAQGroups(): Promise<FAQGroup[]> {
  try {
    return await sanityFetch({
      query: groq`
      *[_type == "faqGroup"]{
        _id,
        _createdAt,
        name,
        "image": image.asset->url,
        children[]->
      }`,
      tags: [GET_FAQ_GROUPS_TAG],
      cache: "no-cache",
    });
  } catch (err) {
    logger.error({ err }, "Error fetching faq groups");
    return [];
  }
}      You, 3 weeks ago • mailing, custom pages, cart, check
```

Рисунок 8.2 – SQL-подібні операції в GROQ

Для запитів, здійснених за допомогою Sanity є спеціальний механізм роботи з кешем, щоб не перенавантажувати мережу і мати прямий доступ до кешу всередені системи. Це дозволяє оперувати даними з найбільш високим рівнем ефективності та оновлювати їх за потреби[16].

8.2.2 Clerk

Clerk забезпечує роботу з користувачами всередині додатку. У системі, що розробляється, авторизація не є важливою складовою, проте мати її ніколи не завадить, наприклад, для кращої аналітики, для контролю розсилок, тощо.

З використанням цього сервісу налаштування авторизації в системі було справою декількох годин. Потужність наданих примітивів вражає своєю оманливаю простотою та широким спектром можливостей.

На рисунку 8.3 зображено приклад використання Clerk. У шести рядках коду вдалося запрограмувати відображення кнопки користувача, де він може отримати доступ до свого особистого кабінету, а також кнопку авторизації, яка направить його до сторінки, де він зможе її пройти, в разі якщо він ще не авторизований.

Рисунок 8.3 – Приклад використання Clerk

Так як в системі, що розробляється, навряд чи колись буде вимога до нестандартних шляхів роботи з авторизацією Clerk є ідеальним рішенням для отримання мінімального функціоналу з вбудованими примітивами, щоб збільшити можливості системи.

8.2.3 PostHog

Сервіс аналітики (і не тільки) PostHog дозволяє ефективно збирати дані про користувацькі дії для розвитку бізнесу у правильному напрямку. Будь-яка система хоче знати якомога більше про те, як її користувачі працюють з нею. Саме тому було додано сервіс збору такої інформації.

На рисунку 8.4 представлено варіант використання цього сервісу для відправки події додавання товару в кошик. Також в системі є події, наприклад, для перегляду товарів. Все це дозволяє знати більше про те, що подобається користувачам, і приймати вигідні бізнес-рішення.

```
export async function addToCart(id: string) {
  const user = await currentUser();

  addToCartCookie(id);
  sendEvent({ distinctId: user?.id ?? "unauthenticated", event: "added to cart" });
}
```

Рисунок 8.4 – Приклад використання PostHog

8.2.4 Resend

Resend – це бібліотка для відправки електронних листів. Зазвичай цей сервіс (його SDK) використовується в тандемі з React Email, що дозволяє писати зміст листів не кидаючи семантичні особливості бібліотеки, яка використовується в усіх інших частинах системи. Така ж сама техніка використовується і в розроблюваній веб-системі.

На рисунку 8.5 показано як може виглядати приклад коду для відправки електронного листа, де EmailTemplate – React-компонент побудований за допомогою React Email (рис 8.6).

```
try {
  const from =
    env.NODE_ENV === "development"
      ? "Development <newsletter@r4zen.dev>"
      : `Newsletter <newsletter@${host}>`;

  const link =
    env.NODE_ENV === "development"
      ? "http://localhost:3000"
      : `https://${host}`;

  const { data, error } = await resend.emails.send({
    from: from ?? "newsletter@jewellery.dev",
    to: userEmail,
    subject: restProps.subject,
    react: EmailTemplate({
      ...restProps,
      name: user.firstName ?? user.username,
      productName: settings.productName,
      link,
    }),
  });

  if (error) {      You, 3 weeks ago • mailing, custo
    return { error };
  }

  return { id: data?.id };
} catch (error) {
  console.log(error);
  return { error };
}
```

Рисунок 8.5 – Приклад використання Resend

```
<Body className="bg-gray-200 pt-12">
  <Container className="border-y-2 border-solid border-gray-400 bg-white px-6">
    <Heading>{subject}</Heading>

    <Text>Hey{name ? `, ${name}` : " there"}!</Text>

    <Section>
      <div dangerouslySetInnerHTML={{ __html: marked.parse(content) }} />
    </Section>

    {link && (
      <Section className="text-center">
        <Button
          href={link}
          className="mx-auto my-6 rounded-lg bg-primary px-8 py-3 text-center text-lg text-primary-foreground"
        >
          Go to {productName}
        </Button>
      </Section>
    )}
  </Container>

  <Text className="my-8 text-center text-slate-500">
    &copy; {productName}. All rights reserved.
  </Text>
</Body>
```

Рисунок 8.6 – React Email компонент

8.2.5 Stripe

Платіжна система Stripe[17] використовується в системі для прийому платежів від клієнтів, що оформили замовлення. На даний момент це один з найвідоміших і найповніших за функціоналом сервісів.

Цей сервіс також надає корисні примітиви, як то наприклад клієнтський компонент-форму для введення платіжних даних і SDK для створення та підтвердження платежів.

На рис. 8.7 надано приклад коду, який дозволяє відобразити форму для внесення даних, яка одразу зв’язана з цілісним шляхом користувача при оформленні замовлення. Це дозволило дуже швидко та без особливих зусиль інтегрувати функціонал цієї платформи у систему.

Рисунок 8.7 – Інтеграція Stripe

8.3 Клієнтська частина (Front-end)

По стандартам нових серверних компонентів (React Server Components) взагалі майже весь код в системі виконується на стороні серверу, а лише потім доставляється клієнту. Це стосується як читання даних, так і запитів до їх модифікації.

Тому по-перше важливо розповісти як саме відбувається доставка візуального змісту до користувача. По-друге необхідно визначитись з архітектурним підходом до розробки користувацького інтерфейсу для вдалого поєднання двох частин додатку: клієнтської та адміністративної.

Філософія серверних компонентів і Next.js спонукає за можливості запитувати дані на стороні серверу[18]. Це зумовлено рядом переваг та дозволяє користуватися новітніми методами роботи з даними у веб-просторі.

На рисунку 8.8 зображено приклад звертання до серверу напряму в серверному компоненті. Поданий код виконується на стороні серверу, після чого отримані дані стають доступними для рендерингу всередині діалектної мови JSX, яка потім конвертується в HTML і доставляється до клієнту. Приклад використання даних всередині JSX подано на рисунку 8.9.

```
const productsNavLinks = await getNavLinks("products");
const footerNavLinks = await getNavLinks("footer");
const socialLinks = await getSocialLinks();
const settings = await getSettings();
```

Рисунок 8.8 – Приклад запиту на читання даних

```
{socialLinks.map((item) => (
  <a
    key={item.provider}
    href={item.url}
    className="h-4 w-4 text-gray-400 hover:text-gray-500"
  >
    <span className="sr-only">{item.provider}</span>
    {providerToImageMapper[item.provider]}
  </a>
))}
```

Рисунок 8.9 – Приклад використання даних

Вирішивши питання з даними, необхідно побудувати структуру проекту таким чином, щоб було інтуїтивно зрозуміло куди йти, щоб досягнути тієї чи іншої зміни. На рисунку 8.10 зображено розташування файлів всередині бази коду, де вони згруповані з певною логікою.

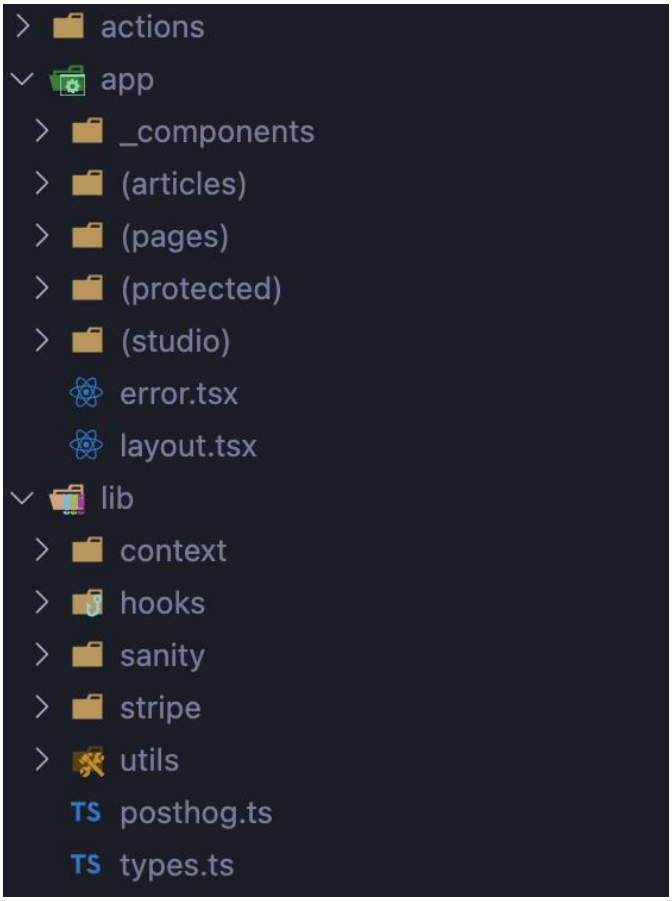

Рисунок 8.10 – Основні архітектурні частини системи

Серед таких груп:

- actions – Next.js Server Actions[19] (рис 8.11);
- app – сторінки системи погруповані за логічним змістом;
- lib – папка з допоміжними компонентами системи, а також сторонніми сервісами.

```
"use server";
     You, 3 weeks ago • mailing, custom pages, cart, checkout, orders, co…
import { revalidateTag } from "next/cache";
import { type Input as Infer } from "valibot";

import { submitReview } from "~/lib/sanity/queries";
import { GET_REVIEWS_TAG } from "~/lib/sanity/tags";
import { type ReviewFormSchema } from "~/lib/utils/form-validators/review";

export async function createReview(
  data: Infer<typeof ReviewFormSchema> & { id: string }
) {
  await submitReview(data);
  revalidateTag(GET_REVIEWS_TAG);
}
```

Рисунок 8.11 – Server Action

Server Actions дозволяють викликати код, що виконується на стороні серверу, з будь-якої частини додатку, таким чином зберігаючи дані серверу у безпеці, але дуже близько до користувацької частини. В цьому полягає їх основна перевага.

Ці actions викликаються в системі тоді, коли потрібно виконати будь-яку дію з даними, до яких клієнтська частина має лише частковий доступ, наприклад для створення замовлення, або відправки листів.

Всього для досягнення поставлених цілей було розроблено наступні сторінки:

Для клієнтської частини:

- макет публічних (маркетингових) сторінок;
- сторінка товарів;
- сторінка товару;
- сторінка кошику;
- сторінка списку бажаного;
- сторінка успішного замовлення;
- сторінка зворотнього зв’язку;
- сторінка частих запитань;
- сторінка успішного замовлення;

- шаблон для спеціальних створених сторінок;
- навігаційне меню;
- низ сайту.

Для адміністративної частини:

- шаблон панелі інструментів;
- спеціальні компоненти для інтерфейсу Sanity (Markdown та ін.).

А також:

- кнопки для зміни сторінок для великих датасетів;
- кнопки дій;
- слайдери з медіа-контентом;
- та ін.

Всі компоненти перерахувати не вдасться, тому що поряд з цими основними сторінками та компонентами було розроблено допоміжні компоненти, які допомогають прискорити розробку та мінімізують повторюваність коду.

9 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Стартап – це часто про новаторські ідеї, які можуть бути удосконаленням існуючих рішень або внесенням чогось абсолютно нового на ринок. Однією з основних цілей, які переслідує такий проект – залучення фінансування для його розвитку та втілення в життя. Цьому сприяє приваблення інвестицій або отримання банківських кредитів. У обох випадках важливо покладатися на глибокий аналіз ринку та потенційних конкурентів, а також зосередитися на демонстрації комерційної привабливості проекту.

В цьому розділі буде розглянуто стартап-проект та проведено його ринковий аналіз, який визначить чи може розроблена система конкурувати на ринку, а також можливі плани для втілення ідеї проекту в реальність.

9.1 Опис ідеї проекту

Першим кроком буде опис ідеї, завдяки чому можна буде окреслити бізнес-ринок, в рамках якого буде проводитися аналіз. Цей опис зображено у таблиці 9.1.

Таблиця 9.1 – Опис ідеї стартап-проекту

Зміст ідеї	Напрямки застосування	Вигоди для користувача
Веб-система для ведення діяльності ювелірного магазину з підтримкою спеціальних сторінок та розділів, максимальною налаштованістю це простий, ефективний та дешевий варіант вийти на ринок та почати розвивати свій бізнес	Розвиток бізнесу у сфері онлайн-продажів, а саме ювелірних товарів	Для власника бізнесу це простий, дешевий варіант, який має високий шанс задовольнити всі його потреби через високу налаштованість та оптимізацію в плані швидкості та індексованості ресурсу

Базуючись на описі стартап-проекту, було здійснено аналіз потенційних техніко-економічних переваг у порівнянні з конкурентними рішеннями. Його результати наведені у таблиці 9.2.

Таблиця 9.2 – Визначення характеристик ідеї проекту

Техніко-економічні характеристики ідеї	Системи конкурентів				W (слабка сторона)	N (нейтральна сторона)	S (сильна сторона)
	Даний проект	Shopify	Woo Commerce	Pandora			
Створення спеціальних сторінок адміністратором без участі технічних спеціалістів	Підтримується	Частково підтримується	Частково підтримується	Не підтримується	+		
Додавання товарів та їх відображення для продажу	Підтримується	Підтримується	Підтримується	Підтримується		+	
Створення платежів та вбудована монетизація	Підтримується	Підтримується	Підтримується	Підтримується		+	

Продовження таблиці 9.2

Вбудована якісна фільтрація продукції яка може бути налаштована адміністратором	Підтримується	Частково підтримується	Частково підтримується	Частково підтримується	+		
Адміністративна панель з можливістю розширення	Підтримується	Підтримується	Підтримується	Частково підтримується		+	
Авторизація користувачів	Підтримується	Підтримується	Підтримується	Підтримується		+	
Підписка на маркетингову розсилку	Підтримується	Підтримується	Підтримується	Підтримується		+	
Вбудовані відгуки	Підтримується	Підтримується	Частково підтримується	Підтримується		+	
Задані макети для сторінок без необхідності їх налаштування для досягнення потрібного вигляду	Частково підтримується	Підтримується	Підтримується	Не підтримується			+

9.2 Технологічний аудит ідеї проекту

В даному підрозділі було розглянуто технологічну здійсненність ідеї проекту. Дані цього аудиту наведені у таблиці 9.3.

Таблиця 9.3 – Технологічна здійсненність ідеї проєкту

№ п/п	Ідея проекту	Технології її реалізації	Наявність технологій	Доступність технологій
1	Веб-система для ведення діяльності ювелірного магазину з підтримкою спеціальних сторінок та розділів	Neovim	Наявна	Безкоштовний доступ
2		Next.js	Наявна	Безкоштовний доступ
3		Sanity CMS	Наявна	Безкоштовний доступ без підвищення плану
4		Clerk	Наявна	Безкоштовний доступ без підвищення плану
5		PostHog	Наявна	Безкоштовний доступ без підвищення плану
6		Stripe	Наявна	Безкоштовний доступ без підвищення плану
Обрані технології реалізації ідеї проєкту: Neovim, Next.js, Sanity, Clerk, PostHog, Stripe				

В результаті проведеного дослідження було визначено, що кожна з необхідних технологій є доступною, а отже реалізація ідеї проекту є можливою.

9.3 Аналіз ринкових можливостей запуску стартап-проекту

В наступному етапі було визначено метрики сфери і проаналізовано попит на послуги в ній задля встановлення чи є такий продукт рентабельним.

Спочатку було визначено обсяг попиту на ринку та ступінь розвитку сфери. Ці дані наведено у таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку

№ п/п	Показники стану ринку (найменування)	Характеристика
1	Кількість головних гравців, од	3 (розробник, адміністратор порталу, користувачі порталу)
2	Загальний обсяг продаж, грн/ум.од	6.3 трильйони доларів за 2023 рік
3	Динаміка ринку (якісна оцінка)	Зростає з кожним роком
4	Наявність обмежень для входу (вказати характер обмежень)	Необхідно створити аккаунти на використовуваних сервісах та мати капітал для закупівлі продукції
5	Специфічні вимоги до стандартизації та сертифікації	Відсутні
6	Середня норма рентабельності в галузі (або по ринку), %	40-70%

Далі, було визначено характеристики клієнтів, тобто потреби ринку, опис яких наведено у таблиці 9.5.

Таблиця 9.5 – Характеристика потенційних клієнтів

№ п/п	Потреба, що формує ринок	Цільова аудиторія (цільові сегменти ринку)	Відмінності у поведінці різних потенційних цільових груп клієнтів	Вимоги споживачів до товару
1	Низький поріг входу на ринок онлайн-продаж	Володарі ювелірних магазинів або підприємці бажаючі спробувати увійти на ринок	Відсутні	− стабільна робота сервісу; − інтуїтивний інтерфейс, який можна налаштовувати; − доступ до аналітики для розвитку продукту; − можливість внести каталог товарів; − індексованість ресурсу пошуковими системами.

Продовження таблиці 9.5

2	Відмова від підписок на сервіс, що не дає повноцінний доступ до коду	Володарі ювелірних магазинів або підприємці	Відсутні	−доступ до коду; − розширюваність продукту.
3	Зниження витрат на підтримку веб-ресурсу	Володарі ювелірних магазинів або підприємці	Відсутні	−можливість самостійно керувати продажами; − відсутність необхідності наймати команду для підтримки ресурсу.

За цим здійснено аналіз ринку на предмет факторів ризику та сприяння розвитку проекту (табл. 9.6-9.7).

Таблиця 9.6 – Фактори загроз

№ п/п	Фактор	Зміст загрози	Можлива реакція компанії
1	Реєстрація бізнесу	Будь-яку бізнес має бутин юридично задекларований в реєстрі.	Офіційно оформити стартап-проект як юридичну особу або діяти представництвом фізичної особи-підприємця

Продовження таблиці 9.6

2	Велика конкуренція	Конкуренція може виявитися занадто великою для того щоб навіть почати отримувати вагому кількість продажів	Маркетингова кампанія, покращення користувацького досвіду, програми лояльності, унікальність продукції.

Таблиця 9.7 – Фактори можливостей

№ п/п	Фактор	Зміст можливості	Можлива реакція компанії
1	Легкий вхід на ринок	За допомогою розробленої системи вхід на ринок є досить простим, бо вона надає всі необхідні для цього функції	Задоволення базових вимог буде достатнім для виходу на ринок
2	Висока налаштованість	Адміністратор, поєднавши свої знання в області бізнесу разом з налаштованістю системи зможе створити користувацькі історії, які можуть привабити більше користувачів	Налаштування системи під потреби користувачів, що принесе більше конвертувань
3	Покращення роботи штату	За необхідності найняти команду, кількість необхідних кадрів буде менше, отже їх ефективність роботи може зрости.	Найняття достатньої кількості персоналу, скорочення зайвих витрат, підвищення заробітних плат для заохочення професіоналів.

Результати проведеного аналізу показують, що основною загрозою є конкуренція на ринку. Для виграшу серед конкуренції було запропоновано

потенційні рішення для заохочення покупців. Варто зазначити, що можливості ринку все одно переважають ризики та свідчать про високий шанс успіху проекту.

Далі, аналіз пропозиції дозволив отримати результати, що подані у таблиці 9.8.

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

Особливості конкурентного середовища	В чому проявляється дана характеристика	Вплив на діяльність підприємства (можливі дії компанії, щоб бути конкурентоспроможною)
1. Тип конкуренції: монополістична	Ринок заповнений аналогами, що конкурують за об’єм продаж	Проведення рекламних компаній. Створення унікальних функцій. Розробка функціоналу або сторінок відповідно до потреб користувачів. Розумна навігація.
2. За рівнем конкурентної боротьби: світовий	Існуючі аналоги розробляються скрізь у світі	Додавання різних мов (i18n) в інтерфейс системи.
3. За галузевою ознакою - внутрішньогалузева	Конкуренція на ринку ювелірних товарів ведеться в певній єдиній галузі.	Зосередження зусиль на пошуку конкурентних переваг, рекламні кампанії, програми лояльності.
4. Конкуренція за видами товарів: товарно-видова	Конкурують товари одного виду.	Виготовлення унікальної або вірусної продукції. Знижені ціни. Краща якість продукції.

Продовження таблиці 9.8

5. За характером конкурентних переваг - цінова	Для значної частки користувачів ціна є визначальною при виборі.	При встановлені цін, враховувати уподобання ринку.
6. За інтенсивністю - марочна	Наявність унікальних ознак, що відрізнятиме продукт.	Розробка унікальних дизайнів та торг. марки.

Ступеневий аналіз конкуренції показав результати, що кажуть про те, що аналоги продукту на ринку відзначаються подібним функціоналом та мають аналогічні недоліки, причому їх відмінності виявляються малозначущими. Таким чином, введення нового продукту може спричинити залучення частини аудиторії від конкурентів.

Далі, здійснено детальний аналіз умов конкуренції, що сприятиме більш глибокому розумінню конкурентного оточення. Результати цього аналізу зведено у таблицю 9.9.

Таблиця 9.9 – Аналіз конкуренції в галузі за М.Портером

	Прямі конкуренти в галузі	Потенційні конкуренти	Постачальники	Клієнти	Товари-замінники
Складові аналізу	Shopify, Woo Commerce	Програмні рішення в сфері ювелірного бізнесу	SanityCMS, Clerk, Stripe, PostHog	Власники ювелірних бізнесів будь-яких розмірів	Існуючі рішення

Продовження таблиці 9.9

Висновки	Велика конкуренція, але інша цінова політика і інший функціонал.	Потенційних конкурентів небагато, тому що ринок вже і так заповнений.	Є залежність від третіх сервісів, проте було обрано найстабільніші рішення, якими користується майже весь веб-світ.	Є певна невідомість у рамках користувацької бази, проте це очікувано, тому що ринок переповнено.	Деякі клієнти вже обрали своїх фаворитів, тому переманити їх може бути складною задачою

З даного аналізу можна усвідомити, що деякі аналоги існують і знайти нових клієнтів може бути складним через адаптованість інших платформ, проте є можливість запропонувати кращі умови. Також, як було сказано раніше, в даній сфері попит дуже стрімко росте, тому завжди знайдеться бажаючий спробувати щось нове клієнт, якщо забезпечити якісне просування продукту.

Для того, щоб досягнути цього пропонується проаналізувати фактори конкурентноспроможності. Дані цього аналізу наведено у таблиці 9.10.

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

№ п/п	Фактор конкурентноспроможності	Обґрунтування (наведення чинників, що роблять фактор для порівняння конкурентних проектів значущим)
1	Зручність у використанні	Система має доцільний інтерфейс і очевидні для використання функції.

Продовження таблиці 9.10

2	Гнучкість у модифікації	Система дозволяє модифікувати зміст та створювати нові сторінки простим для звичайного користувача чином.
3	Швидкість	Веб-система використовує найновітніші технології, що дозволяють досягати вищих швидкостей та кращої індексованості.
4	Доступність	Веб-система доступно абсолютно кожному, бо адаптована під різні пристрої.
5	Здатність до масштабування бізнесу	Архітектура та функціонал розроблений таким чином, що можна з легкістю додавати нові сторінки та функціонал до системи.

Подивившись на дані цього аналізу, можна зробити висновок про те, що розроблювана система покриває найважливіші функції, якими володіють конкуренти, а також покращує деякі з них. Це каже про те, що продукт матиме досить сильну конкурентоспроможність.

У порівнянні з основними конкурентами було проаналізовано сильно та слабкі сторони стартап-проекту, що наведені у таблиці 9.11.

Таблиця 9.11 – Порівняльний аналіз сильних та слабких сторін стартап-проекту

№ п/п	Фактор конкурентоспроможності	Бали 1-20	Рейтинг товарів-конкурентів порівняно з проектом						
			-3	-2	-1	0	+1	+2	+3
1	Зручність у використанні	15		WooCommerce		Pandora		Spotify	

Продовження таблиці 9.11

2	Гнучкість у модифікації	15		Pandora			Spotify	WooCommerce	
3	Швидкість	18		Pandora		Spot Ify, Woo Commerce			
4	Доступність	16				Pandora, Spotify, Woo Commerce			
5	Здатність до масштабування бізнесу	14		Pandora			Spotify, Woo Commercd		

Проведений порівняльний аналіз показав, що проект має свої переваги та недоліки у порівнянні з основними конкурентами. Це свідчить про те, що для того, щоб виграти місце на ринку важливо буде розробити правильну стратегію.

Перед тим, як розробляти таку стратегію, пропонується також розглянути можливості впровадження проекту за допомогою SWOT-аналізе, що детально зобразить основні чотири складові продукту. Результати цього аналізу наведено у таблиці 9.12.

Таблиця 9.12 – SWOT-аналіз

Сильні сторони	Слабкі стороні
- автономність; - доступність для будь-кого; - можливість масштабування; - висока налаштованість; - аналітика, метрики; - доволі багатий функціонал; - зручність та простота використання.	- час на розробку; - недосконаліть інтерфейсу; - відсутність готових блоків для побудови сторінок; - Неповноцінність адмін-панелі, необхідність переходити на інші ресурси для перегляду деяких даних.
Можливості	Загрози
- удосконалення графічного інтерфейсу; - додавання нових функцій; - створення шаблонів для сторінок.	- відмова ринку, внаслідок поганих бізнес рішень; - відсутність вбудованої адмін-панелі; - поява на ринку сильного конкурента; - відсутність ресурсів на покращення системи; - можливий повний провал при виході на ринок, якщо не мати бізнес-план.

Використовуючи результати SWOT-аналізу було побудовано альтернативи ринкової поведінки для зайняття ринку. Вони наведені у таблиці 9.13.

Таблиця 9.13 – Альтернативи ринкового впровадження

№ п/п	Альтернатива (орієнтовний комплекс заходів) ринкової поведінки	Ймовірність отримання ресурсів	Строки реалізації
1	Підготувати передачу проекту під ключ без ніяких налаштувань клієнтом	Середня	3 місяці
2	Презентація проекту, його просування в Інтернет просторі.	Середня	3+ місяці
3	Реалізувати прев'ю, доступний для всіх ресурс, де можна встановити власну копію продукту без доступу до коду	Висока	1-2 місяці

Отже, беручи до уваги результати проведениого аналізу, привабливою альтернативою є надання потенційним користувачам можливості встановити свою власну копію проекту без доступу до коду, щоб протестувати функціонал.

9.4 Розроблення ринкової стратегії проекту

Для створення ринкової стратегії по-перше необхідно визначити цільові групи споживачів. Саме цим групам буде пропонуватися використання проекту. Після цього можна визначити базову стратегію розвитку стартап-проекту. Таблиця 9.14 відображає дані про такі групи споживачів.

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

№ п/п	Опис профілю цільової групи потенційних клієнтів	Готовність споживачів сприйняти продукт	Орієнтовний попит в межах цільової групи (сегменту)	Інтенсивність конкуренції в сегменті	Простота входу у сегмент
1	Власники ювелірних магазинів	Висока	Високий	Висока	Висока
2	Підприємці бажаючі спробувати увійти на ринок	Висока	Високий	Дуже висока	Висока

Як можна бачити, обидві виявлені групи споживачів можуть бути дуже зацікавлені використанням продукту. Далі було сформовано базову стратегію розвитку (табл. 9.15).

Таблиця 9.15 – Визначення базової стратегії розвитку

№ п/п	Обрана альтернатива розвитку проекту	Стратегія охоплення ринку	Ключові конкурентоспроможні позиції відповідно до обраної альтернативи	Базова стратегія розвитку
1	Надання простоти у роботі, але водночас високої масштабованості, що відсутня у товарів-замінників	Становлення фокусу на сильних сторонах додатку, а саме високої налаштованості, і розвиток у цьому напрямку з урахуванням використаних підходів.	Можливість реалізовувати різноманітні бізнес-сценарії у системі і, як наслідок, приваблення більшої кількості клієнтів.	Стратегія диференціації

Стратегія конкурентної поведінки була розроблена на основі визначеної стратегії розвитку, що наведена у таблиці 9.16. Вона є стратегією диференціації.

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

№ п/п	Чи є проект «першопрохідцем»	Чи буде компанія шукати нових споживачів, або забирати існуючих у конкурентів?	Чи буде компанія копіювати основні характеристики товару конкурента, і які?	Стратегія конкурентної поведінки
1	Ні, система є покращенням існуючих аналогів.	В основному, компанія хоче забирати існуючих користувачів, надаючи кращий досвід використання.	Так, деякий основний функціонал буде зкопійований: - стандартний функціонал онлайн-магазину; - зручність інтерфейсу; - система макетів для створення сторінок за бажанням.	Наслідування лідеру.

Наступним кроком було визначення стратегії позиціонування (табл. 9.17).

Таблиця 9.17 – Визначення стратегії позиціонування

№ п/п	Вимоги до товару цільової аудиторії	Базова стратегія розвитку	Ключові конкурентоспроможні позиції власного стартап-проекту	Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових)
1	Одноразовий платіж та помірна ціна	Зменшення вартості, відмова від підписок	Використання відкритих, доступних технологій	Дешевизна

Продовження таблиці 9.17

2	Можливість продовжити розробку власноруч	Створення проекту під ключ, повний доступ без обмежень	Доступ до повного коду програми	Повнота доступу
3	Початкове розширення без втручання спеціалістів	Автономність продукту дозволяє отримати перші результати навіть без участі команди	Виняткова розширюваність	Масштабованість

За результатами проведеного аналізу, було обрано такі стратегії ринкової поведінки: стратегія диференціації в якості базової стратегії, стратегія наслідування лідеру як конкурентна поведінка.

9.5 Розроблення маркетингової програми стартап-проекту

Останнім та одним з найважливіших етапів підготовки стартап-проекту для виходу на ринок є розробка стратегії маркетингу для нього.

Спочатку необхідно сформувати концепцію товару, що отримує споживач. Для цього було зведено результати деяких попередніх аналізів в одну логічну групу і визначено головні відмінності від концепцій конкурентів, що можуть стати вирішальними для вибору потенційного товару серед всіх інших.

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

№ п/п	Потреба	Вигода, яку пропонує товар	Ключові переваги перед конкурентами
1	Потреба в вищому рівні контролю наповнення	Майже кожна логічна частина системи може бути налаштована індивідуально під продукт	Правильно визначена структура для ведення ювелірного бізнесу, можливість створювати колекції, продукти, фільтри та ін.
2	Потреба у можливості самостійно розширювати систему	Адміністративвна панель з вичерпним функціоналом.	Налаштовані стандартні параметри під ювелірний магазин
3	Потреба в контролі бізнесу	Система використовує сервіси, що дозволяють контролювати все, що відбувається в додатку.	Вбудована аналітика, маркетингові розсилки, ведення обліку та спостерігання за користувачами
4	Реалізація власних бізнес-ідей для маркетингу у вигляді окремих сторінок та користувацьких історій	Система надає можливість створювати різноманітні шляхи для користувача в залежності від потреб бізнесу.	Вищий рівень контролю над наповненням сторінок.

Даний продукт є новим на ринку і слідує стратегічній ідеї забирання клієнтів серед всіх способів отримання споживачів. Це є як недоліком, бо робить складніше весь процес, так і перевагою, бо в контрасті з іншими продуктами на ринку, він робить акцент на покращенні існуючого функціоналу аналогів. В наступному етапі

було розроблено три рівневу маркетингову модель продукту, що наведено у таблиці 9.19 та було наведено цінові межі на продукт, що зображено у таблиці 9.20.

Таблиця 9.19 – Опис трьох рівнів моделі товару

Рівні товару	Сутність та складові		
I. Товар за задумом	Автономна веб-система з можливостями розширення та побудови спеціальних користувацьких шляхів з використанням Next.js та Sanity CMS.		
II. Товар у реальному виконанні	Властивості/характеристики	М/Нм	Вр/Тх /Тл/Е/Ор
	1. Зменшення необхідної робочої сили на підтримку	Нм	Вр/Тл/Е
	2. Швидке створення прототипів та розширення	Нм	Тх/Тл/Е
	3. Зручність у використанні	Нм	Тх/Е/Ор
	4. Збір даних про систему, доступ до них у використованих сервісах	Нм	Тх/Тл/Е
	5. Доступність та швидкість подачі контенту	Нм	Тл/Е/Ор
	6. Комплексне рішення	М	Вр/Тл
	Якість: дотримання архітектурних стандартів, модульність коду, опрацювання всіх негативних та позитивних шляхів його виконання		
	Повний доступ до вихідного коду		
	Марка: Jewellery.dev Ltd. – Flexible Jewellery Store. Торгівельна марка самого продукту визначається клієнтом.		
III. Товар із підкріпленням	До продажу: наявна документація, формат придбання без ліцензії, повний доступ до коду		
	Після продажу: підтримка команди бізнесу, додаткова розробка за потреби		
За рахунок чого потенційний товар буде захищено від копіювання: реєстрація права на інтелектуальну власність			

Таблиця 9.20 – Визначення меж встановлення ціни

№ п/п	Рівень цін на товари-замінники	Рівень цін на товари-аналоги	Рівень доходів цільової групи споживачів	Верхня та нижня межі встановлення ціни на товар/послугу
1	8 000 грн, одноразовий платіж	Від 2500 грн / місяць	>20 000 грн у перший місяць і подальший розвиток	8 000 грн – 14 000 грн одноразово

Наступним кроком було визначено оптимальну систему збуту, що наведено у таблиці 9.21.

Таблиця 9.21 – Формування системи збуту

№ п/п	Специфіка закупівельної поведінки цільових клієнтів	Функції збуту, які має виконувати постачальник товару	Глибина каналу збуту	Оптимальна система збуту
1	Придбання вихідного коду програми	Розгортання проекту, підключення всіх необхідних сервісів. Підтримка впродовж користування терміном 2 тижні, після чого необхідна додаткова підписка. Впровадження нових функцій за додаткову оплату.	Глибока	Організація маркетингових кампаній, пошук клієнтів серед реальних фізичних підприємств.

Було вирішено самостійно здійснити маркетинг та просувати продукт через різноманітні інформаційні канали, включаючи розміщення рекламних оголошень.

Останнім етапом було розроблено концепцію маркетингових комунікацій, враховуючи обрану стратегію позиціонування та унікальність клієнтської поведінки. Нова концепція спрямована на інформування та напоминання споживачам про продукт, а також на підтримку його успішного впровадження (табл. 9.22).

Таблиця 9.22 – Концепція маркетингових комунікацій

№ п/п	Специфіка поведінки цільових клієнтів	Канали комунікацій, якими користуютьс я цільові клієнти	Ключові позиції, обрані для позиціонуванн я	Завдання рекламного повідомленн я	Концепція рекламного звернення
1	Бажання розвинути бізнес в інтернет просторі, контролюв ати його власноруч та зменшити витрати	Пошук в Інтернеті та локально на місці	Широкий функціонал, унікальні можливості (висока модифікованіс ть), відсутність підписок як моделі платежу	Висвітлити переваги використанн я системи, її масштабован ість, корисність для обраної сфери бізнесу та привабливу модель оплати	Опис переваг та акцент на унікальних функціях, доступності системи, зручності у використанні та легкості процесу придбання

В останньому розділі було здійснено аналіз запуску на ринок розробленої інформаційної системи як стартап-проекту. За результатами було виявлено, що в проекту є можливість комерціалізації, оскільки попит зростає з кожним роком, ринок великий та користується широким використанням та інтересом, а також вхід на ринок для клієнтів продукту не має бути складним, тому пошук споживачів є досить

можливим. Було визначено потенційні групи клієнтів, конкурентоспроможність та наявність перспектив впровадження. Найпростішим та найпривабливішим для користувачів способом розповсюдження є надання безкоштовного пробаного періоду. Можлива подальша розробка продукту для забезпечення ще більшої автономності та покриття більшої кількості потреб споживачів.

ВИСНОВКИ

В результаті виконання даної роботи була розроблена веб-система для ведення ювелірного бізнесу.

Спочатку, було проведено аналіз предметної області системи для отримання визначення загального уявлення про можливі вимоги та можливі сценарії застосування.

Після цього кроку, було здійснено порівняльний аналіз існуючих рішень у даній сфері. Було виокремлено основні переваги та недоліки конкурентів. За допомогою цього аналізу, на його основі було визначено перелік основних можливостей системи, що будуть необхідні для її корисності.

У наступному, другому, розділі, базуючись на результатах виконаного аналізу, було сформовано функціональні та нефункціональні вимоги до системи, звідки з'явилася можливість побудувати основні варіанти використання системи.

У розділі 3 було описано можливі сценарії використання системи. В системі, що була розроблена, існує всього дві основні ролі. Тож для них було описано основні шляхи всередині додатку. За допомогою описаних раніше вимог, було зазначено основні функції і методи взаємодії з ними для виявлених груп користувачів. Тож в цьому розділі було встановлено яким функціоналом має бути наділена система.

Беручи за основу побудовану діаграму з розділу 3, було проаналізовано можливі варіанти побудови архітектури системи. Було проведено аналіз існуючих архітектурних підходів, а також визначено технології, які будуть найбільш доцільними для досягнення такої архітектури. Через те, що додаток мав бути гнучким та налаштовуваним за своєю природою, було обрано для розробки відповідну систему керування змістом, обрано технології для взаємодії з нею, а також детально описані причини за якими було обрано ту чи іншу екосистему та сторонній сервіс. Результатом цього розділу став виявлений найоптимальніший варіант технологій для втілення цілей проекту в життя.

Слід за описом використовуваних технологій було описано загальну структурну схему системи. Така схема зображує зв'язки між сутностями в системі, описує їх форму

та дає загальний погляд на те, як виглядатиме набір даних, з яким працює розроблювана система.

Після опису архітектурних рішень та вигляду даних всередині системи, було розроблено озеро контенту. Для цього спочатку було виконано проектування даних, визначено основні сутності системи та їх атрибути. На тому ж самому етапі було здійснено опис схеми озера контенту з визначеними типами атрибутів сутностей та їх обмеженнями.

Коли всі особливості архітектури та даних були нарешті визначені, наступним кроком стала розробка користувацького інтерфейсу для системи. Так як адміністратор в системі був привілейованою роллю, для нього було створено окремі сторінки, де він міг би керувати налаштуваннями та спеціальними функціями системи. Клієнтський інтерфейс адаптований для будь-яких пристроїв, що робить його доступним для більшості користувачів. Він був розроблений з використанням останніх технологій веб-екосистеми задля використання новітніх підходів, що забезпечують швидкість роботи системи та швидкість її розробки. Інтерфейс створено так, щоб користувачеві було дуже просто дістатися необхідної частини системи без додаткових зусиль. Загалом, зовнішній вигляд додатку націлений на демонстрацію максимальної кількості товарів та визначених адміністратором ключових елементів. Такий підхід підвищує шанс конвертації клієнтів та робить використання сайту найбільш ефективним.

В останній частині було здійснено програмну реалізацію системи, пояснено як взаємодіють між собою клієнтська та серверна частини, проаналізовано якість поєднаних підходів та технологій.

По закінченню технічної розробки було проведено аналіз можливості реалізації даного продукту у вигляді стартап-проекту. Дослідження показало, що хоч система і має інноваційний характер та впроваджує свого роду унікальні підходи, буде досить важко обійти конкуренцію звичними шляхами. Тому було прийнято до уваги такі фактори як ціна, масштабованість та легкість у використанні та створенні команди навколо системи. У висновку з цього розділу можна зазначити, що зайняття позиції на ринку схожих рішень буде важким, проте можливим, і така система має місце в своїй ніші.

Результатом роботи є веб-система, що повноцінно задовольняє визначеним вимогам.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Онлайн платформа для продажів Shopify. URL: https://www.shopify.com/.
2. Платформа Wix. URL: https://wix.com/.
3. Pandora. URL: https://us.pandora.net/.
4. Ціни на теми Shopify. URL: https://themes.shopify.com/themes?price%5B%5D=paid&sort_by=most_relevant.
5. Безпекові ризики WordPress. URL: https://jetpack.com/blog/wordpress-security-issues-and-vulnerabilities/.
6. WooCommerce. URL: https://woocommerce.com/.
7. Типи вимог до продукту. URL: https://visuresolutions.com/uk/%D0%BF%D0%BE%D1%81%D1%96%D0%B1%D0%BD%D0%B8%D0%BA-%D0%B7-%D0%B2%D1%96%D0%B4%D1%81%D1%82%D0%B5%D0%B6%D0%B5%D0%BD%D0%BD%D1%8F-%D1%83%D0%BF%D1%80%D0%B0%D0%B2%D0%BB%D1%96%D0%BD%D0%BD%D1%8F-%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B0%D0%BC%D0%B8/%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D1%82%D0%B0-%D0%BD%D0%B5%D1%84%D1%83%D0%BD%D0%BA%D1%86%D1%96%D0%BE%D0%BD%D0%B0%D0%BB%D1%8C%D0%BD%D1%96-%D0%B2%D0%B8%D0%BC%D0%BE%D0%B3%D0%B8/.
8. Next.js. URL: https://nextjs.org/.
9. SvelteKit. URL: https://kit.svelte.dev/.
10. Nuxt.js. URL: https://nuxt.com/.
11. SPA vs MPA. URL: https://kadiska.com/what-is-a-single-page-application-spa/.

12. The process of hydration in JavaScript web-applications. URL: https://github.com/vercel/next.js/discussions/22276#discussioncomment-7195587.

13. Server Side Rendering. URL: https://nextjs.org/docs/pages/building-your-application/rendering/server-side-rendering.

14. Static Site Generation. URL: https://nextjs.org/docs/pages/building-your-application/rendering/static-site-generation.

15. Структурна схема. URL: https://www.wiki.ukua.nina.az/%D0%A1%D1%82%D1%80%D1%83%D0%BA%D1%82%D1%83%D1%80%D0%BD%D0%B0_%D1%81%D1%85%D0%B5%D0%BC%D0%B0.html.

16. Sanity cache revalidation. URL: https://www.sanity.io/plugins/next-sanity.

17. Stripe. URL: https://stripe.com/en-gb.

18. Next.js. Fetching data on the server. URL: https://nextjs.org/docs/app/building-your-application/data-fetching/patterns#fetching-data-on-the-server.

19. Next.js Server actions. URL: https://nextjs.org/docs/app/building-your-application/data-fetching/server-actions-and-mutations

20.

Додаток І

Вихідний код системи

Посилання на репозиторій

GitHub: https://github.com/r4zendev/university-project