Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Клієнт-серверний сервіс
онлайн-торгівлі ювелірними виробами
на основі React та Node.js
(тема)

Виконав:
здобувач 4 року навчання,
групи КІУКІ-21-6
Тетяна ТКАЧЕНКО
(власне ім’я, прізвище)

Спеціальність
123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма
Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: ас. Ігор МИХАЙЛІЧЕНКО.
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Ткаченко Тетяні Андріївні
(прізвище, ім’я, по батькові)

1. Тема роботи Клієнт-серверний сервіс онлайн-торгівлі ювелірними виробами на основі React та Node.js

затверджена наказом по університету від “ 26 ” травня 2025 р. № 424 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 17 червня 2025 р.

3. Вхідні дані до роботи

1) Документація React та Redux
2) Документація HTML, CSS
3) Документація до пакетного менеджеру NPM
4) Документація Node.js та Express
5) Документація Nodemailer
6) Середовище розробки Figma
7) Середовище розробки Visual Studio Code

4. Перелік питань, що потрібно опрацювати у роботі

1) аналіз предметної області
2) аналіз використаних технологій
3) програмна реалізація
4) висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 19 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз актуальності та огляд існуючих рішень	27.05.25-28.05.25	
2	Вибір технологій, бібліотек, архітектур та	29.05.25-30.05.25	
	Інструментів розробки		
3	Розробка алгоритмів	31.05.25-01.06.25	
4	Розробка серверної частини системи	02.06.25-04.06.25	
5	Розробка клієнтської частини системи	05.06.25-07.06.25	
6	Відлагодження комп'ютерної системи	08.06.25-09.06.25	
7	Оформлення матеріалів кваліфікаційної роботи	10.06.25-11.06.25	
8	Подання кваліфікаційної роботи керівникові	12.06.25-13.06.25	
	та її попередній захист		
9	Подання кваліфікаційної роботи на рецензу-	14.06.25-16.06.25	
	вання		

Дата видачі завдання " 26 " травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Ігор МИХАЙЛІЧЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 91 с., 41 рис., 1 табл., 1 дод., 15 джерел.

UI/UX, ОНЛАЙН-МАГАЗИН, КЛІЄНСЬКА ЧАСТИНА, СЕРВЕРНА ЧАСТИНА, ФРЕЙМВОРК, ФІЛЬТРАЦІЯ, СОРТУВАННЯ, РОУТИНГ, ПАГІНАЦІЯ.

Метою кваліфікаційної роботи є розробка сучасного та конкурентоспроможного клієнт-серверного сервісу для продажу ювелірних виробів, який матиме зручний функціонал та зрозумілий для користувача інтерфейс. Основними особливостями є спеціально розроблений стиль бренду, а також додавання унікального функціоналу, зокрема вибір пакування замовлення, вибір валюти тощо. За допомогою застосунку можна швидко робити покупки, легко здійснювати пошук товарів за окремими фільтрами або через рядок пошуку, сортувати товари, додавати товари, що сподобалися до «Улюбленого», а також додавати товари, які хоче придбати у «Кошик».

У ході виконання кваліфікаційної роботи було проаналізовано логічність розташування елементів у наявних вебдодатках. Для аналізу та порівняння було обрано декілька онлайн-магазинів, що допомогло виявити UI/UX рішення, які покращують користувацький досвід. Також було здійснено аналіз стеку технологій та їх можливостей для веброзробки. До уваги були взяті такі аспекти, як продуктивність, масштабованість тощо. Вибір технологій, включаючи React для front-end частини та Express.js для back-end, було обумовлено можливістю створення гнучкої архітектури.

Результатом роботи стала клієнт-серверна система для онлайн-магазину ювелірних виробів, який поєднав у собі усі необхідні можливості та функції, що відповідають усім поставленним цілям та вимогам.

ABSTRACT

Bachelor’s thesis: 91 pages, 41 figures, 1 tables, 1 appendices, 15 sources.

UI/UX, ONLINE STORE, CLIENT SIDE, SERVER SIDE, FRAMEWORK, FILTERING, SORTING, ROUTING, PAGINATION.

The purpose of the thesis is to develop a modern and competitive client-server service for selling jewelry, which will have user-friendly functionality and a clear and intuitive user interface. The main features are a specially designed brand style, and the addition of unique functionality, including the choice of order packaging, currency selection, etc. With the website, users can quickly shop, easily search for products by individual filters or through the search bar, sort products, add favorite products to their “Favorites”, and add products they want to buy to their “Cart”.

In the course of the thesis, the logical arrangement of elements in existing web applications was analyzed. Several online stores were selected for analysis and comparison, which helped to identify UI/UX solutions that improve the user experience. Also, the technology stack and their capabilities for web development were analyzed. Such aspects as performance, scalability, etc. were taken into account. The choice of technologies, including React for the front-end and Express.js for the back-end, was based on the ability to create a flexible architecture.

The result of the work was a client-server system for an online jewelry store that combined all the necessary features and functions that meet all the goals and requirements.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

UI – User Interface

UX – User Experience

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

JSON – JavaScript Object Notation

DOM API – Document Object Model Application Programming Interface

SPA – Single-Page Application

NPM – Node Package Manager

CLI – Command Line Interface

REST API – Representional State Transfer Application Programming Interface

MVCC – Multi-Version Concurrency Control

ACID – набір принципів, які забезпечують надійність транзакцій у базах даних та гарантують цілісність, узгодженість, ізоляцію та довговічність.

GiST – Generalized Search Tree

GIN – Generalized Inverted Index

BRIN – Block Range Index

Рендеринг – це процес виведення або оновлення вмісту на ерані користувача.

Sitemap – схематичне представлення структури вебзастосунку.

User Flow – схема, яка зображує шлях виконання цільової дії користувача.

MVC – Model-View-Controller

CRUD (Create, Read, Update, Delete) – основні операції для управління даними.

SMTP – Simple Mail Transfer Protocol

ВСТУП

У світі іноваційних технологій інтернет-магазини грають досить важливу та вагому роль, а саме у комерції та продажі різноманітних товарів. Саме вони сприяють оптимізації процесу купівлі, забезпечиючи його зручність та швидкість для користувачів.

Здійснення онлайн-покупок стало набирати популярність серед покупців завдяки простоті взаємодії з інтернет-ресурсами та мінімізації часових витрат користувачів. Внаслідок автоматизації обробки замовлень та інтеграції платіжних систем зникла потреба фізичної присутності покупців.

Під час розвитку цього напрямку було важливо, щоб функціонал інтернет-магазинів був зрозумілим, простим та привабливим. З часом вони стали відповідати стандартам сучасного UI/UX дизайну та почали забезпечувати швидку та якісну взаємодію між продавцем та покупцем. Проте все ще залишаються такі рішення, які не є максимально ефективними та легкими у розумінні та користуванні, вони досі потребують вдосконалення. Такі сайти можуть мати різні проблеми: відсутність інтуїтивно зрозумілого інтерфейсу, нестача простих, проте швидких функцій тощо.

Метою даної роботи є створення клієнт-серверного сервісу з продажу ювелірних виробів, який зможе поєднати у собі візуально приємний та зрозумілий інтерфейс із оптимізованою програмною архітектурою. Особливу увагу було приділено розробці функціоналу, який гарантуватиме легкість використання та високу продуктивність системи.

Проєкт охоплює повний процес розробки інтернет-магазину, починаючи з аналізу конкурентів та завершуючи повною розробкою та інтеграцією ключових та розповсюджених функцій у вебсайт. Це має на меті забезпечення ефективного та безперебійного процесу здійснення покупок.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Тенденція розвитку електронної комерції

Наразі електронна комерція з кожним днем достатньо стрімко зростає у багатьох країнах світу. Це показує те, наскільки сучасні технології залучені у повсякденне життя користувачів.

Загалом відсоток використання інтернету для онлайн покупок стрімко збільшується. Про це свідчить одне з досліджень, що було здійснене статистичною службою Європейського Союзу Eurostat. Даний аналіз показав, що серед опитаних у 2024 році людей віком від 16 до 74 років аж 94% користувалися інтернетом за останні 12 місяців [1]. При цьому 77% з них користувалися ним для того, щоб купити та замовити товари чи послуги онлайн. Такі високі показники свідчать про те, що у сучасному світі для інтернету та онлайн шопінгу є дуже багато місця у нашому повсякденні.

Особливо популяризовано це серед молоді, бо саме вони переважно взаємодіють із гаджетами та роблять основний вклад у електронну торгівлю та формування статистичних даних. З кожним роком цей відсоток зростає і кількість залучених у це людей розширюється.

В умовах розвитку подій та зростаючої тенденції, питання інтеграції різних компаній та посилення цифрової присутності набуває більшого значення та впливу. Попит на адаптацію онлайн-платформ під сучасні тренди, смаки та тенденції підвищується, а значить постає питання у підлаштуванні платформ, враховуючи уподобання усіх вікових груп, забезпеченні зручного та зрозумілого інтерфейсу не лише для молоді, а й для людей більш старшого віку, наданні різноманітного асортименту товарів та послуг. Саме такі дії зможуть призвести до залучення ще більшої кількості користувачів до онлайн покупок та підвищення довіри до цифрових платформ.

1.2 Вимоги до сучасних інтернет-магазинів

Сучасний інтернет-магазин повинен відповідати певній низці вимог для того, аби забезпечити приємний та ефективний досвід користування для покупців. Основним та ключовим аспектом є інтуїтивно зрозумілий та простий дизайн, який має містити загальновживані функції такі як швидкий пошук, фільтрація та сортування товарів, чітка структура, легка навігація для прискореного знаходження необхідних товарів та можливість легкого оформлення замовлення у пару кліків.

Необхідним елементом успішного сайту є адаптивний дизайн та кросбраузерність. Він має чітко та безпомилково відображаится на різних пристроях та платформах. Дизайн повинен виглядати однаково влучно і привабливо на усіх типах гаджетів, включаючи смартфони, комп’ютери, планшети, адже саме таким обладнанням часто користується основна частина вірогідних покупців. Сумісність з веббраузерами також є не менш важливою, адже серед популярних вебресурсів є декілька кандидатів.

Для утримання уваги клієнтів треба запровадити швидку роботу інтернет-магазину, що є одним з ключових моментів розробки. Зменшенню часу завантаження сторінки може сприяти мінімізація запитів до сервера.

Різноманіття асортименту сприяє тривалішому перегляду товарів, оскільки користувачі витрачають більше часу на пошуки оптимального варіанта. Швидше орієнтуватися у великій кількості товарів та не розгубитися допоможе зручний пошук, фільтрація та сортування. Багато компаній при налаштуванні пошуку на сайті використовують технологію автозаповнення, надання підказок у вигляді товарів, які у назві мають частину введеного користувачем промту. Також популярним серед користувачів є використання спеціальних фільтрів за певними параметрами або сортування за критеріями популярності, новизни тощо.

Окрім цього люди часто звертають увагу на наявність інформації про товар, фото, відгуки та інше. Це дає змогу більше дізнатися про продукт, краще

зрозуміти чи є він тим, що шукали і швидше прийняти рішення щодо його замовлення. Чим більше відгуків, якісних фото чи відео речі є на сайті, тим вищим є рівень довіри клієнта до інтернет-магазину.

Обов’язкова умова для онлайн-торгівлі є безпечні методи оплати. Сучасний вебмагазин повинен підтримувати усі популярні платіжні системи, наприклад, PayPal, Google Pay, Apple Pay, а також забезпечити для користувачів безпеку їхніх транзакцій.

Особитий акаунт користувача, де він зможе знайти усю інформацію про збережені товари у «Улюбленому» чи «Корзині», про стан замовлення, історію власних покупок тощо.

Для того, щоб сформувати позитивне враження користувачів, необхідно, щоб процес пошуку, вибору та замовлення товару здійснювався в якомога меншу кількість кроків. Це забезпечить гарний досвід користування та залучить клієнта повернутися до інтернет-магазину для здійснення повторної покупки.

Останнім з ключових аспектів є захист даних та високий рівень безпеки. Використовуючи HTTP і SSL для впровадження безпечного з’єднання та захищаючи персональні дані кожного користувача можна покращити та підвищити рівень довіри клієнтів та закріпити за собою статус надійного інтернет-магазину.

Ці аспекти є показниками того, що володарям таких інтернет-магазинів важливий комфорт кожного і вони є конкурентоспроможніми та успішними.

1.3 Технологічні аспекти розробки онлайн-магазину

Процес створення та розробки онлайн-магазину є дуже об’ємним та містким. На даному етапі треба детально розробити структуру сайту та визначити основний функціонал майбутнього магазину. Виходячи з цих вимог буде залежати вибір стеку технологій, який буде використано при розробці клієнтської та серверної частин. Зручність використання та продуктивність

сервісу насамперед залежать від обраних технологій та інструментів.

Одним із перших кроків є вибір стеку технологій. Він визначатиме, як саме буде працювати майбутній сайт і які можливості буде мати. Зазвичай, він містить клієнстську частину або ж front-end, серверну частину або ж back-end та базу даних.

Перед тим, як почати розробляти застосунок, необхідно створити прототип сайту та його дизайн. Дана візуалізація допоможе наглядно зрозуміти структуру усіх сторінок, розташування елементів на сайті і їх стани під час взаємодії. Для створення макету сайту здебільшого використовують Figma або Adobe XD. Необхідно контролювати щоб дизайн сайту був чітким, інтуїтивно зрозумілим, естетичним і при цьому потрібно, щоб він відповідав принципам UX/UI-дизайну. Сайт має візуально приваблювати користувачів і залишати гарне враження від використання.

Front-end відповідає за те, що відображається користувачу і створюється за допомогою використання HTML, CSS та JavaScript. Також при розробці активно використовують популярні фреймворки та різноманітні бібліотеки, які значно спрощують розробку клієнської частини. Вони допомагають спростити та пришвидшити процес створення сторінок, а також забезпечують кращу взаємодію сайту з користувачем. Серед них широко використовується React.js, Vue.js та Angular.

На стороні back-end частини реалізується загальна логіка роботи застосунку, відбувається обробка запитів, а також здійснюється зберігання даних. Для реалізації серверної частини в роботі використовують різні мови програмування та фреймворки для покращення і полегшення створення сайту. Цей вибір дуже залежить від потреб та вимог конкретного проєкту. Тож слід враховувати можливості та переваги кожного можливого варіанту. Поширеними серед них є Node.js, Python, PHP тощо.

Також важливою складовою є база даних. Саме вона забезпечує збереження та отримання даних, містить в собі інформацію про наші товари, користувачів, доставку, оплату тощо. Важливо правильно обрати систему

керування базами даних, адже від цього напряму залежить безпека та масштабованість інтернет-магазину. Серед наявних реляційних баз даних популярними є PostgreSQL та MySQL, а серед NoSQL-баз даних поширеними є MongoDB і Firebase Firestore.

Кожен із наведених технологічних аспектів має досить високий вплив на процес створення застосунку. Якщо зробити правильний вибір необхідних технологій та інструментів і розумно впровадити їх у проєкт, то можна створити ефективний та гарний інтернет-магазин, який буде виграшно виглядати на сучасному ринку і буде користуватися великим попитом серед клієнтів.

1.4 Огляд наявних рішень

Для того, щоб чітко зрозуміти і виділити основні завдання майбутнього онлайн-магазину ювелірних виробів, було проаналізовано деякі з існуючих українських вебсайтів, що спеціалізуються на продажу прикрас. Є певні критерії, на які слід звертати увагу, для того, аби виокремити аспекти, які необхідно реалізувати у проєкті. Метою є огляд, порівняння та виявлення переваг та недоліків, які слід врахувати.

1.4.1 Kochut

Kochut Jewellery – це український онлайн-магазин, який спеціалізується на авторських прикрасах русної роботи, що виготовлені за унікальними дизайнами.

Дана платформа має гарний та сучасний дизайн, певний стиль розташування елементів, який додає преміальності і якого дотримано на усіх сторінках. (рисунок 1.1). Присутні акценти у вигляді великих фото виробів. Проте відсутня будь-яка динаміка через брак анімацій.

Рисунок 1.1 – Головна сторінка онлайн-магазину «Kochut»

Застосунок має легку та зрозумілу навігацію для користувача.

Є можливість обрати певну валюту, товари можна сортувати за популярністю, новизною, алфавітом, ціною та знижками.

Користувач може обрати певні фільтри, щоб переглянути конкретні товари, але елементи фільтруються не динамічно. (рисунок 1.2). Клієнту потрібно завжди шукати маленьку кнопку «Показати вироби» та натискати її та чекати на перезавантаження сторінки.

Також мінусом є відсутність динамічного пошуку, який міг би значно спростити та покращити використання сайту.

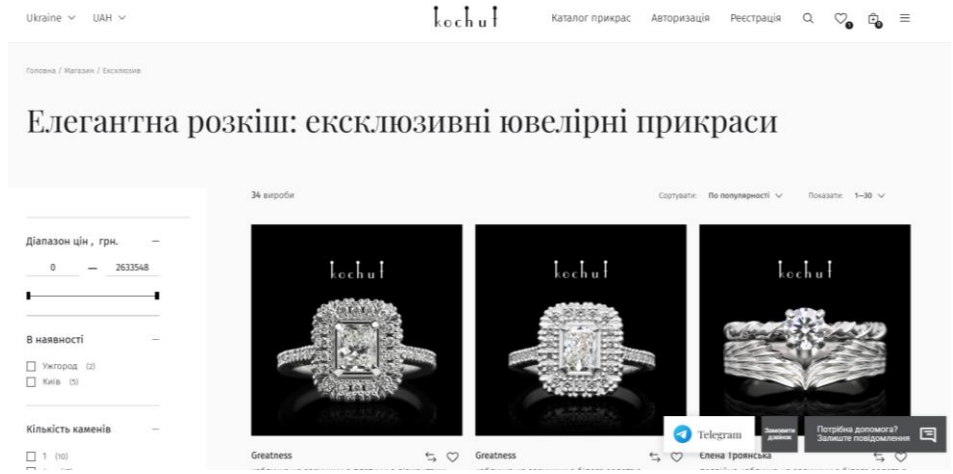

Рисунок 1.2 – Сторінка каталогу онлайн-магазину «Kochut»

Сайт має розділ «Улюблене», який дає можливість клієнту зберігати

товари, що сподобалися, але немає можливості напряму з «Улюбленого» додати цей товар у «Кошик». Це є недоліком, адже людині необхідно ще раз заходити на сторінку товару і робити дію повторно.

Плюсом також є адаптивний дизайн, який дає змогу демонструвати сторінки сайту у гарному вигляді на всіх девайсах.

Також погляд утримують дійсно якісні фото кожного виробу. На кожному фото є логотип магазину, що справляє враження надійності та професіональності.

Каталог має досить велике різноманіття прикрас з різного матеріалу, різною обробкою поверхні, покриттям, широким вибором коштовного каміння. Але при цьому час завантаження сайту є досить великим, що значно знижує швидкість пошуку.

1.4.2 MONO Jewerly

MONO Jewerly є українським вебзастосунком з продажу ювелірних прикрас та аксесуарів. Вони спеціалізуються на широкому асортименті виробів.

Вебсайт MONO Jewerly має доволі мінімалістичний, приємний та інтуїтивно зрозумілий дизайн. Усе оформлено у світлих тонах та досить ніжному стилі, який підкреслює особливість кожної з прикрас. (рисунок 1.3). На перших сторінках можна дізнатися інформацію про наявні знижки у магазині.

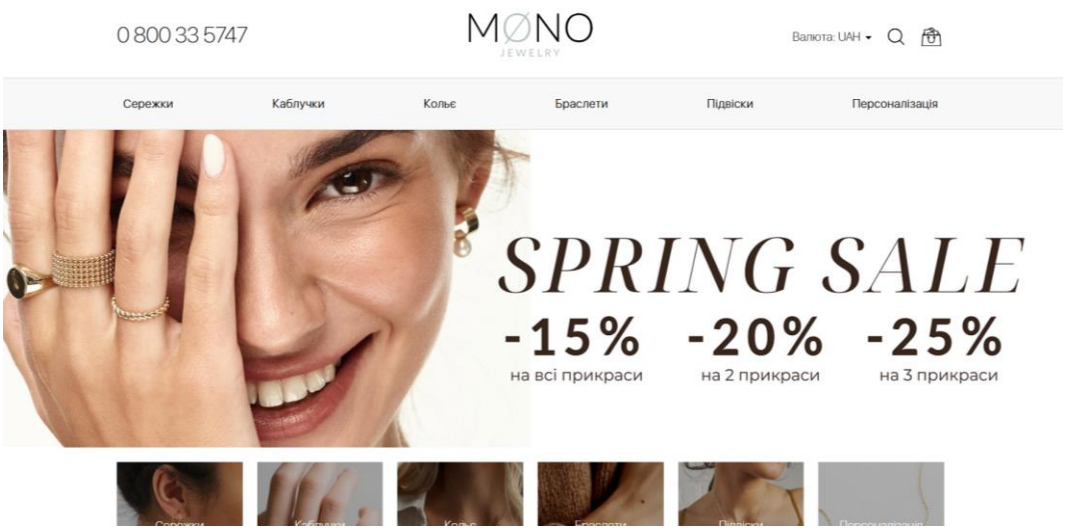

Рисунок 1.3 – Головна сторінка онлайн-магазину «MONO Jewerly»

Зручним аспектом вебверсії є наявність адаптивного дизайну, що дозволяє комфортно взаємодіяти з сайтом незалежно від типу девайса.

Слід зазначити відсутність анімацій, які допомогли б зробити онлайн-магазин виразним і спонукали клієнтів знову повертатися до перегляду сторінок, що затримують увагу.

Каталог має динамічний пошук, що дійсно є перевагою. Він швидко показує клієнту прикраси, які були знайдені на сайті за його введеним рядком. Загалом сторінка каталогу є зрозумілою та комфортною для використання. (рисунок 1.4).

Присутня можливість сортування товарів за новизною та ціною.

Також наявна фільтрація товарів за окремими критеріями, але при цьому вона не є динамічною і необхідно чекати на перезавантаження усієї сторінки.

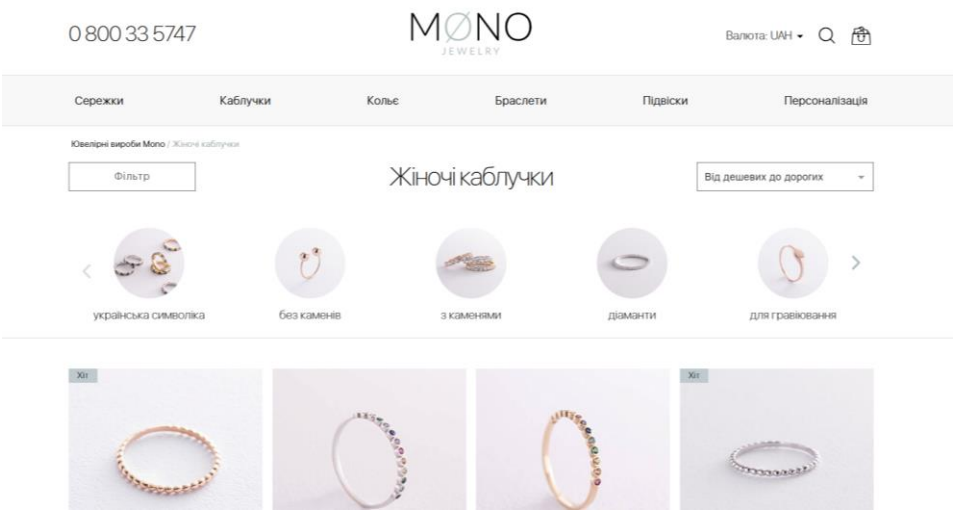

Рисунок 1.4 – Сторінка каталогу онлайн-магазину «MONO Jewerly»

Незручністю є відсутність розділу «Улюблене», до якого користувач міг би додати прикрасу, яка сподобалася, але яку він ще не готовий придбати. Такий розділ є важливим для магазину прикрас, адже такі покупки є дорогими і багатьом їх слід спочатку обдумати.

Практичним є можливість зміни валюти, адже це додає сайту професійності та демонструє орієнтацію магазину на міжнародну аудиторію.

Слід також зазначити фото прикрас. Вони зроблені якісно, з гарним освітленням, у однаковому стилі.

1.4.3 Світ прикрас

Світ прикрас – це ювелірний інтернет-магазин, який спеціалізується на продажі коштовностей. Наявний доволі великий асортимент виробів із різного металу та з використанням коштовного каміння.

Дизайн сайту магазину «Світ прикрас» потребує оновлення, адже використовуються застарілі підходи, що викликають у клієнтів незручності під час перегляду сторінок. (рисунок 1.5). Навігація не є інтуїтивно зрозумілою і не викликає гарних почуттів після використання.

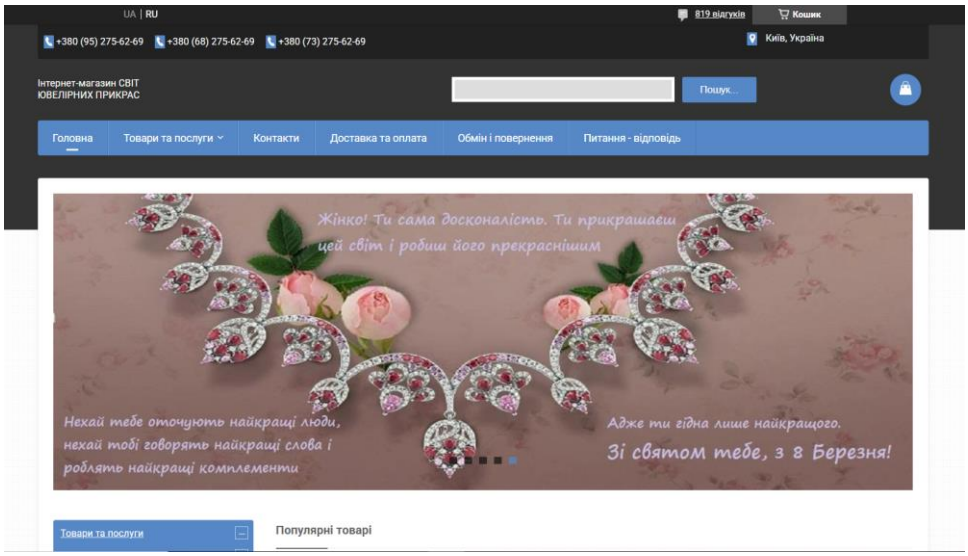

Рисунок 1.5 – Головна сторінка онлайн-магазину «Світ прикрас»

На сторінках каталогу немає єдиного концепту, стилю, кольорів та шрифтів. Було використано забагато акцентних та яскравих відтінків, що розсіюють увагу користувача і не закликають до здійснення покупки виробів. (рисунок 1.6).

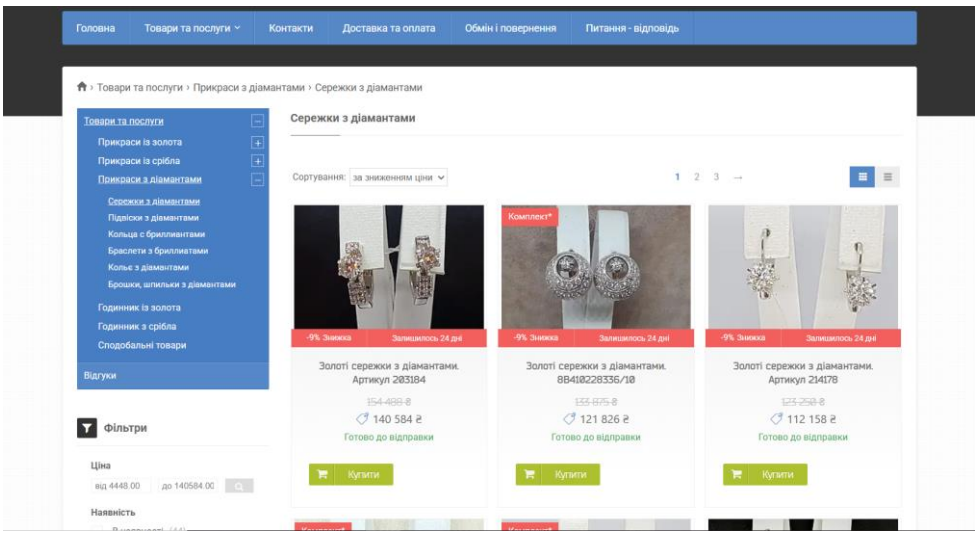

Рисунок 1.6 – Сторінка каталогу онлайн-магазину «Світ прикрас»

Відсутній також і адаптивний дизайн, що ускладнює перебування на сайті при використанні будь-якого пристрою з малим екраном.

Так само немає і будь-яких анімацій. Навіть при наведенні миші на кнопку «Купити» не відбувається ніяких змін.

Плюсом є наявність динамічного пошуку, що значно полегшує та прискорює знаходження певного ювелірного виробу або категорії.

Онлайн-магазин має можливість фільтрувати товари за окремими критеріями та фільтрами. Перевагою є динамічний пошук, тобто при виборі кожного з фільтрів оновлюється список прикрас, що підходять до обраних фільтрів. Проте через постійне перезавантаження сторінки фільтрація займає дуже багато часу і користувачу потрібно довго чекати.

Присутнє сортування товарів за порядком, зниженням чи зростанням ціни та за новизною.

Магазин немає і розділу «Улюблене», тобто є можливість додати товар лише одразу у «Кошик». Це не є гарним та ефективним рішенням для дорогих онлайн-магазинів.

Валюту змінювати не можна, а отже це не зручно для іноземних покцпців або для тих, хто орієнтується на іноземні ціни.

Необхідно зазначити фото товарів, що зроблені не якісно. Знімки ювелірних виробів мають погану якість і взагалі не передають коштовність прикрас.

Загалом цей магазин не справляє враження сучасного, надійного та якісного вебсайту.

1.4.4 Підсумки огляду наявних рішень

Під час огляду деяких із існуючих онлайн-магазинів, які спеціалізуються на продажі ювелірних виробів, було виокремлено певні критерії, за якими їх і було порівняно.

За цими аспектами було проведено аналіз, результати якого наведені у вигляді таблиці. (таблиця 1.1).

У даній таблиці під номером 1 представлено онлайн-магазин «Kochut», під номером 2 йде «MONO Jewerly», а номером 3 є «Світ прикрас». Також у зведенні використано позначки «+» та «-», які вказуються на наявність або

відсутність певного аспекту відповідно.

Таблиця 1.1 – Результати огляду наявних рішень

Критерії	1	2	3
Сучасний дизайн	+	+	-
Адаптивний дизайн	+	+	-
Анімації	-	-	-
Динамічний пошук	-	+	+
Сортування	+	+	+
Динамічна фільтрація	-	-	+
«Улюблене»	+	-	-
Можливість одразу додати товар з «Улюбленого» у «Кошик»	-	-	-
Можливість зміни валюти	+	+	-
Якісні фото виробів	+	+	-

Такий аналіз був необхідним кроком для створення дійсно якісного, сучасного та зручного онлайн-магазину, враховуючи функціонал вже існуючих застосунків.

1.5 Постановка задачі

Докладно розглянувши декілька існуючих онлайн-магазинів, які продають ювелірні вироби, було виокремлено важливі аспекти, які слід використати у проєкті, або навпаки уникати їх. Після цього це все було

поєднано у формулювання завдання.

Метою розробки є сучасний, простий і зрозумілий у використанні, конкурентоспроможний вебзастосунок, який представлятиме собою клієнт-серверний сервіс онлайн торгівлі ювелірними виробами. Він матиме перелік стандартних функцій, а також додаткові аспекти, які б змогли надовше затримувати увагу клієнтів та змушували б повертатися до сайту знову для здійснення покупок.

Першим кроком є створення дизайну, який буде передавати атмосферу бренду. Оскільки першим контактним пунктом з покупцем є сайт, то його дизайн має значний вплив на враження від магазину та рівень довіри до нього. Тож важливо не тільки зробити його привабливим, а і врахувати всі функціональні можливості вебзастосунку. Зокрема, кожна з функцій повинна мати зручний інтерфейс і забезпечувати ефективний процес взаємодії з покупцем.

Сучасні онлайн-магазини мають і адаптивний дизайн. Це макети усіх сторінок сайту, які підлаштовані під екрани менших розмірів. Це значно покращує користування застосунком, а також справляє гарне враження на споживачів.

UI дизайн має створювати гарну картину з естетичними вимогами бренду, які підкреслять унікальність кожного з виробів. У цей час UX дизайн повинен орієнтуватися на зручність взаємодії людини з сайтом.

При заходженні на сайт необхідно, щоб користувачу було відображено головну сторінку, де буде інформація про бренд, зображено деякі з товарів і можливість підписатися на розсилку магазину, яка дозволить людям першим дізнаватися про новинки, спеціальні пропозиції тощо.

Для авторизації потрібно зробити окрему сторінку, де користувач зможе ввести електрону адресу та пароль для входження в систему.

Якщо ж користувач не матиме акаунту, то для цього буде сторінка реєстрації з потребою ввести дані для створення акаунту клієнта.

Сторінкою, де покупцем буде проведено основний відсоток часу, є

каталог товарів. Даний розділ має задовільняти такі критерії, як простота використання, логічна навігація та зрозуміла структура. Ключовими аспектами на даному етапі є фільтрація, динамічний пошук та сортування. Фото виробів мають відповідати стандартам якості та бути у однаковому стилі.

Вікно фільтрації слід робити зручним та логічним для того, щоб користувачі могли швидко та легко обирати певні категорії та параметри, щоб знаходити потрібні товари. Для зручності треба реалізувати динамічну фільтрацію, коли б після кожного обраного фільтру оновлювався список товарів, аби клієнту не потрібно було постійно гортати і шукати маленьку кнопку «Застосувати фільтри».

Динамічний пошук також є функцією, яку дуже часто використовують при знаходженні на сторінках онлайн-магазину. При введенні кожного символу потрібно шукати лише ті вироби, які схожі за рядком, що ввів користувач, і демонструвати дані прикраси.

Також гість сайту може сортувати список товарів за певним принципом. Серед популярних варіантів це сортування за спаданням та зростанням ціни, новизною тощо. Це дасть змогу краще підібрати варіанти, наприклад, за ціною або популярністю та пришвидшить пошук.

Клієнту слід мати змогу виділити вподобаний виріб. Для таких цілей доцільним є створення розділу «Улюблене». Сюди можна додати позицію і зі сторнки каталогу, і зі сторінки самого товару. Присутня можливість додати товар вже у «Кошик», коли користувач вирішить, що хоче купити виріб, або ж видалити, якщо прикраса більше не актуальна.

Обов’язковою є окрема сторінка кожного з аксесуару. На ній буде розташована уся важлива інформація, наприклад, назва, метал, з якого зроблено виріб, чи є камені та які саме тощо. Перевагою також буде можливість обрати пакування замовлення. Додатково можна вказати інформацію щодо доставки.

Сторінка із розділом «Кошик» є однією із кінцевих точок знаходження клієнта на сайті. На ній повинен бути перелік обраних до замовлення

ювелірних виробів, поля, де можна розмістити інформацію щодо замовлення та кнопка для здійснення замовлення. Товари з кошику можна видалити або змінити їхню кількість.

На сайті слід виокремити і сторінку про акаунт споживача. На ній логічно буде відображати інформацію про акаунт, товари, які людина замовила і на які вже чекає, кнопку для виходу з акаунту тощо.

Крім того потрібна сторінка «Контакти», де користувач зможе знайти усю необхідну та діючу контактну інформацію. При виникненні будь-яких питань людина може використати оптимальний шлях зв’язку та поспілкуватися із консультантом.

Для того, щоб більше привернути увагу клієнта та забезпечити йому приємний процес перебування на сайті слід використати анімації. Вони роблять сайт живим, утримують увагу відвідувача і спонукають повертатися до покупок на даному вебсайті.

Система буде мати двокомпонентну архітектуру, яка складається з клієнтської та серверної частин. Клієнстька частина реалізована за допомогою фронтенд-фреймворку React. Він відповідає за взаємодію із користувачем, відображення інтерфейсу застосунку, здійснення фільтрації, сортування тощо. Серверна частина побудована на основі Node.js з використанням фреймворку Express. Даний розділ відповідає за обробку HTTP-запитів, оперування даними. Він формує базову логіку обробки замовлень, керування списком улюблених товарів та кошиком і надає REST API, щоб взаємодіяти із клієнтською частиною.

2 АНАЛІЗ ВИКОРИСТОВУВАНИХ ТЕХНОЛОГІЙ

2.1 Огляд засобів розробки

При розробці даного вебзастосунку було використано популярні та сучасні технології, які є зручними, ефективними та часто використовуються для реалізації подіних проєктів.

Етап розробки прототипу інтерфейсу та дизайну онлайн-магазину було реалізовано у середовищі Figma. Макети усіх сторінок та модальних вікон, зокрема: головної сторінки, сторінки входу та реєстрації, каталогу, улюбленого, сторінки товару, кошика, контактів, сторінки помилки, модального вікна фільтрів, сортування, оформлення замовлення – були розроблені з використанням інструментів Figma. Це дало змогу логічно розташувати всі елементи, продумати структуру та узгодити стилістику перед тим, як реалізувати їх за допомогою коду.

При створенні візуальної концепції та детального дизайну брендового пакування було задіяно Adobe Photoshop. Завдяки інструментам редагування шарів, кольорів та зображень було забезпечено якісне візуальне оформлення.

Для розробки клієнтської частини було використано JavaScript-фреймворк React. З його допомогою можна швидко створювати односторінкові додатки з сучасним адаптивним інтерфейсом. Навігація між сторінками без перезавантаження реалізована шляхом використання React Router. Для того, щоб стилізувати та розташувати усі елементи на сторінці використовувався CSS.

Збереження даних про кошик, товари в улюбленому та інформації щодо авторизації користувача, здійснюється через бібліотеку Redux. Щоб реалізувати асинхронні операції, наприклад, запити на сервер, було використано Redux Thunks.

Серверну частину було створено за допомогою Node.js, а саме з

використанням фреймворку Express.js. Завдяки даному фреймворку можна будувати REST API для роботи із запитами клієнта. Дана частина забезпечує взаємодію зі списком улюблених товарів та кошиком, а також надає доступ до даних.

Інформація про товари магазину зберігається у базі даних PostgreSQL. Завдяки бібліотеці pg Node.js здійснюється доступ до даних через SQL-запити.

Розробка інтерфейсу здійснювалася у середовищі Visual Studio Code. Для контролю версій проєкту використовувалася система Git. Для того, щоб розмістити код та надати можливість слідкувати за розробкою проєкту було використано платформу GitHub. Завдяки зручній системі гілок та легкого відстеження змін, процес додавання нового функціоналу став легким та контрольованим.

2.2 Інструментарій розробки

2.2.1 Figma

Figma – це онлайн-сервіс для створення дизайну інтерфейсів, який широко використовується для створення макетів вебсайтів, мобільних додатків тощо [2].

Вебзастосунок має зручний, зрозумілий інтерфейс та весь необхідний функціонал для швидкої розробки. (рисунок 2.1).

Дане середовище має можливість організувати спільну роботу над проєктом одночасно декільком користувачам, якщо цього потребує проєкт. Для цього не потрібно мати встановлене локальне програмне забезпечення, адже достатньо мати лише веббраузер.

Хмарне збереження файлів дає змогу володіти доступом до проєкту з будь-якого пристрою, який має інтернет з’єднання.

Зручність використання Figma у тому, що усі зміни у проєкті зберігаються автоматично в режимі реального часу. Це надає змогу

безперервно працювати над проєктом без потреби постійного ручного збереження.

Застосунок дозволяє налаштувати інтерактивність прототипів. Дана функція імітує реальні переходи між сторінками та забезпечує можливість краще продумати навігацію майбутнього проєкту.

Окрім цього дане середовище є зручним і для фронтенд-розробників, адже має функцію, яка допомагає легко переглядати технічну інормацію про кожен з наявних елементів дизайну. Це можуть бути розміри шрифтів, кольори, відступи та інші параметри. Також є можливість завантажити кожне із зображень макету у різних форматах.

За бажанням можна залишати коментарі для інших або важливі нотатки для себе. Дана функція спрощує процес обговорення деталей у процесі розробки та є ефективною для учасників команди.

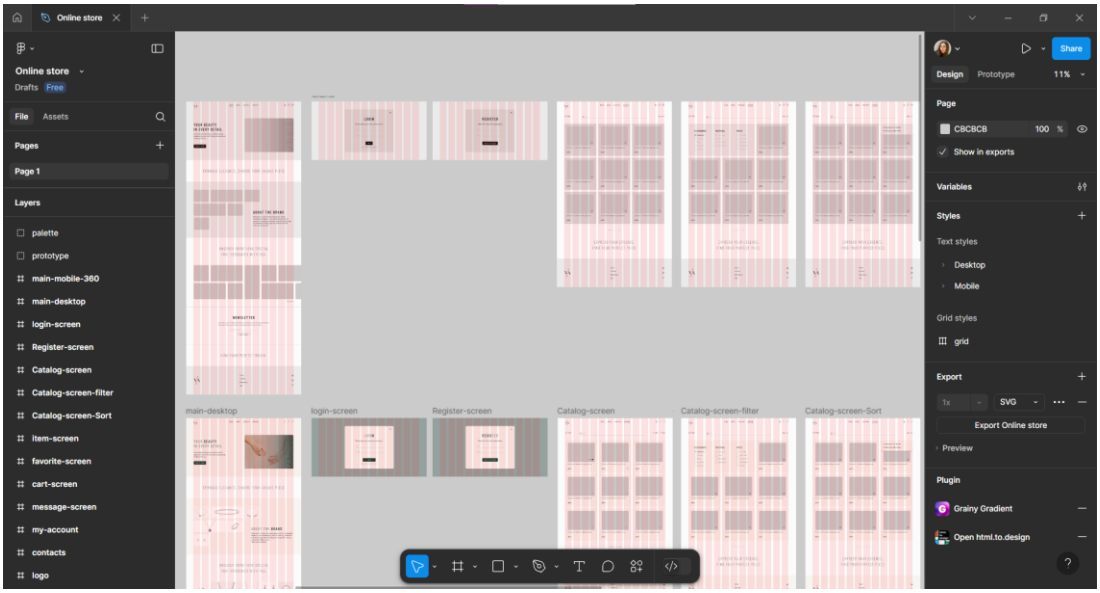

Рисунок 2.1 – Інтерфейс середовища розробки Figma

2.2.2 Adobe Photoshop

Adobe Photoshop – це багатофункціональний растровий графічний редактор [3]. Його широко використовують у таких сферах, як дизайн, фотографія, веброзробка тощо.

Застосунок має широкий інструментарій для обробки зображень, створення і редагування макетів, колажів, рекламної графіки, інтерфейсів вебзастосунків та іншого.

Редактор відомий своїм великим та гнучким набором різних інструментів та функцій, які забезпечують реалізацію задумів розробника. (рисунок 2.2). Користувач має можливість працювати із кольорами, шарами, текстом, масками, фільтрами та іншими складовими, які допомагають створити багатоскладову композицію. Саме тому його часто використовують у широкому спектрі проєктів, наприклад створення мокапів чи ретушування фотографії. У застосунку можна також готувати елементи для використання у цифрових медіа чи для друку.

Перевагою програми є підтримка роботи з високоякісними форматами, такими як PNG, JPEG, SVG, PSD тощо.

Під час роботи із програмою зміни зберігаються автоматично, адже наявна функція збереження резервних копій через певні проміжки часу.

Додаток підтримує можливість роботи з графічними планшетами, що є зручним при створенні власної графіки та елементів чи при обробці фото. Окрім того продукти Adobe інтегруються між собою, що значно спрощує робочий процес.

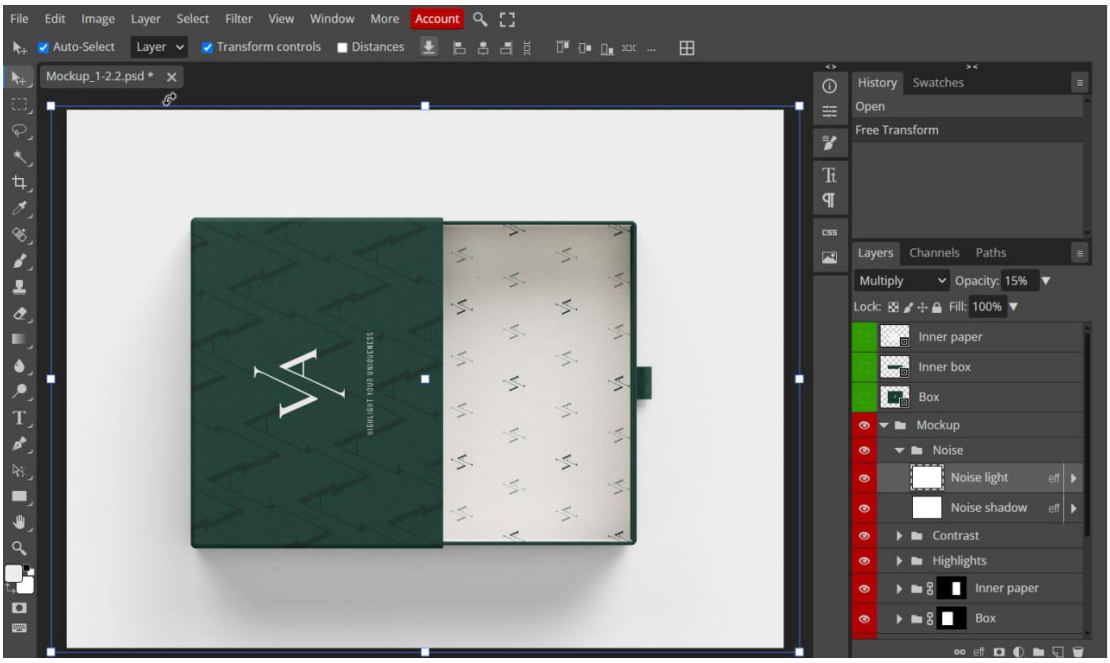

Рисунок 2.2 – Інтерфейс середовища розробки Adobe Photoshop

2.2.3 Visual Studio Code

Visual Studio Code – це кросплатформний редактор вихідного коду, який є одним із широко використовуваних інструментів завдяки своїй розширюваності [4]. Його було створено компанією Microsoft.

Середовище розробки підтримує велику кількість мов програмування, серед яких є JavaScript, HTML, CSS, C++ та багато інших.

Застосунок має інтуїтивно зрозумілий інтерфейс, що забезпечує зручність роботи. (рисунок 2.3). Користувач має змогу розділення екрану, щоб бачити одразу декілька вікон. Окрім того є можливість зміни теми оформлення та власного розташування панелей, які допомагають зручно оформити робочий простір.

Редактор також має розширення, що дають змогу збільшити функціональність роботи, адаптуючи його під вимоги конкретного завдання.

Процес розробки спрощується завдяки глибокій інтеграції застосунку із системою контролю версій Git та платформою GitHub.

Visual Studio Code дозволяє розробнику виконувати команди

безпосередньо з редактора, наприклад, при роботі з Git, адже має вбудований термінал. Окрім цього підтримується вбудований відладчик, завдяки якому можна зупиняти виконання програми у будь-яких точках, переглядати вміст змінних і покроково аналізувати логіку роботи програми для виявлення та виправлення помилок.

Написання коду пришвидшує функція IntelliSense, яка включає в себе підказки щодо параметрів і типів, автозавершення функцій тощо.

Додаток є кросплатформеним і доступним на різних операційних системах, зокрема Windows, macOS, Linux.

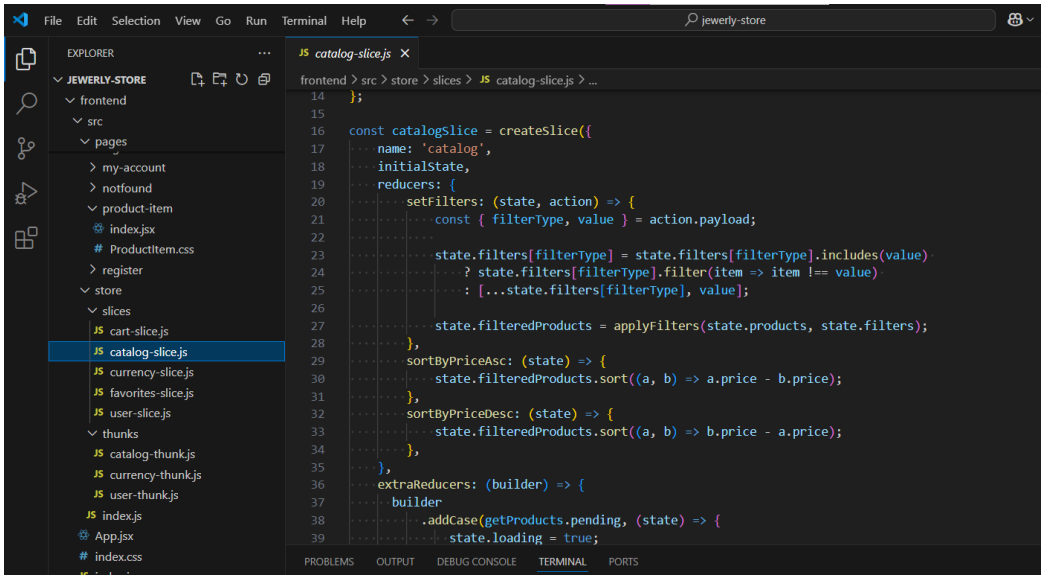

Рисунок 2.3 – Інтерфейс середовища розробки Visual Studio Code

2.3 Технології для розробки клієнтської частини

2.3.1 React

React – це декларативна JavaScript-бібліотека з відкритим вихідним кодом, яка призначена для створення інтерфейсів користувача [5]. Вона була розроблена компанією Meta і наразі активно підтримується.

Головною ідеєю даної бібліотеки є побудова інтерфейсу за допомогою компонентів.

Коннопентом є незалежна частина, яку можна використовувати у декількох частинах загального коду. Кожен з елементів відповідає за окремий фрагмент логіки чи відображення на сторінці. Дані частини можуть бути вкладеними, утворюючи при цьому дерево компонентів, що описує всю структуру UI.

React викоритовує спеціальну технологію, яка називається віртуальний DOM (Virtual DOM). Вона забезпечує оптимізацію оновлення сторінки, дозволяючи рендеринг лише окремих модулів інтерфейсу, де саме відбулися зміни.

2.3.2 JavaScript

JavaScript – це високорівнева динамічна об’єктно-орієнтована прототипна мова програмування, яка часто використовується при створенні інтерактивних сторінок і сайтів та відзначається динамічною типізацією і має прототипну модель об’єктів [6].

Вона працює у браузерах і надає змогу реалізовувати маніпуляцію елементами HTML через DOM API, перевіряти форми, обробляти події користувача, оновлювати контент сторінки без повного перезавантаження. При створенні SPA вона є незамінною.

JavaScript має різні фреймворки, зокрема React, Vue, Angular. Вони дозволяють розширити функціонал та можливості мови, допомагаючи ефективно реалізовувати певні задачі.

Також JavaScript можуть використовувати середовища, наприклад Node.js, на серверній стороні для того, щоб обробляти запити, працювати з базами даних тощо.

2.3.3 SPA

Односторінкові додатки (SPA) – це архутектурний підхід до створення

вебдодатків, який спрямовано на мінімізацію кількості повних перезавантажень сторінки браузера, забезпечивши при цьому динамічну взаємодію з користувачем.

Основна ідея полягає у тому, що необхідна структура додатку та ресурси, серед яких є HTML, CSS, JavaScript, завантажується лише один раз на початку використання, а далі контент оновлюється вибірково, лише окремими частинами.

Перевагою такого підходу є швикість реагування інтерфейсу. Це знижує навантаження на сервер та покращує користувацький досвід.

2.3.4 HTML

HTML – це стандартизована мова розмітки, яка детермінує зміст та структуру вебдокументів і є фундаментальною основою при створенні вебсторінок [7].

Її призначення полягає в організації та візуалізації контенту в інтернеті, забезпечуючи при цьому узгодженість між усіма іншими браузерними платформами.

Дана мова має ієрархічну систему елементів, які представлені у вигляді тегів, що описують різні сегменти документа, такі як заголовки, параграфи, зображення, посилання та інше.

Особливість HTML є семантична розмітка, яка використовує спеціальні теги, наприклад <header>, <footer>, для надання змісту структури. Це надає розробникам можливість створювати сторінки із логічно упорядкованою структурою.

2.3.5 CSS

CSS – це спеціальна мова стилів, яка використовується для опису візуального подання документу, які були створенні за допомогою HTML чи

інших мов розмітки.

Вона дозволяє розробникам для окремих елементів створити правила оформлення, такі як шрифти, кольори, розміри та положення елементу на сторінці.

Стилі до певних елементів на основі їх класів, атрибутів або ідентифікаторів можна застосовувати завдяки селекторам. Вони служать певними «індикаторами» для вибору елементів, до яких буде застосовано дані правила.

Також завдяки даній мові можна створювати адаптивні дизайни, які динамічно змінюють власний вигляд, залежно від прописаних стилів для конкретного елементу.

2.3.6 React Router

React Router – це декларативна бібліотека, яка слугує для управління маршрутизацією у додатках, що були створені на React.

З її допомогою розробник може створювати динамічні багатосторінкові інтерфейси та відображати окремі компоненти застосунку залежно від поточного URL, не потребуючи при цьому перезавантаження усієї сторінки.

Маршрути у React Router визначаються як React-компоненти. Вони асоціюють певний URL-шлях з конкретним компонентом для рендерингу.

Важливою функціональною можливістю React Router є вкладені маршрути, які дозволяють структурувати багатокомпонентні інтерфейси.

2.3.7 NPM

NPM – це система управління пакетами, яка є стандартним менеджером для середовища Node.js. Вона використовується для того, щоб керувати, встановлювати, оновлювати та видаляти модулі, фреймворки й бібліотеки, які потрібні для розробки JavaScript-застосунків.

Він має три основні компоненти:

- репозиторій NPM, який містить безліч модулів та бібліотек JavaScript, які є у відкритому доступі для усіх розробників.

- командний рядок NPM (CLI), який є інструментом, що дозволяє розробникам взаємодіяти із репозиторієм, використовуючи командний рядок. Він є основним засобом для інтеграції пакетів у локальне середовище розробки.

- файл «package.json», який описує залежності проєкту, основну метаінформацію про проєкт, зокрема ім’я, версія, автор тощо та скрипти для автоматизації завдань.

2.4 Технології для розробки серверної частини

2.4.1 Node.js

Node.js – це платформа для серверної розробки, яку побудовано на рушії V8 від Google Chrome. Вона забезпечує можливість виконання JavaScript-коду поза межами веббраузера для того, щоб створювати високопродуктивні мережеві додатки [8].

Метою Node.js є забезпечення ефективного використання ресурсів при обробці одночасних запитів. Це відбувається завдяки неблокуючій моделі введення-виведення та подієво-орієнтованій архітектурі, які дають змогу використовувати однопотокову модель з циклом замість того, щоб створювати окремий потік для кожного вхідного запиту. Такий підхід дозволяє обробляти велику кількість одночасних з’єднань і значно підвищує продуктивність, порівнюючи із багатонитевими серверами.

2.4.2 Express.js

Express.js – це гнучкий та легкий у використанні Node.js фреймворк,

який призначений для полегшення створення серверних додатків та API. Він забезпечує маршрутизацію та обробку HTTP-запитів та інтеграцію з проміжним програмним забезпеченням.

Express.js має асинхронну архітектуру для одночасної обробки багатьох запитів. Він підтримує адаптивну систему маршрутизації для швидкого визначення обробників HTTP. Його особливістю також є модульна архітектура, що дає змогу інтегрувати лише ті компоненти, які необхідно.

Система маршрутизації дає можливість чітко визначити обробники для різних HTTP-методів, зокрема GET, POST тощо, та URL-шляхів, що використовується для побудови RESTful API та вебзастосунків.

2.4.3 REST API

REST API – це архітектурний стиль для сворення мережевих застосунків, що спрямовані на взаємодію між системами клієнта і сервера завдяки стандартним протоколам, зокрема HTTP [9].

Його метою є стандартизація цієї взаємодії, використовуючи класичні HTTP-методи, серед яких є GET, POST, PUT, DELETE тощо, для того, щоб виконувати операції над ресурсами без необхідності збереження стану на сервері.

Обмін даними передбачає використання стандартних форматів даних, таких як JSON чи XML, що надає можливість легкої серіалізації та сумісність із різними мовами програмування.

REST API забезпечує можливість масштабування через слабке зв’язування між компонентами, що дає змогу сервісам розвиватися автономно, не впливаючи на цілісну архітектуру.

2.4.4 PostgreSQL

PostgreSQL – це відкрита, об’єктно-реляційна система керування базами

даних, яка відзначається ефективністю та можливістю масштабування [10].

Дана система дає змогу реалізовувати складні запити, застосовує багатоверсійне управління паралелізмом (MVCC), керує транзакціями з гарантіями ACID та пропонує розширені механізми індексування, такі як B-дерева, GiST, GIN та BRIN.

Також PostgreSQL підтримує багато типів даних та надає змогу створювати функції, використовуючи різні мови програмування.

Дана платформа має вагомі переваги, що виділяють її серед аналогів. Зокрема, повна відповідність стандартам SQL та підтримка складних запитів гарантують ефективну роботу з великими об’ємами інформації.

2.4.5 Nodemailer

Nodemailer – це провідна бібліотека для Node.js, призначена для відправлення електронної пошти з вебдодатків. Вона забезпечує надійну інтеграцію поштових сервісів та підтримує широкий спектр транспортних протоколів [11].

Дана бібліотека підтримує SMTP, SendGrid та інші поштові сервіси, забезпечує відправлення текстових та HTML-повідомлень з вкладеними файлами та надає можливості роботи з шаблонами електронних листів.

Також Nodemailer відзначається високою продуктивністю при обробці великих об’ємів поштових повідомлень та забезпечує асинхронну обробку запитів.

Основними перевагами є інтуїтивний API з детальною документацією, гнучкість конфігурації транспортних механізмів та стабільність роботи. Бібліотека дозволяє налаштовувати різні параметри безпеки, включаючи TLS/SSL шифрування, що забезпечує захист даних під час передачі.

3 РЕАЛІЗАЦІЯ КЛІЄНТ-СЕРВЕРНОГО СЕРВІСУ

3.1 Розробка користувацього інтерфейсу та дизайну

3.1.1 Створення структури сайту та прототипування

Figma є популярним застосунком для розробки UI/UX-дизайну і має широкий попит, оскільки надає багато можливостей, високу зручність роботи та забезпечує онлайн доступ до проєкту.

Для створення клієнт-серверного сервісу було обрано даний додаток, адже він має простий інтерфейс для швидкої роботи, дає змогу переглядати властивості елементів і використовувати їх для верстки онлайн-магазину.

На першому етапі відбувалася розробка sitemap, тобто визначення ключових сторінок сайту, зв’язків між ними та логіки навігації. Метою даного кроку є визначення загальної структури майбутнього вебресурсу. (рисунок 3.1).

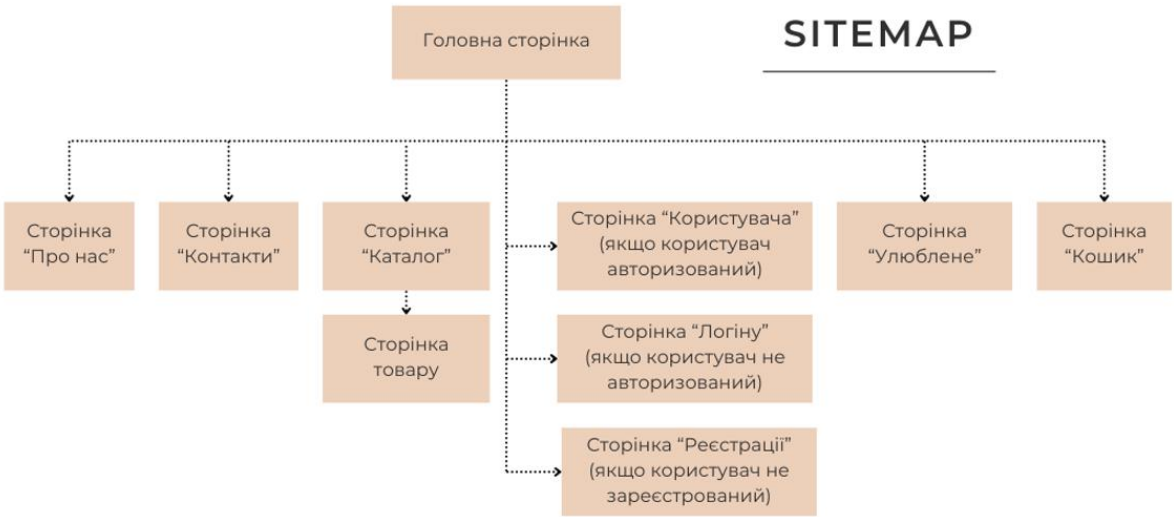

Рисунок 3.1 – Sitemap клієнт-серверного сервісу з продажу ювелірних виробів

Далі було проаналізовано поведінку переміщень користувача по застосунку при виконанні певних дій. Для цього було створено User Flow, щоб побудувати логіку переходів клієнтів, наприклад, для замовлення товару на сайті. (рисунок 3.2).

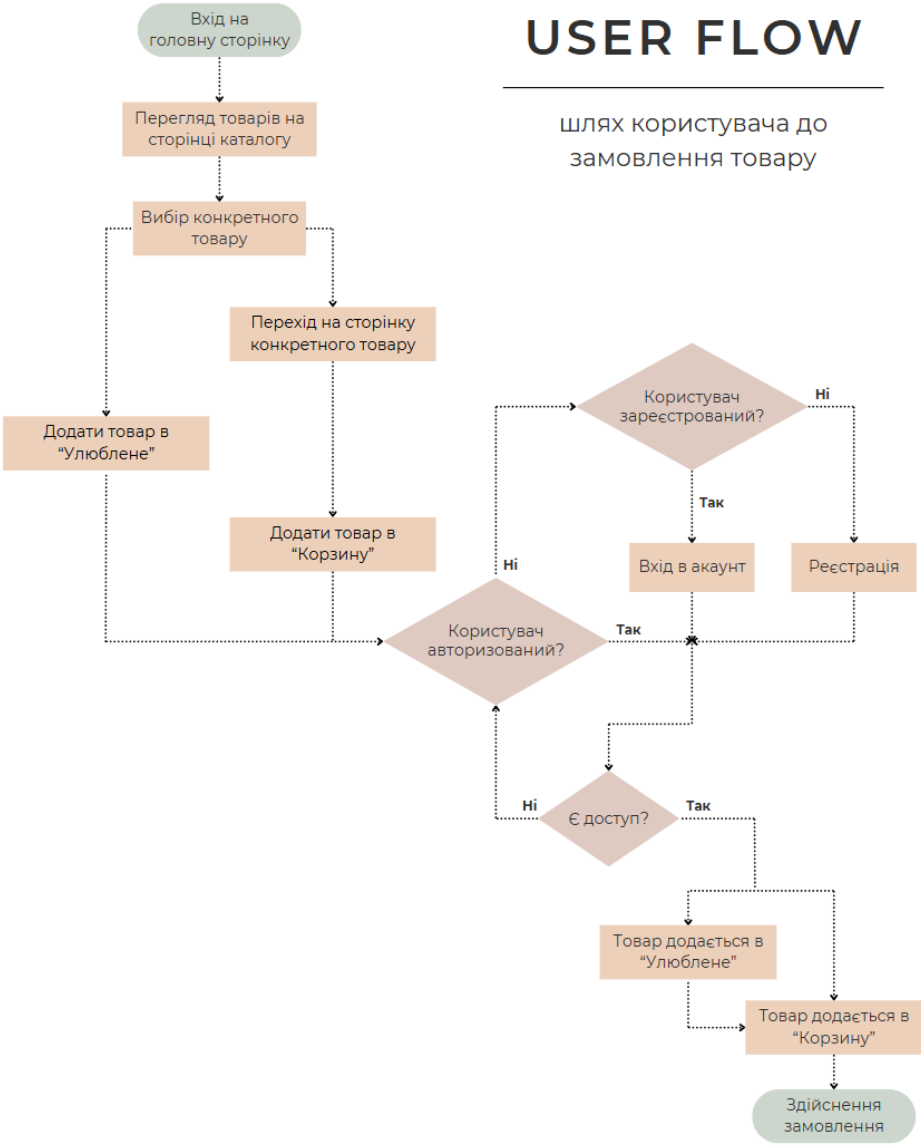

Рисунок 3.2 – User Flow для описання шляху до замовлення товару

Отже, спочатку користувач потрапляє на головну сторінку додатку, звідки може перейти до сторінки «Каталогу». У цьому розділі є можливість перегляду різних варіантів, які можна додавати до «Улюбленого» або, перейшовши на сторінку конкретного товару, також додавати його до «Кошику». На даному кроці відбувається перевірка чи авторизувався клієнт.

У ситуації, коли покупець не має акаунту, йому буде запропоновано його створити. У іншому випадку відвідувач застосунку здійснює вхід у акаунт, після чого повинна з’явитися можливість використання розділів «Улюбленого» та «Кошику». Якщо клієнт вже авторизувався до цього і отримав доступ, тоді товар повинен додатися до «Улюбленого» чи «Кошику». Для того, аби здійснити замовлення, необхідно, щоб виріб перебував у «Кошику», де клієнт зможе заповнити форму і завершити процес оформлення покупки.

За допомогою визначеної у Sitemap структури відбувалася подальша розробка прототипів усіх сторінок. (рисунок 3.3)

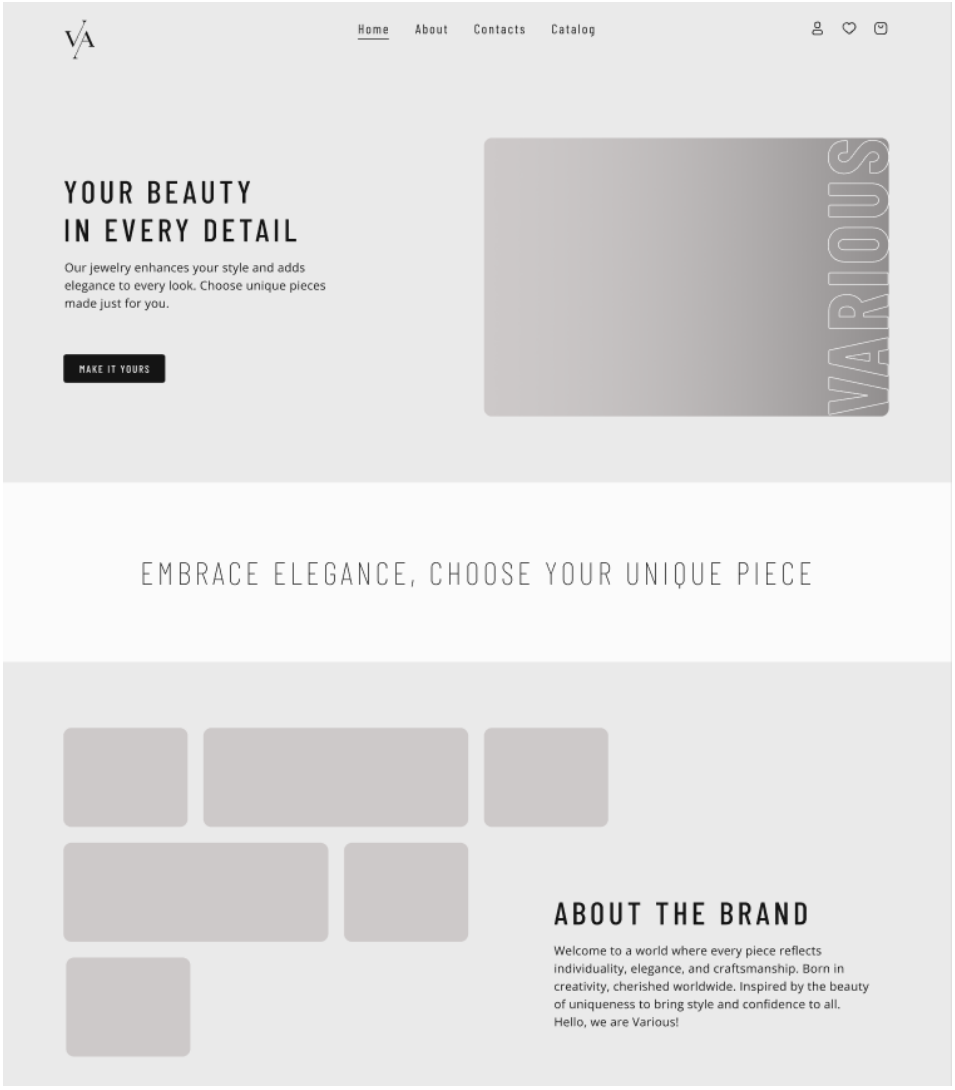

Рисунок 3.3 – Прототип головної сторінки

Було створено каркасні макети, які визначали базове розташування елементів, що не мають деталізованого дизайну. Метою прототипування є перевірка зручності використання елементів інтерфейсу, адже вони дають розуміння того, наскільки інтуїтивно користувач буде взаємодіяти із застосунком. На даному етапі визначаються також проблеми навігації чи доступність вікон та кнопок.

3.1.2 Візуальна концепція та дизайн

Наступним кроком було проведено аналіз існуючих онлайн-магазинів, що спеціалізуються на продажі прикрас, зокрема дизайнерські рішення, наявний функціонал, зручність інтерфейсу тощо. Огляд наявних варіантів надав розуміння, які рішення та креативні підходи слід використати під час розробки.

При створенні візуальної концепції важливим є забезпечення чіткого формування стилістики та впізнаваності компанії. Ключовими аспектами, що потрібно передати у дизайні стали:

- використання кольорів та шрифтів, що підкреслять імідж;
- вишуканість та мінімалістична композиція;
- легкість під час взаємодії.

Для цього було використано поєднання темно-зеленого та молочного кольорів. Перший транслює ексклюзивність, благородність та стабільність, а другий символізує легкість, чистоту та вишуканість. Дані відтінки разом формують одночасно елегантну та стриману концепцію, що відповідають характеру бренду.

Типографіка підтримує стилістику, адже один із шрифтів, що використано у логотипі, має засічки, а інший є простим та легким для сприйняття. Одночасне використання даних шрифтів забезпечує баланс між класикою та читабельністю, підкреслюючи преміальність магазину.

Дотримання вимог дало змогу забезпечити функціональний та

комфортний для користувача інтерфейс. (рисунок 3.4).

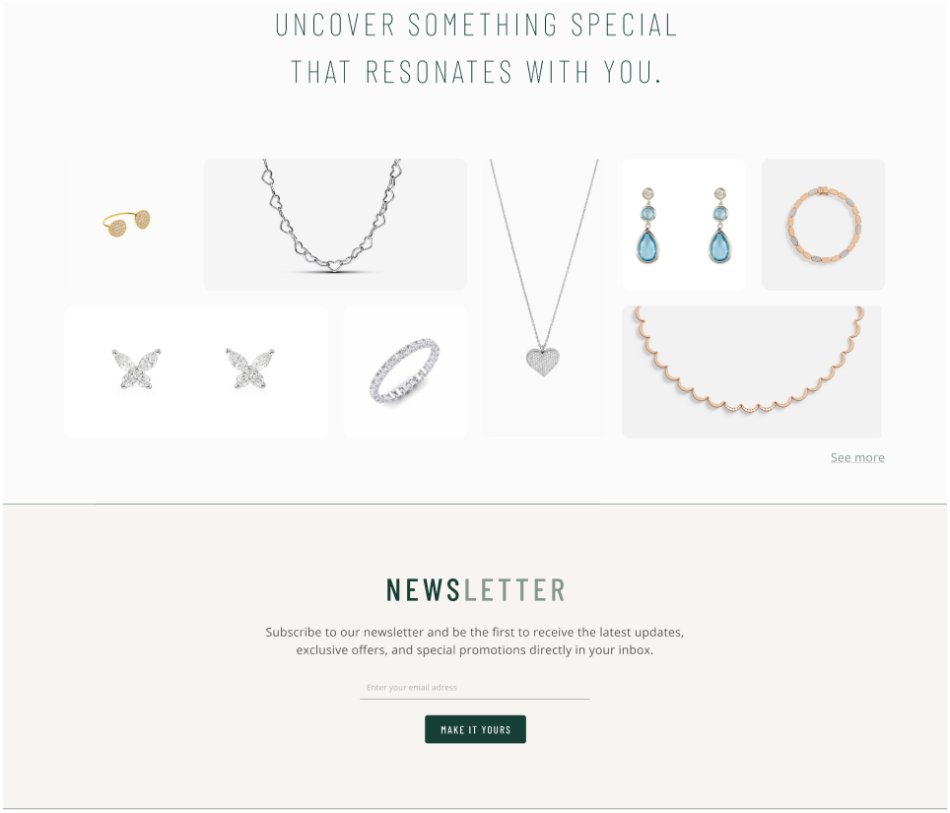

Рисунок 3.4 – Дизайн головної сторінки

3.1.3 Адаптивний дизайн

Адаптивний дизай є потребою кожного вебзастосунку, адже наразі вебдизайн має відповідати різним просторям для забезпечення оптимального відображення незалежно від розміру екрана. Відсутність адаптації може спричинити некоректне розташування контенту, що ускладнить читання та навігацію.

Дане оформлення створюється завдяки відповідним медіа-запитам, що коригують відображення контенту згідно розміру дисплею. Зазвичай визначаються стилі для мобільної, планшетної та десктопної версій.

Для маленьких екранів зазвичай навігація змінюється на компактне гамбургер-меню для зручності. (рисунок 3.5).

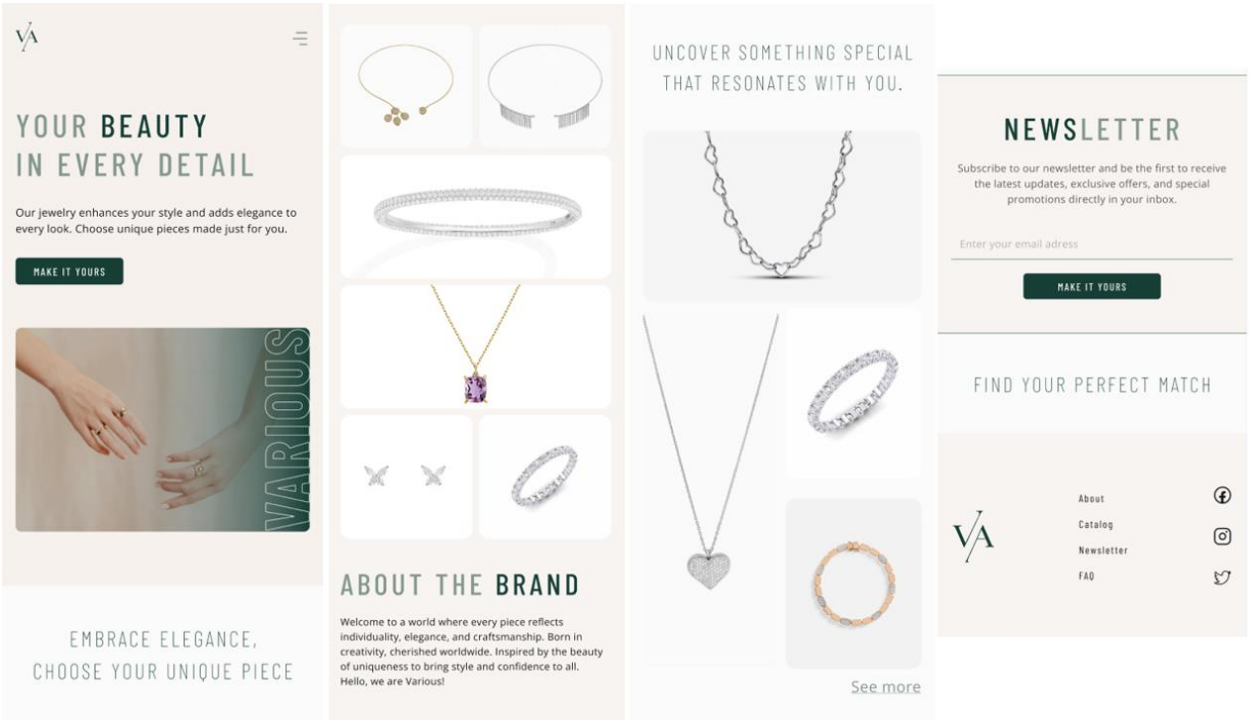

Рисунок 3.5 – Мобільна версія головної сторінки

Для пришвидшення процесу адаптивності дизайну у програмі редактора використовують Auto Layout при створенні першого розміру екрану. Надалі, корегуючи ширину, елементи будуть автоматично приймати правильне положення згідно сітки.

3.2 Реалізація клієнтської частини

3.2.1 Структура клієнтського застосунку

Проєкт побудовано згідно чіткої структури, що заснована на принципах модульності та логічного поділу файлів. (рисунок 3.6). Такий підхід забезпечує комфортну навігацію для розробника, ефективне керування ресурсами та можливість легкого розширення функціональності.

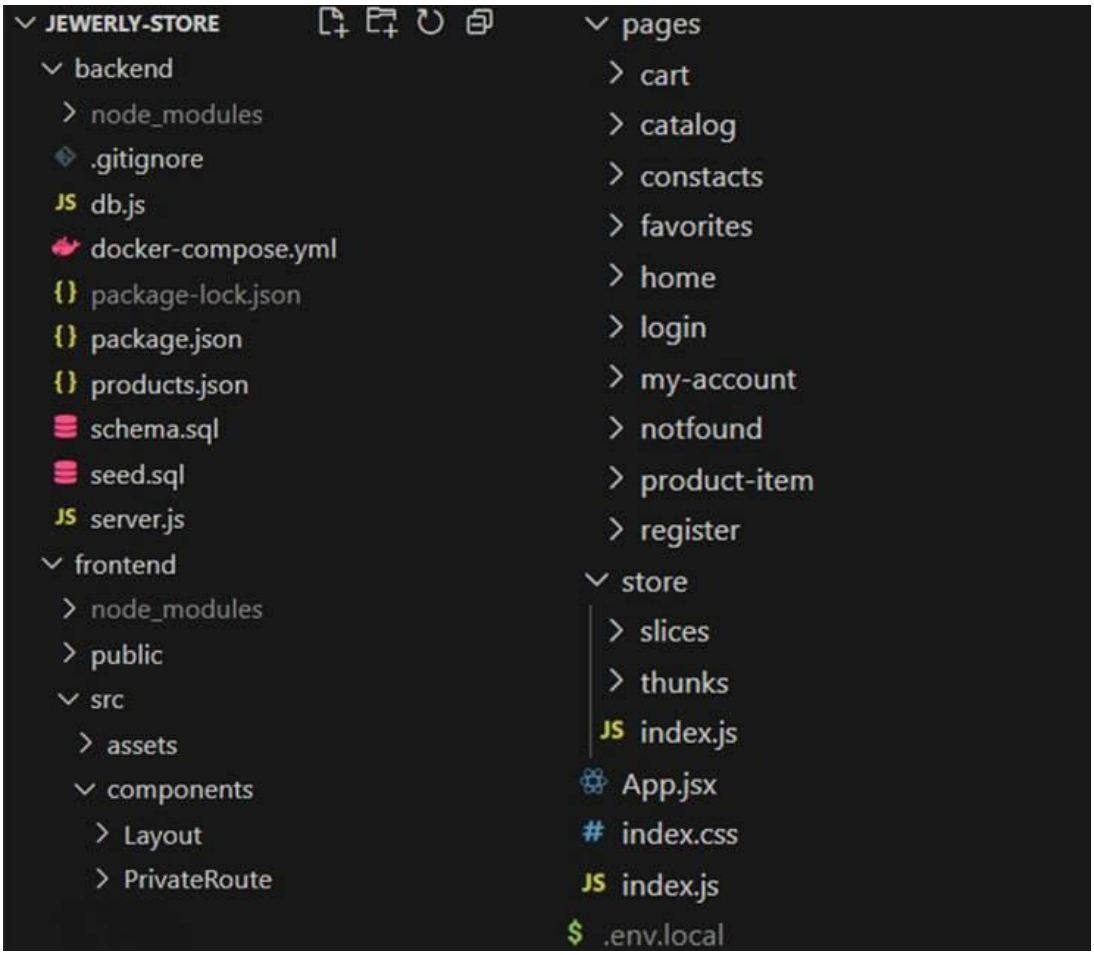

Рисунок 3.6 – Структура клієнтського застосунку

Файлова система клієнт-серверного сервісу містить кілька основних директорій, кожна з яких має власне призначення:

- assets – використовується для зберігання для таких ресурсів, як зображення та іконки;
- components – містить в собі набір UI-компонентів, що неодноразово використовуються у проєкті, для зменшення кількості коду;
- pages – має багато піддиректорій, кожна з яких має декілька файлів, які відповідають за певну сторінку сайту, зокрема головна сторінка, сторінка каталогу, товару, кошику, улюбленого, входу, реєстрації, помилки тощо;
- store – забезпечує управління глобальним станом застосунку та має піддиректорії slices і thunks, що відповідають за логіку управління даними та механізми обробки асинхронних запитів до API відповідно.

3.2.2 Реалізація маршрутизації через React Router

Для реалізації маршрутизації було обрано React Router, оскільки він надає зручну організацію переходів між розділами додатку.

До списку маршрутів сайту було додано усі необхідні сторінки сервісу:

- головна (/home);
- каталог (/catalog);
- сторінка товару (/catalog/:id);
- про нас (/about);
- контакти (/contacts);
- вхід (/login);
- реєстрація (/register);
- акаунт користувача (/my-account);
- улюблене (/favorites);
- кошик (/cart);
- сторінка помилки (/*).

Було розроблено приватні маршрути, тобто деякі розділи доступні лише для авторизованих користувачів, зокрема сторінки «Кошик» та «Улюблене» (лістинг 3.1).

Лістинг 3.1 – Реалізація приватних маршрутів

```
<Route element={<PrivateRoute />}>
     <Route path='cart' element={<Cart />} />
     <Route path='favorites' element={<Favorites />} />
     <Route path='my-account' element={<MyAccount />} />
</Route>
```

Для цього створено компонент PrivateRoute, що перевіряє наявність у клієнта токена доступу. У разі, якщо відвідувач сайту не авторизований, його буде автоматично направлено на форму для входу (лістинг 3.2).

Лістинг 3.2 – Перевірка доступу до приватних маршрутів

```
const PrivateRoute = () => {
    const token = useSelector(state => state.user.token);
    return token ? <Outlet /> : <Navigate to='/login' />;
};
```

Окрім статичних маршрутів підтримуються динамічні маршрути, наприклад, шлях /catalog/:id для відображення окремої сторінки для кожного виробу. Усі прикраси мають унікальний id, який дає змогу отримувати відповідні дані та відображати їх у окремому розділі конкретного товару.

У випадку введення некоректної або неіснуючої URL-адреси користувача буде перенаправлено на сторінку помилки. Такий підхід допомагає уникнути непередбачуваних ситуацій та одразу показує про наявність збою.

3.2.3 Керування станом за допомогою Redux

Як основний механізм управління станом у клієнтському застосунку було використано Redux. Даний вибір надав можливість ефективно контролювати взаємодію між компонентами, зменшити повторення логіки та забезпечити масштабованість. Для оптимізації роботи з Redux було використано Redux Toolkit.

У застосунку архітектура Redux структурована за допомогою чіткого розділення стану на логічні частини, які називаються слайсами (slices). Кожен з них відповідає за окрему частину логіки роботи сервісу. Зокрема:

- cart-slice – містить дані про товари, які були додані до кошика користувачем;
- catalog-slice – відповідає за збереження списку товарів та параметрів фільтрації, сортування і пошуку;
- currency-slice – керує вибором валюти та містить актуальні курси обміну, що отримуються через асинхронні запити;

- favorites-slice – управляє списком позицій, які були додані до улюблених товарів;

- user-slice – містить логіку збереження даних користувача, коли його було авторизовано;

Взаємодія з Redux у компонентах реалізується через useDispatch та useSelector, за допомогою яких можна змінювати стан та отримувати актуальні дані зі сховища відповідно.

Зокрема, dispatch(setFilters({filterType, value})) передає нові параметри при оновленні фільтрів товарів у каталозі. Для належної обробки цих змін у catalog-slice задіюється відповідний редуктор setFilters (лістинг 3.3).

Лістинг 3.3 – Редуктор setFilters застосування фільтрів у catalog-slice

```
setFilters: (state, action) => {
            const { filterType, value } = action.payload;

                                        state.filters[filterType]     =
state.filters[filterType].includes(value)
                  ? state.filters[filterType].filter(item => item
!== value)
                : [...state.filters[filterType], value];

              state.resultProducts = applyFilters(state.products,
state.filters);
},
```

Дана функція перевіряє чи наявний обраний товар у відповідному масиві фільтрів та якщо обраного варіанту немає, тоді він додається, якщо присутній – видаляється. Тобто даний редуктор змінює стан фільтрів та оновлює перелік виробів, що підпадають під обрані аспекти, динамічно.

Крім того у застосунку було реалізовано асинхронні дії (thunks), використовуючи Redux Thunk. Це надало змогу виконувати певні запити до серверної частини, отримувати дані з API, а потім змінювати стан після обробки результату.

Для отримання переліку товарів із сервера було реалізовано функцію getProducts (лістинг 3.4).

Лістинг 3.4 – Асинхронне отримання списку товарів через Redux Thunk

```
export const getProducts = createAsyncThunk('catalog/getProduct',
async () => {
    try {
                              const      response      =      await
fetch(`${BASE_URL}${GET_PRODUCTS}`);
        if (!response.ok) {
            throw new Error('Failed to load products!');
        }
        return await response.json();
    } catch (error) {
        throw error;
    }
});
```

Дана функція є Redux Thunk, який для отримання списку товарів виконує асинхронних запит до сервера. Вона повертає розпарсовані дані, якщо надісланий GET-запит на визначену URL-адресу було виконано успішно. У разі виникнення помилки, функція генерує виняток, який буде оброблено у цілях забезпечення коректної роботи сайту.

3.2.4 Динамічна фільтрація, сортування та пошук

Необхідними аспектами кожного онлайн-магазину при взаємодії з каталогом товарів є такі функції, як фільтрація, сортування та пошук. Вони допомагають користувачу швидко орієнтуватися у списку позицій, знаходити потрібні позиції та налаштовувати відображення виходячи із певних критеріїв.

Фільтрація дає можливість відвідувачу сайту обирати певні критерії, згідно яких буде відфільтровано список виробів.

У даному проєкті функція має такі основні критерії:

- категорії прикрас («Necklace», «Choker» тощо);
- матеріал («Rose gold», «Silver» тощо);
- ціновий діапазон («$100-500», «$500-1000» тощо).

Користувач має можливість одночасного застосування декількох фільтрів.

Функціонал реалізовано через Redux, де стан активних фільтрів зберігається у filters об’єкту catalog-slice. Зміни чекбоксів обробляються у handleFilterChange (лістинг 3.5).

Лістинг 3.5 – Функція обробки змін у фільтрах

```
const handleFilterChange = (filterType, value) => {
        dispatch(setFilters({filterType, value}))
}
```

Усі параметри відслідковуються компонентом модального вікна, який у реальному часі показує, які пукти у розділах фільтрації обрано, а також які товари підлягають обраним аспектам. (рисунок 3.7).

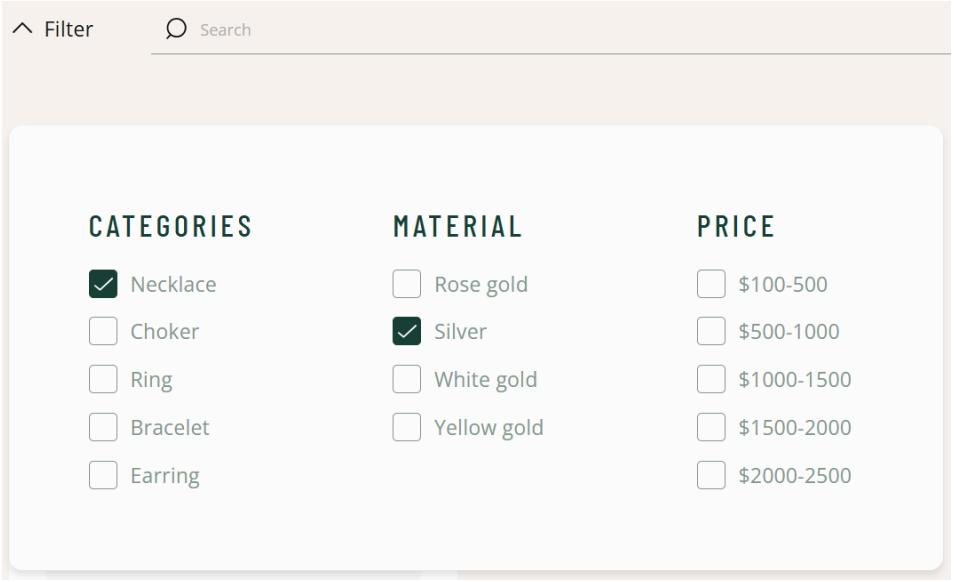

Рисунок 3.7 – Модальне вікно фільтрації

Сортування допомагає зручно та швидко впорядкувати товари на сторінці. Це зручно при роботі з великим асортиментом продукції.

Функція має два варіанти сортування списку позицій:

- За зростанням ціни – спочатку виводяться товари із найменшою ціною;
- За спаданням ціни – спочатку виводяться товари із найбільшою ціною.

Сортування здійснюється шляхом виклику dispatch(sortByPriceAsc()) або dispatch(sortByPriceDesc()), згідно натиснутої кнопки, які ініціюють зміну стану у catalog-slice (лістинг 3.6).

Лістинг 3.6 – Функції сортування

```
sortByPriceAsc: (state) => {
     state.resultProducts.sort((a, b) => a.price - b.price);
},
sortByPriceDesc: (state) => {
     state.resultProducts.sort((a, b) => b.price - a.price);
},
```

Функції викликаються при натисканні на відповідну кнопку у модальному вікні сортування. (рисунок 3.8).

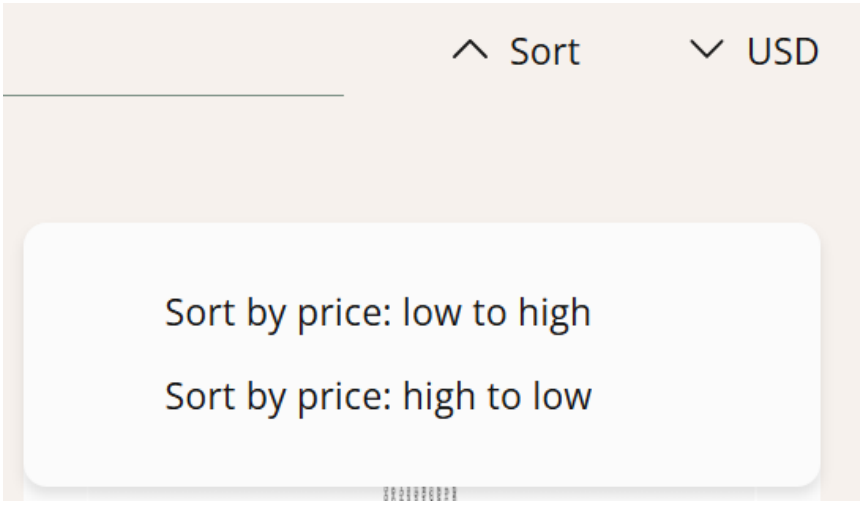

Рисунок 3.8 – Модальне вікно сортування

Пошук є важливим елементом навігації каталогу, оскільки дозволяє знайти потрібний товар за ключовими словами. (рисунок 3.9).

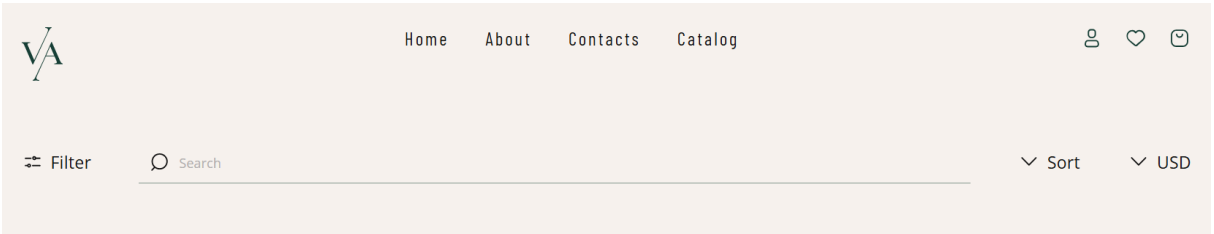

Рисунок 3.9 – Пошуковий рядок

Функцію реалізовано через оновлення searchQuery, що зберігається у catalog-slice (лістинг 3.7).

Лістинг 3.7 – Функція пошуку

```
setSearchQuery: (state, action) => {
            state.searchQuery = action.payload;
            state.filters = {
                categories: [],
                materials: [],
                price: []
            };
              state.resultProducts = applyFilters(state.products,
state.filters, action.payload);
}
```

Після введення пошукового запиту стан searchQuery змінюється та обробляється у редукторі setSearchQuery. Також скидаються усі активні фільтри для того, аби отримати дійсні результати та вибірку з усього каталогу.

3.2.5 Зміна валюти

Було додано можливість зміни валюти, щоб користувачі могли конвертувати ціни товарів та користуватися сайтом, наприклад, у інших країнах. Отримання діючого валютного показнику забезпечує точність розрахунку цін.

Вартість розраховується згідно актуального курсу валюти, яка була обрана при натисканні на відповідну кнопку. (рисунок 3.10). Наразі наявні три грошові одиниці:

- USD;
- UAH;
- EUR.

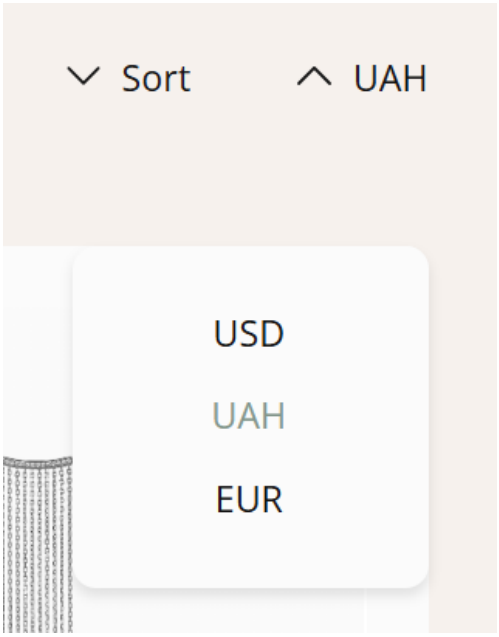

Рисунок 3.10 – Модальне вікно зміни валюти

Функціонал реалізовано за допомогою Redux Thunk, який дозволяє виконувати асинхронні запити до API. За рахунок функції fetchRates надсилається GET-запит до Currency Freaks, використовуючи при цьому API-ключ, який було надано на сайті сервісу (лістинг 3.8).

Лістинг 3.8 – Реалізація функціоналу зміни валюти

```
export const fetchRates = createAsyncThunk(
    'currency/fetchRates',
    async (_, thunkAPI) => {
        try {
                                    const    response    =    await
fetch(`https://api.currencyfreaks.com/latest?apikey=${API_KEY}&s
ymbols=USD,UAH,EUR`);

            if (!response.ok) {
                throw new Error('Failed to fetch currency rates');
            }

            const data = await response.json();
            return data.rates;
        } catch (error) {
            return thunkAPI.rejectWithValue(error.message);
        }
    }
);
```

При успішному виконанні запиту буде отримано об’єкт, що містить у собі курси валют. Якщо виникне помилка, то буде згенеровано виняток та з’явиться значення помилки.

За зміну поточної валюти на іншу та її збереження у сховищі відповідає функція setCurrency у currency-slice. Завдяки цьому вибір певної грошової одиниці не втратиться при оновленні сторінки (лістинг 3.9).

Лістинг 3.9 – Функціонал зміни валюти та її збереження

```
setCurrency(state, action) {
     state.selected = action.payload;
     localStorage.setItem('currency', action.payload);
}
```

У майбутньому функціонал зміни валюти можна розширити, додавши підтримку нових грошових одиниць.

Варто впровадити механізм, що буде автоматично визначати геолокацію клієнта. Тобто, слід відслідковувати, де саме користуються онлайн-магазином, аби визначити які валютні курси є актуальними. Також надалі слід автоматично встановлювати початкову валюту за регіоном.

Дані маніпуляції можуть покращити досвід використання сайту закордонними відвідувачами онлайн-магазину, забезпечуючи їм зручний доступ до товарів у звичній для них валюті.

3.3 Реалізація серверної частини

3.3.1 Побудова серверної логіки

Серверна складова онлайн-магазину функціонує на базі Node.js з використанням Express.js.

Забезпечення чіткого розподілення відповідальності між різними елементами, оптимізація управління ресурсами та спрощення розширення

функціоналу досягається завдяки використанню архітектури MVC [12]. Схему комунікації між сервером та клієнтом наведено нижче. (рисунок 3.11).

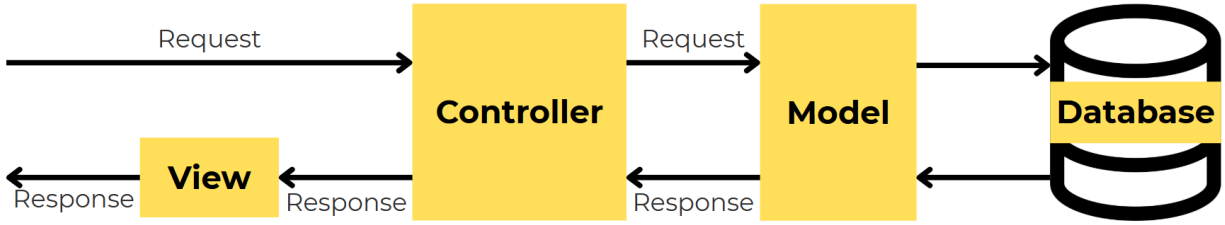

Рисунок 3.11 – Архітектура серверної частини

Кожен із рівнів має окреме призначення у даній архітектурі.

Першим кроком у клієнтській частині, що побудована як SPA-додаток на основі React, відбувається надсилання HTTP-запиту до серверу. Метою запиту може бути авторизація, отримання списку товарів, керування кошиком, оновлення валютного курсу тощо.

Обробка запитів відбувається завдяки контролерам, які обробляють запити, перевіряють введені дані на коректність та взаємодіють із моделями для виконання певних дій. Для того, щоб ефективно структурувати маршрути відповідно до логіки їх функціональності було використано REST API.

За взаємодію із базою даних відповідає модель. Вона забезпечує запис, читання, оновлення чи видалення інформації. Дані операції виконуються завдяки SQL-запитам у PostgreSQL, що викликаються із контролерів.

База даних містить у собі усі необхідні дані, в даному випадку про товари, користувачів, їх замовлення тощо.

Уся інформація передається у JSON, а HTML-сторінки не генеруються у SPA-архітектурі. Кожна API-відповідь містить виключно дані, що призначені для клієнтського додатку. Це сприяє ефективному розподілу логіки представлення та логіки бізнес-процесів.

Відповідь надсилається клієнту після того, як дані було опрацьовано. Надалі вона обробляється за допомогою Redux, а після цього відображається відповідним React-компонентом інтерфейсу.

3.3.2 Роутинг та обробка запитів

За допомогою фреймворку Express.js було забезпечено маршрутизацію у серверній частині [13]. Кожен із маршрутів відповідає за обробку конкретного виду запиту до певного ресурсу.

Роутинг у проєкті реалізовано згідно принципів REST-архітектури, використовуючи HTTP-методи, зокрема GET, POST, PUT, DELETE, для виконання базових операцій над даними.

Для усіх модулів, таких як авторизація, каталог товарів, кошик, улюблене тощо, було створено окремі маршрути.

Незалежність логіки між front-end та back-end зберігається за рахунок того, що передача інформації між клієнтом та сервером здійснюється у JSON-форматі.

Інформація про товари отримується із таблиці products, яка знаходиться у базі даних PostgreSQL. Вилучення відомостей відбувається через GET-запити, які надсилаються до маршруту /api/products (лістинг 3.10).

Лістинг 3.10 – Реалізація GET-запиту на отримання списку товарів

```
app.get('/api/products', async (req, res) => {
    try {
        const result = await db.query('SELECT * FROM products');
        res.json(result.rows);
    } catch (err) {
        console.error(err);
        res.status(500).json({ message: 'Database error' });
    }
});
```

У відповідь надходять дані, що структуровані, як масив об’єктів.

Робота кошику товарів реалізована через відповідні API-маршрути. Було створено окремий набір точок доступу /api/cart, які підтримують такі методи для управління товарними позиціями клієнта:

- GET – отримання вмісту кошика;
- POST – додавання товарів;
- PUT – оновлення кількості;
- DELETE – видалення позицій.

Система відстежує приналежність кошика конкретному користувачеві через токен авторизації, який передається у заголовках запитів.

Усі операції з кошиком зберігаються в таблиці cart, яка містить зв'язки між користувачами та позиціями на сайті через відповідні ідентифікатори.

Для додавання виробу до кошика із тіла запиту вилучаються ідентифікатор товару та бажана кількість (лістинг 3.11).

Лістинг 3.11 – Реалізація POST-запиту для керування кошиком користувача

```
app.post('/api/cart', async (req, res) => {
    const { id, quantity } = req.body;
    const token = req.headers.authorization;
    const user = await db.query('SELECT id FROM users WHERE token
= $1', [token]);
    const userId = user.rows[0].id;

    await db.query(
        `INSERT INTO cart (user_id, product_id, quantity)
         VALUES ($1, $2, $3)
         ON CONFLICT (user_id, product_id)
               DO   UPDATE   SET   quantity   =   cart.quantity   +
EXCLUDED.quantity`,
        [userId, id, quantity]
    );

    const result = await db.query(
        `SELECT p.*, c.quantity FROM cart c
         JOIN products p ON c.product_id = p.id
         WHERE c.user_id = $1`, [userId]
    );
    res.json(result.rows);
});
```

Спочатку відбувається авторизація користувача за токеном доступу через запит до бази даних, а потім вставляється новий запис до таблиці cart. Кількість певних позицій збільшується при його повторному додаванні.

Маршрут /api/rates виконує запит до зовнішнього API, після цього отримані дані обробляються та передаються клієнту у форматі JSON. Такий підхід забезпечує актуальність обмінних курсів для користувачів онлайн-магазину.

Механізм обробки помилок відбувається за рахунок конструкцій try/catch. Далі він повертає відповідні HTTP-статуси, зокрема статус 500, що означає помилку сервера, та статус 401 для помилки авторизації.

3.3.3 Робота з базою даних

У розробленому програмному забезпеченні для зберігання й управління даними використовується система PosgreSQL. Взаємодія з базою даних реалізована через спеціалізований модуль db.js, що забезпечує підключення та уніфікований інтерфейс для виконання запитів через метод db.query().

Структура бази даних складається з чотирьох основних таблиць:

- users – зберігає дані користувачів, зокрема електронну пошту, пароль, токен;
- products – містить інформацію про ювелірні вироби;
- cart – реалізує функціональність кошика через зв’язок користувача з товарами;
- favorites – зберігає улюблені товари користувачів.

Було реалізовано повний набір операцій CRUD [14].

Отримання даних відбувається через виконання SELECT-запитів, які вибирають необхідні записи з відповідної таблиці бази даних.

Таким чином було реалізовано отримання списку товарів, що забезпечило швидкий доступ до інформації (лістинг 3.12).

Лістинг 3.12 – Реалізація запиту на отримання всіх товарів

```
const result = await db.query('SELECT * FROM products');
```

Створення та оновлення реалізовано, зокрема, при додаванні товарів до кошика (лістинг 3.13).

Лістинг 3.13 – Реалізація запиту на додавання та оновлення товару

```
await db.query(
        `INSERT INTO cart (user_id, product_id, quantity)
         VALUES ($1, $2, $3)
         ON CONFLICT (user_id, product_id)
               DO UPDATE SET quantity = cart.quantity +
EXCLUDED.quantity`,
        [userId, id, quantity]
);
```

Особливістю реалізації є використання ON CONFLICT для обробки додавання нового товару і оновлення кількості існуючого в одному запиті.

Видалення даних демонструє реалізація вилучення товарів із кошика (лістинг 3.14).

Лістинг 3.14 – Реалізація запиту на видалення товару з кошика

```
await db.query('DELETE FROM cart WHERE user_id = $1 AND product_id
= $2', [userId, productId]);
```

Важливим аспектом реалізації є використання зв’язків між таблицями. Щоб отримати вміст кошика виконується певний JOIN-запит (лістинг 3.15).

Лістинг 3.15 – Реалізація запиту на отримання вмісту кошика

```
const result = await db.query(
        `SELECT p.*, c.quantity FROM cart c
         JOIN products p ON c.product_id = p.id
         WHERE c.user_id = $1`, [user.rows[0].id]
    );
```

Для отримання актуального стану кошика застосовується JOIN-оператор, який об’єднує дані з таблиць cart та products. Даний крок дозволяє у

відповіді надати клієнтській частині всю необхідну інформацію для відображення товарів, зокрема назви, ціни, кількості тощо.

Обрана архітектура взаємодії з базою даних забезпечує надійність зберігання через систему реляційних зв'язків, запобігання дублюванню записів через використання ON CONFLICT, підвищену безпеку даних завдяки параметризованим SQL-запитам та масштабованість, яка дозволяє легко розширювати функціональність.

3.3.4 Логіка авторизації та аутентифікації

У контексті розробленого вебсервісу важливо розрізняти два взаємопов'язані, але окремі процеси, а саме:

- аутентифікація – це процедура перевірки достовірності пред'явлених користувачем облікових даних;
- авторизація – це надання користувачу доступу до ресурсів системи після підтвердження його особи.

Впроваджена система зберігає інформацію про користувачів у таблиці users бази даних PostgreSQL. Таблиця містить такі поля, як електронна адреса, пароль та токен.

Процедура аутентифікації реалізована через обробку POST-запиту до маршруту /api/login (лістинг 3.16).

Лістинг 3.16 – Реалізація POST-запиту на автентифікацію користувача

```
app.post('/api/login', async (req, res) => {
    const { email, password } = req.body;
    try {
        const result = await db.query(
            'SELECT * FROM users WHERE email = $1 AND password = 
$2',
            [email, password]
        );
        if (result.rows.length > 0) {
            const user = result.rows[0];
           return res.json({ token: user.token, email: user.email
});
```

```
        }
       res.status(401).json({ message: 'Invalid email or password'
});
    } catch (err) {
        console.error(err);
        res.status(500).json({ message: 'Database error' });
    }
});
```

Після отримання паролю та електронної пошти з тіла запиту відбувається виконання параметризованого SQL-запиту для пошуку користувача з відповідними обліковими даними [15].

При знаходженні відповідного запису формується успішна відповідь з токеном та електронною поштою. При відсутності даних у таблиці повертається помилка аутентифікації із кодом 401. Реалізована також обробка серверних помилок за статусом 500.

Система використовує токен-базовану аутентифікацію, що надає користувачам безпечний доступ до ресурсу. У результаті успішної аутентифікації клієнтська частина зберігає отриманий токен у локальному сховищі браузера. Надалі він використовується для авторизації клієнта при зверненні до захищених ресурсів.

Авторизація здійснюється шляхом перевірки валідності токена, що передається у заголовку authorization. Дана перевірка наявна у обробнику доступу до кошика (лістинг 3.17).

Лістинг 3.17 – Реалізація перевірки користувача через валідацію токену

```
const token = req.headers.authorization;
    const user = await db.query('SELECT id FROM users WHERE token
= $1', [token]);
```

Аналогічний механізм авторизації застосовується до всіх захищених ресурсів системи, зокрема до функціоналу роботи з кошиком та зі списком улюблених товарів.

Для виходу із системи реалізовано спеціальний маршрут /api/logout, який обробляє певний POST-запит (лістинг 3.18).

Лістинг 3.18 – Реалізація POST-запиту на вихід із системи

```
app.post('/api/logout', (req, res) => {
    return res.json({ message: 'Logged out successfully' });
});
```

Загалом логіку виходу реалізовано на клієнтській частині, адже видаляється збережений токен з локального сховища. Проте серверна частина підтверджує успішність даної операції.

3.3.5 Email-сповіщення: замовлення та підписка

Система електронних сповіщень реалізована за допомогою бібліотеки Nodemailer для забезпечення автоматичної комунікації з користувачами. Функціонал охоплює два основних сценарії:

- підтвердження замовлення після оформлення покупки;
- підтвердження підписки на новини бренду.

Ініціалізація поштового транспорту здійснюється через конфігурацію SMTP-сервера Gmail з використанням облікових даних, які зберігаються у змінних середовища. Такий підхід забезпечує безпеку та можливість легкого переналаштування поштових параметрів (лістинг 3.19).

Лістинг 3.19 – Конфігурація поштового транспорту

```
const transporter = nodemailer.createTransport({
  host: 'smtp.gmail.com',
  port: 587,
  secure: false,
  auth: {
    user: EMAIL_USER,
    pass: EMAIL_PASSWORD,
  },
});
```

Маршрут /api/order обробляє POST-запити для відправлення електронних листів з підтвердженням замовлення. Система отримує дані про

замовлення від клієнтської частини, включаючи email користувача, повне ім’я, список товарів та загальну суму (лістинг 3.20).

Лістинг 3.20 – Обробка запиту підтвердження замовлення

```
app.post('/api/order', async (req, res) => {
  const { email, fullName, orderItems, displayTotal } = req.body;

  const message = {
    from: process.env.EMAIL_USER,
    to: email,
    subject: 'Your order at Various has been accepted!',
    text: ${fullName}, thank you for your order!\n\nThe following
items were ordered:\n${orderItems.map(item =>
      ${item.name}      (x${item.quantity})).join('\n')}\n\nTotal
amount: ${displayTotal}\n\nSoon our consultant will contact you!
  };

  try {
    await transporter.sendMail(message);
    res.json({ success: true, message: 'Email sent' });
  } catch (err) {
    //Обробка помилок
});
```

Функціонал формує персоналізоване повідомлення з деталями замовлення, включаючи список виробів та їх кількість. (рисунок 3.12).

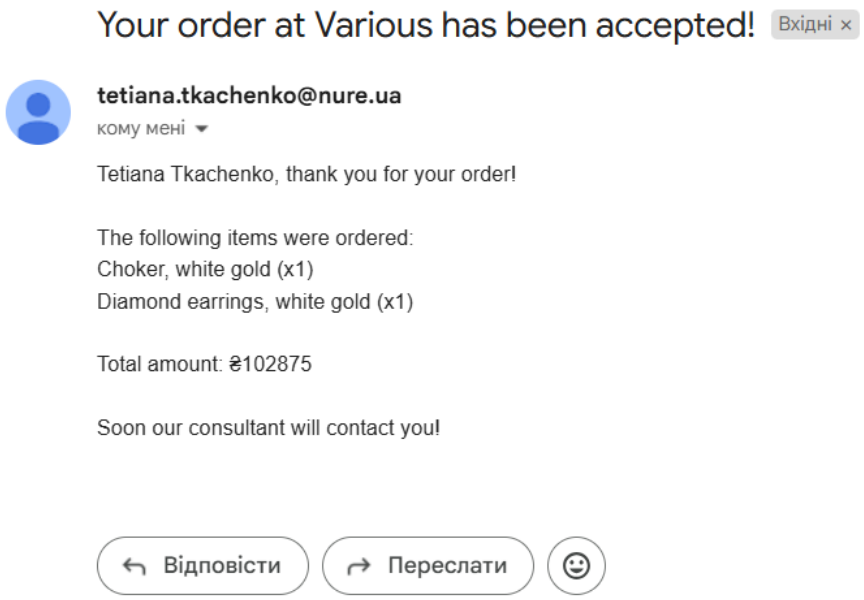

Рисунок 3.12 – Електронний лист з підтвердженням замовлення

При успішному відправленні листа сервер повертає підтвердження, у випадку помилки – відповідний статус з описом проблеми.

Система підписки реалізована через маршрут /api/subscribe, що забезпечує автоматичне надсилання підтвердження підписки користувачам на електронну адресу (лістинг 3.21).

Лістинг 3.21 – Реалізація надсилання електронного листа з підтвердженням підписки

```
app.post('/api/subscribe', async (req, res) => {
  const { email } = req.body;

  const message = {
    from: process.env.EMAIL_USER,
    to: email,
    subject: 'Thank You for Subscribing to Various 💎',
    text: `Thank you for showing interest in Various! ...
  };

  try {
    await transporter.sendMail(message);
    res.json({ success: true, message: 'Subscription email sent'
});
  } catch (err) {
    res.status(500).json({ success: false, message: 'Failed to
send subscription email' });
  }
});
```

Маршрут працює за принципом POST-запиту, отримуючи від клієнта електронну пошту через тіло запиту. Система використовує попередньо налаштований транспортер для відправки повідомлення, який базується на SMTP-протоколі.

Функціонал формує повідомлення про успішне оформлення підписки на новини магазину. (рисунок 3.13).

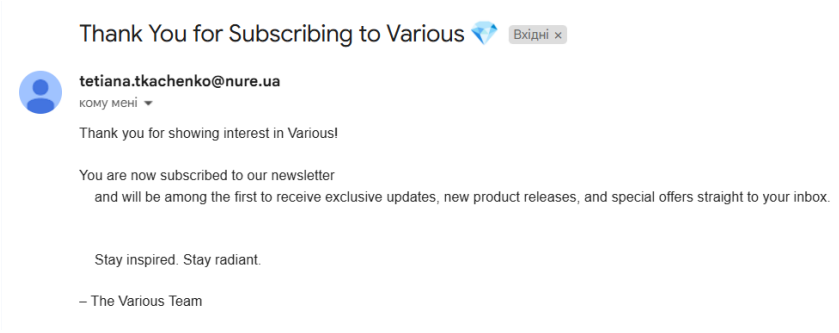

Рисунок 3.13 – Електронний лист з підтвердженням підписки

Обидва маршрути використовують асинхронну обробку для забезпечення неблокуючої роботи сервера та включають обробку помилок для підвищення надійності системи.

Відправлення електронних листів відбувається через захищене з’єднання SMTP, що гарантує безпеку передачі даних.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Ознайомлення з інтерфейсом

Онлайн-магазин ювелірних виробів складається з кількох взаємопов’язаних сторінок, кожна з яких виконує специфічні функції для забезпечення комфортних покупок коштовностей.

Головна сторінка є відправною точкою для всіх відвідувачів сайту. Вона презентує кращі ювелірні вироби, актуальні моделі та нові колекції. Сторінка призначена для ознайомлення з брендом та швидкого переходу до певних розділів.

Каталог товарів це основна робоча сторінка для пошуку та вибору ювелірних виробів. У даному розділі розміщено весь асортимент магазину з системою фільтрів за різними категоріями та характеристиками, з можливістю сортування товарів та з динамічним пошуком.

Сторінка товару надає детальну інформацію про конкретний товар, зокрема його характеристики, фото, ціну та деталі доставки, а також дає змогу обрати пакування.

Розділ акаунту об’єднує функції користувача, а саме перегляд історії замовлень, можливість виходу з акаунту тощо.

Сторінка улюбленого зберігає вироби, які відвідувач застосунку додав до списку улюблених товарів для подальшого перегляду, порівняння або покупки.

Кошик забезпечує процес оформлення замовлення з можливістю зміни кількості товарів.

Сторінка контактів містить всю необхідну інформацію для зв’язку з магазином, зокрема адреса салону та години роботи, номери телефонів та електронну пошту.

4.2 Навігація головною сторінкою

Головна сторінка містить усі необхідні елементи для комфортної навігації. (рисунок 4.1).

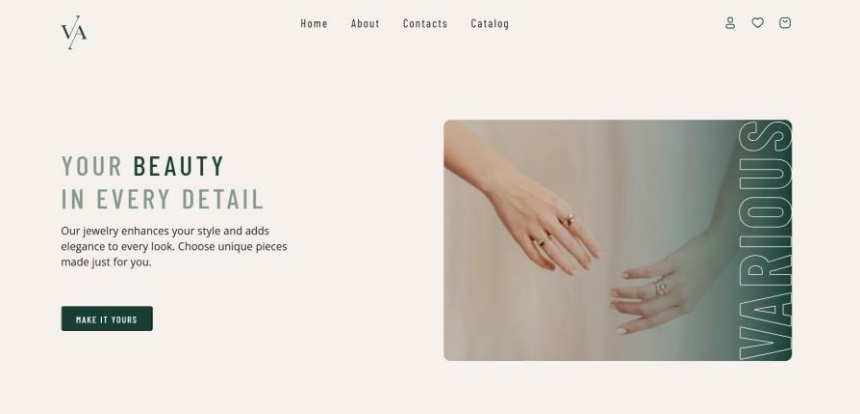

Рисунок 4.1 – Головна сторінка онлайн-магазину

У верхній лівій частині розташовано логотип, по центру є меню навігації зі сторінками «Головна», «Про бренд», «Контакти» і «Каталог» та справа знаходяться функціональні кнопки для перегляду особистого кабінету або входу чи реєстрації, улюблених товарів та кошику.

Нижче розташовано розділ з деякими виробами бренду та надана певна інформація про бренд. (рисунок 4.2).

Рисунок 4.2 – Розділ «Про бренд»

Кожен ювелірний виріб представлено у вигляді стильної картки з якісною фотографією.

Між розділами наявні надихаючі фрази та цитати, які передають мету бренду.

Наступним є розділ із популярними позиціями онлайн-магазину. Справа наявна кнопка «Побачити більше», яка надає змогу користувачу перейти до сторінки каталогу для перегляду більшої кількості товарів. (рисунок 4.3).

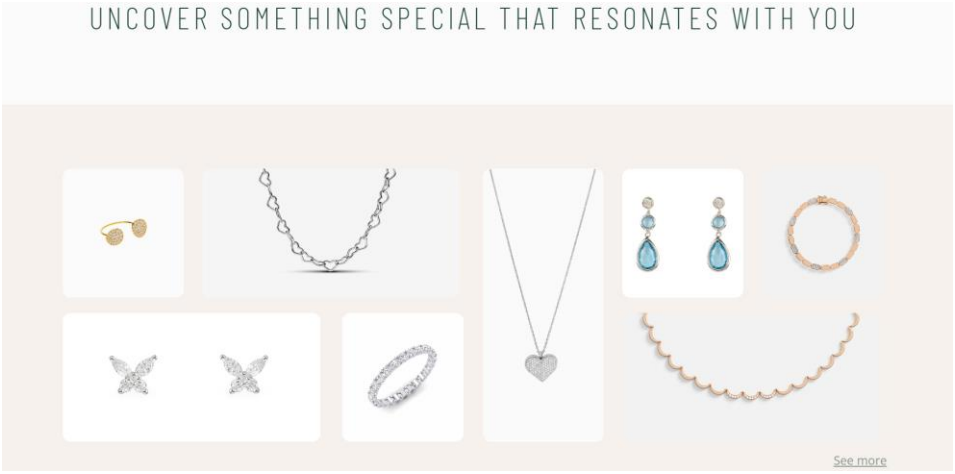

Рисунок 4.3 – Розділ з популярними виробами

В кінці даної сторінки є форма, заповнивши яку користувач може підписатися на розсилку новин про бренд, нові товари та акції. (рисунок 4.4).

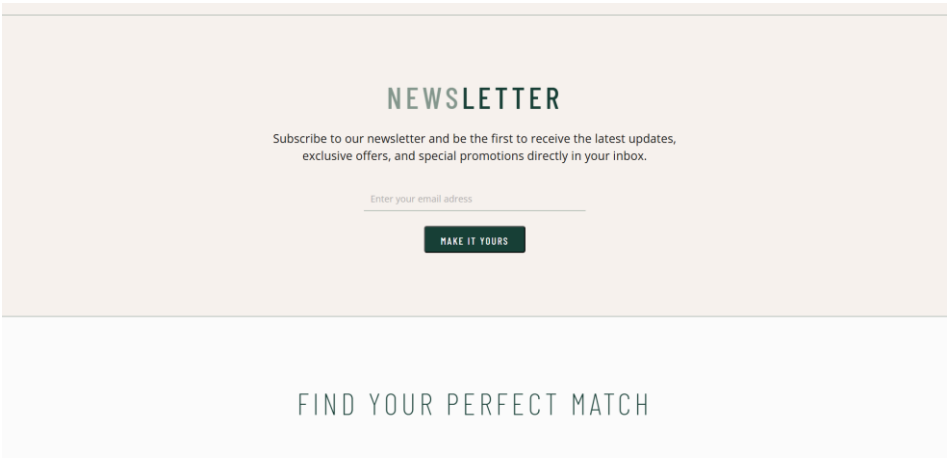

Рисунок 4.4 – Форма для підписки на розсилку про бренд

4.3 Робота з каталогом товарів

Каталог виробів є ключовою сторінкою онлайн-магазину, де представлено весь асортимент коштовностей. Каталог організовано за логічною структурою, що дозволяє знайти потрібні прикраси. (рисунок 4.5).

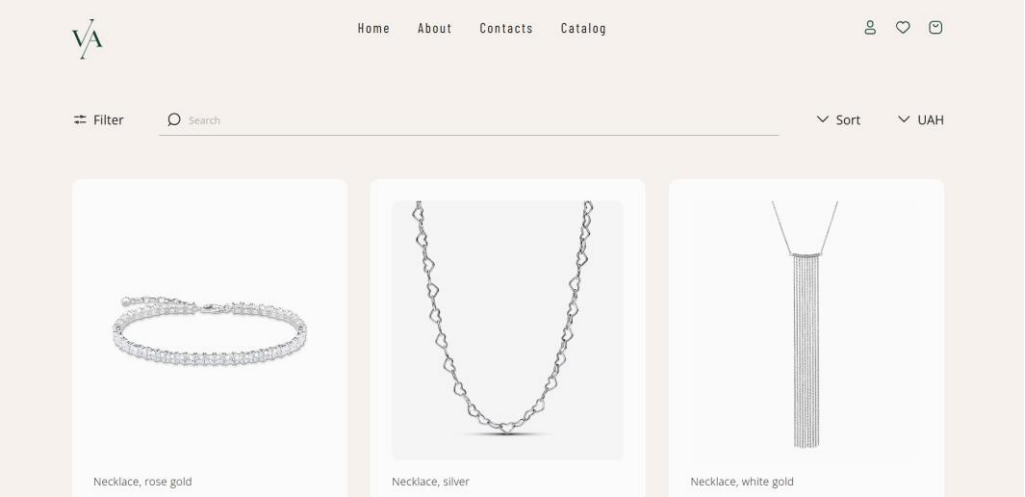

Рисунок 4.5 – Сторінка каталогу

Основним функціоналом даного розділу є динамічна фільтрація, динамічний пошук, сортування та зміна валюти.

Фільтрація розташована зліва та при натисканні на іконку з’являється відповідне модальне вікно. (рисунок 4.6).

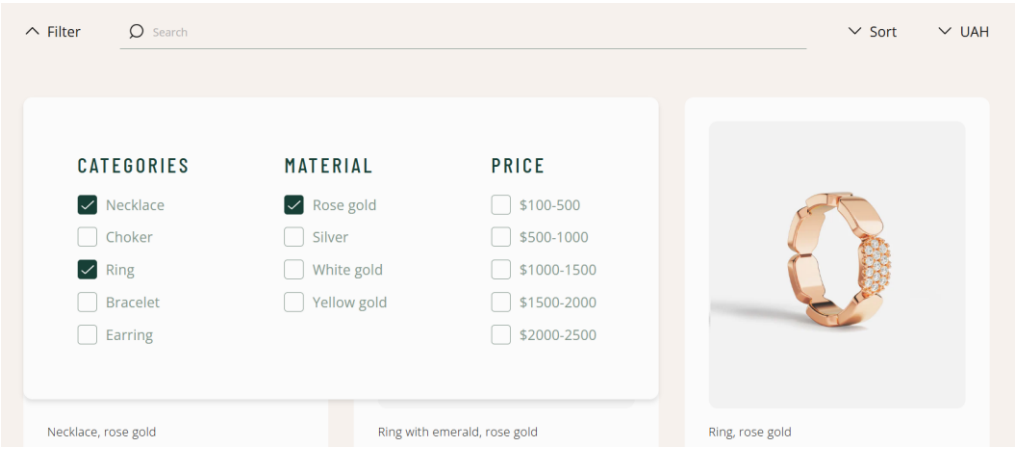

Рисунок 4.6 – Модальне вікно фільтрації

За допомогою даної функції можна шукати виріб за певними критеріями, зокрема:

- категорія виробу: кольє, чокер, кільце, браслет та сережки;
- матеріал: рожеве золото, срібло, біле золото, жовте золото;
- ціна: відповідні діапазони цін.

Для вибору певного аспекту необхідно натиснути на поле, що розташоване зліва від певної назви. Після цього відбувається динамічна фільтрація згідно обраних параметрів, яка не потребує додаткового застосування відмічених аспектів. Закриття вікна відбувається при натисканні на відповідну іконку стрілки.

Рядок, за допомогою якого можна здійснювати пошук за назвою виробу, матеріалу та використаних каменів, розташовано по центру сторінки. (рисунок 4.7).

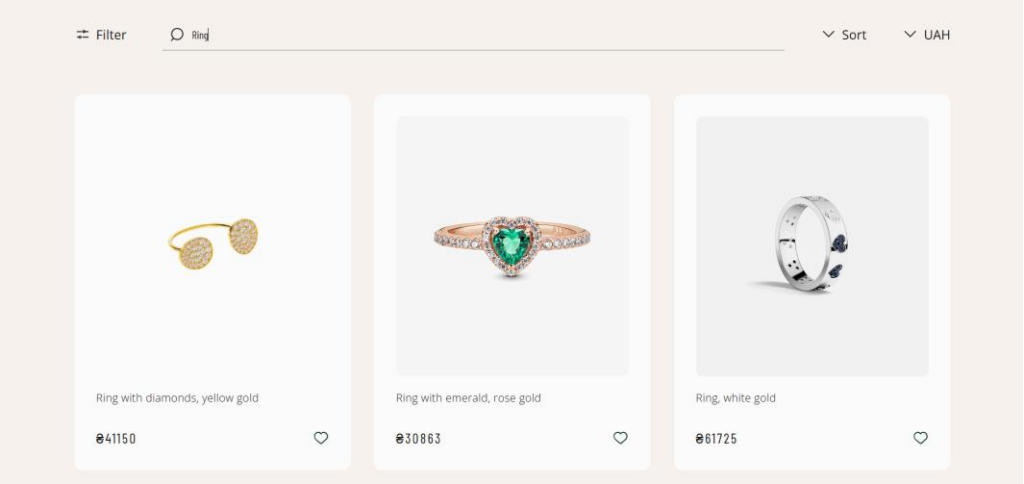

Рисунок 4.7 – Пошуковий рядок

Пошук є динамічним, тобто миттєво реагує на кожен введений символ, що дозволяє одразу відображати відповідні товари та забезпечує зручний і швидкий огляд. Даний функціонал не потребує натискання кнопок.

Також передбачена можливість сортування виробів для зручного орієнтування на сторінці каталогу. (рисунок 4.8).

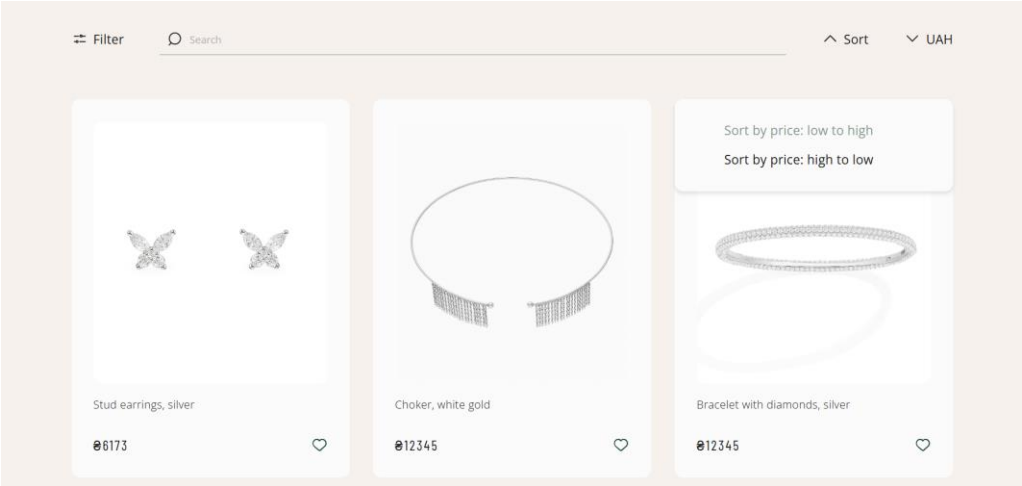

Рисунок 4.8 – Модальне вікно сортування

Функціонал дозволяє впорядковувати список за різними критеріями, зокрема:

- за зростанням ціни;
- за спаданням ціни.

Сортування реалізоване також динамічно, без необхідності оновлення сторінки чи натискання кнопок.

На сторінці передбачена можливість зміни валюти, що дозволяє користувачам переглядати ціни товарів у зручному для них форматі. Для того, щоб це зробити, необхідно натиснути на відповідну іконку справа. (рисунок 4.9).

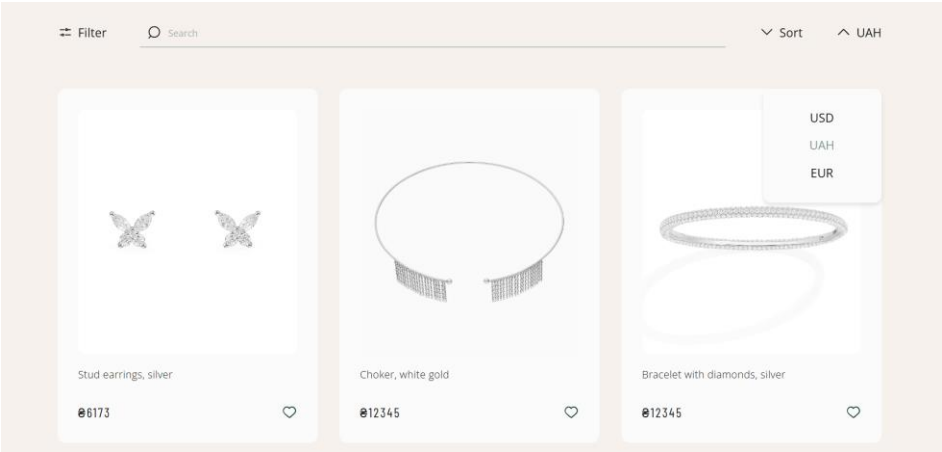

Рисунок 4.9 – Модальне вікно зміни валюти

Далі необхідно обрати потрібну грошову одиницю, після чого ціни усіх виробів на сайті, а також загальної вартості у кошику буде перераховано згідно актуального курсу.

Зі сторінки каталогу користувач може перейти на сторінку окремого товару, натиснувши на картку певного виробу, а також може додати його до улюбленого, аби переглянути його потім чи додати у кошик.

4.4 Створення облікового запису та авторизація

Для здійснення покупок ювелірних виробів та повного використання функціоналу магазину необхідно мати особистий обліковий запис.

Якщо він вже є, тоді необхідно ввести дані для входу, зокрема електронну пошту та пароль, у відповідні поля форми. (рисунок 4.10).

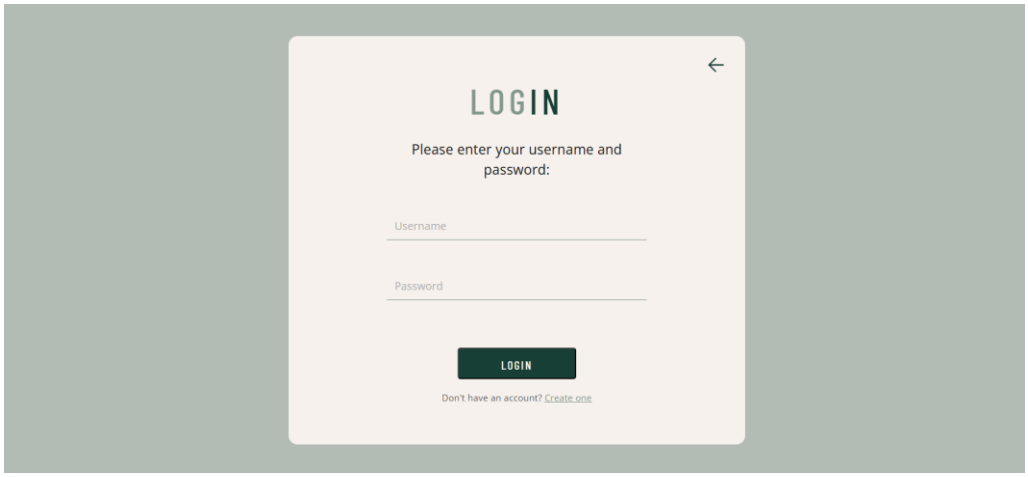

Рисунок 4.10 – Вікно входу в особистий акаунт

При правильному введенні даних здійсниться вхід в акаунт та перехід на сторінку каталогу для пошуку та вибору прикрас.

У разі коли користувач не має облікового запису, то необхідно його створити. У формі входу натиснувши на фразу «Створити акаунт» буде відкрито сторінку реєстрації. (рисунок 4.11).

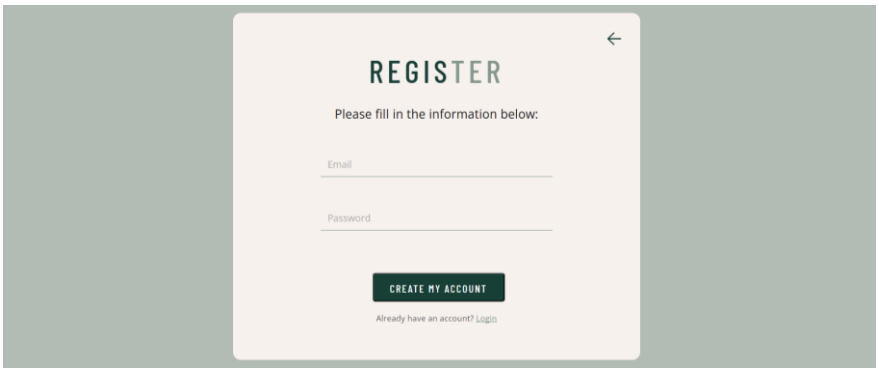

Рисунок 4.11 – Вікно створення облікового запису

Необхідно заповнити два поля:

- електронна пошта – дійсна адреса електронної пошти, яка буде використовуватися для входу в систему;
- пароль – надійний набір символів для захисту акаунту.

Після цього необхідно натиснути на кнопку «Створити акаунт» і при успішній реєстрації буде виконано перехід на сторінку каталогу.

4.5 Взаємодія зі сторінкою акаунту

Сторінка акаунту є центральним місцем управління особистою інформацією клієнта та відстеження історії його покупок. Вона надає повний огляд всіх здійснених замовлень та їх поточного статусу. (рисунок 4.12).

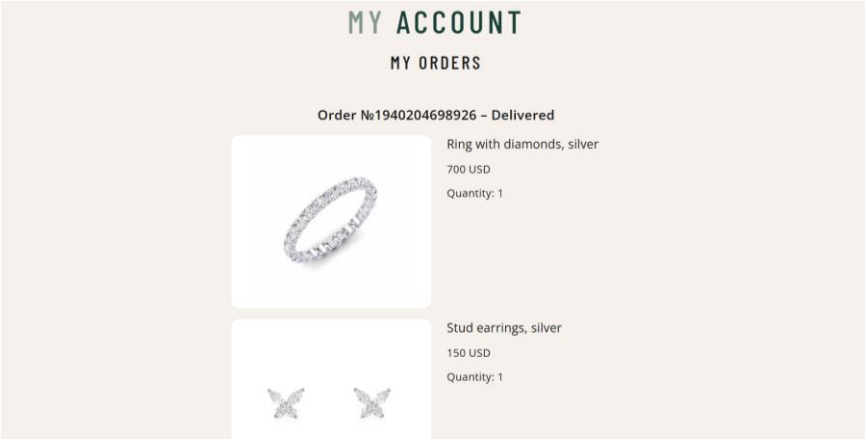

Рисунок 4.12 – Сторінка особистого акаунту

Замовлення класифікуються за двома основними статусами:

1) доставлені замовлення – ті, які були успішно завершені та доставлені замовнику, відображаються зі статусом «Доставлено»;

2) замовлення в обробці – такі, що були щойно оформлені через сторінку кошику та додані зі статусом «Обробляється».

Кожне замовлення містить детальну інформацію для відстеження, зокрема унікальний номер замовлення, інформацію щодо назви виробу, його ціни, кількості та зображення.

Також у даному розділі розміщена опція для безпечного завершення сесії відвідувача сайту. (рисунок 4.13).

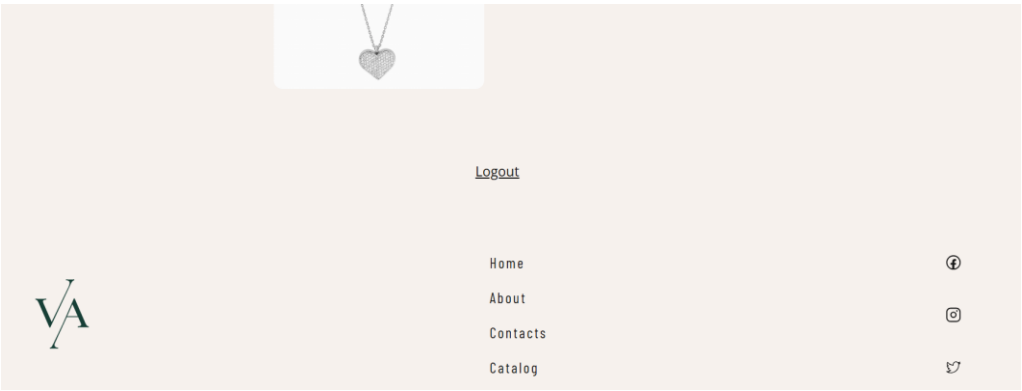

Рисунок 4.13 – Кнопка для вихоуду з акаунту

4.6 Перегляд і додавання товару

Перегляд детальної інформації про товар та його додавання до улюбленого або кошику здійснюється на спеціалізованій окремій сторінці товару, яка представляє інтерфейс для взаємодії з певним ювелірним виробом.

Даний розділ забезпечує повний огляд характеристик ювелірного виробу. (рисунок 4.14).

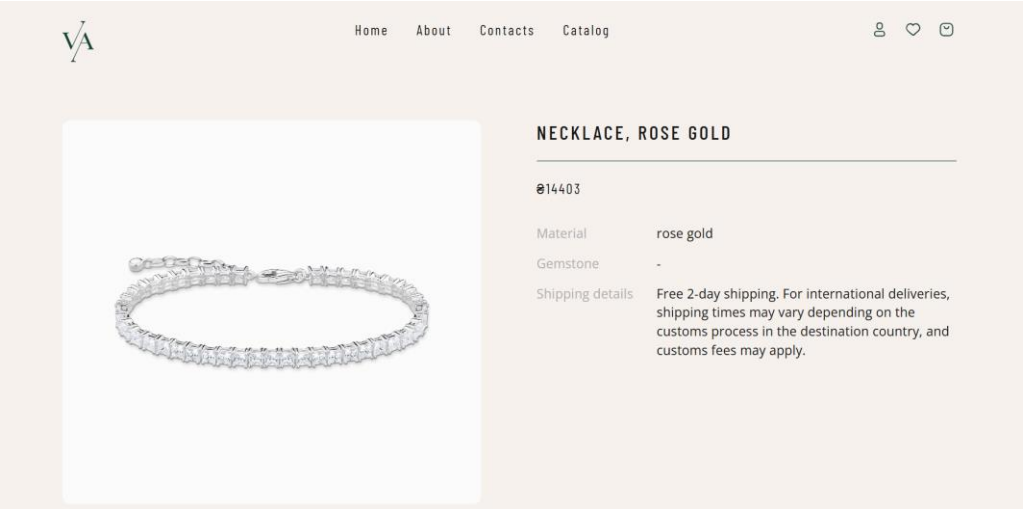

Рисунок 4.14 – Сторінка окремого товару

Перехід на сторінку здійснюється з каталогу продукції. Клієнт може обрати будь-який виріб із представленого асортименту, після чого автоматично відкривається детальна сторінка обраного товару.

Інформація містить такі відомості, як назва виробу, ціна, матеріал, використане коштовне каміння, деталі доставки. Фото товарів є якісними і гарно зображують кожен елемент. Дані про терміни доставки допомагають планувати отримання замовлення.

На сторінці товару користувач має можливість обрати спеціальне пакування для ювелірного виробу. (рисунок 4.15).

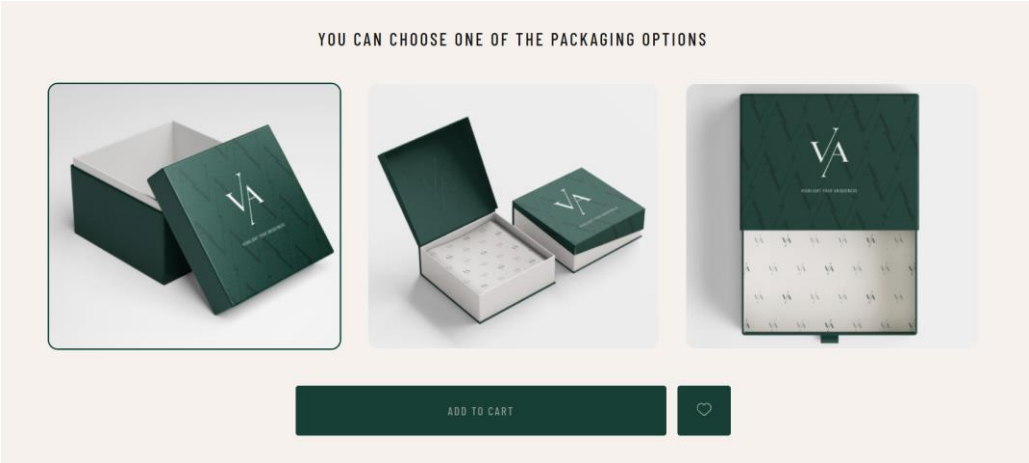

Рисунок 4.15 – Вибір пакування ювелірного виробу

Після ознайомлення з характеристиками та вибором відповідного пакування користувач можі здійснити одну з двох дій:

- Додати в улюблене та зберегти товар для майбутнього розгляду;
- Додати у кошик та почати процес оформлення замовлення.

4.7 Робота з улюбленими товарами

Сторінка улюблених товарів є персональним простором клієнта, де зберігаються всі вироби, які були відмічені як улюблені під час перегляду каталогу або сторінок окремих товарів.

У даному розділі вироби відображаються у зручному форматі з основною інформацією, необхідною для швидкого прийняття рішення. (рисунок 4.16).

Для кожного з товарів відображається:

- фото виробу;
- назва прикраси;
- ціна.

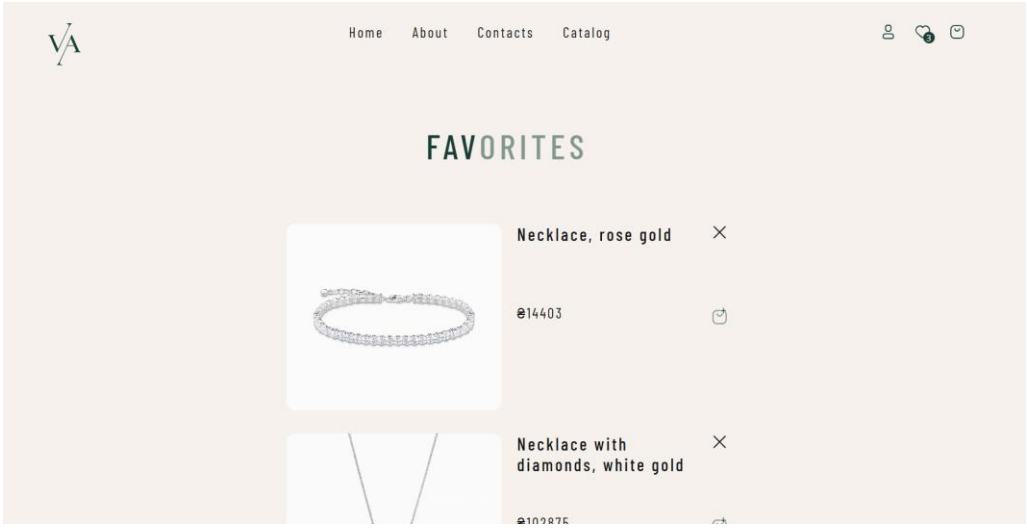

Рисунок 4.16 – Сторінка з улюбленими товарами

З картками прикрас доступні дві дії:

- видалення виробу з улюбленого: це можна зробити натиснувши на хрестик у правому верхньому кутку картки товару;
- додавання виробу у кошик: дія виконується при натисканні на іконку кошику з правого боку.

У нижній частині сторінки розміщена кнопка «Додати усе до кошику», які надає зручну можливість одночасного перенесення всіх улюблених товарів до корзини. (рисунок 4.17).

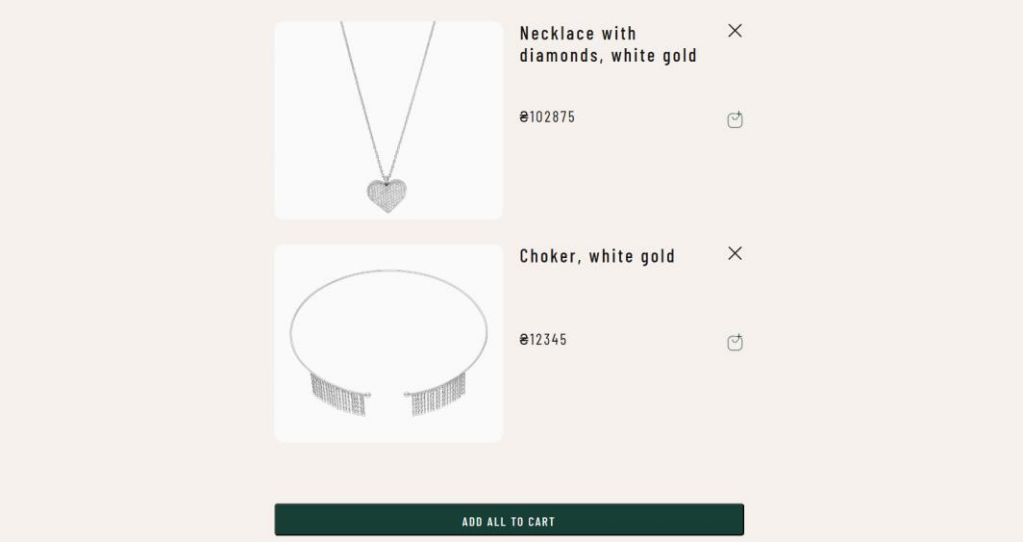

Рисунок 4.17 – Кнопка для додавання усіх позицій до кошику

4.8 Оформлення замовлення через кошик

Сторінка кошику є завершальним етапом процесу покупки, де користувач може переглянути всі обрані товари, відкоригувати їх кількість та оформити остаточне замовлення. Вона поєднує функції управління товарами з процесом збору необхідної інформації для доставки та зв’язку. (рисунок 4.18).

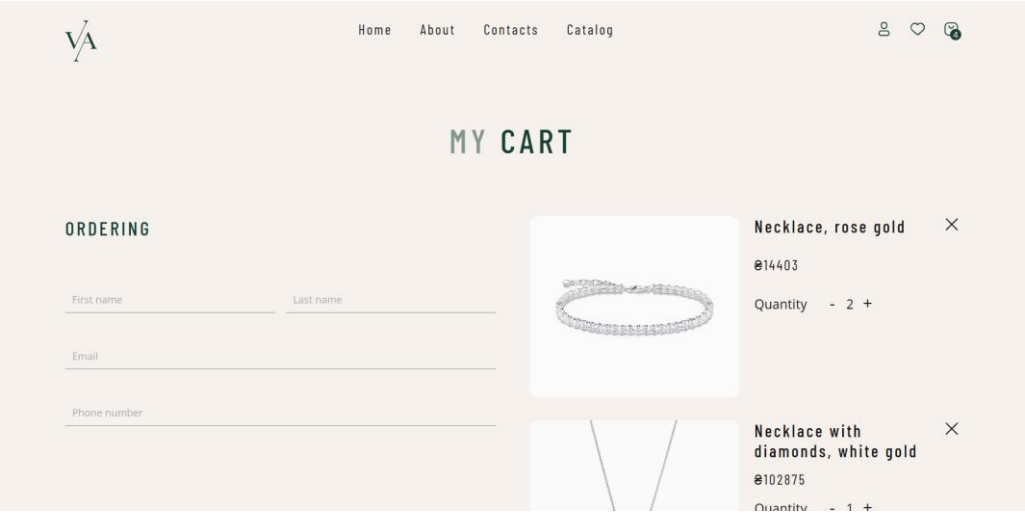

Рисунок 4.18 – Сторінка кошику

Справа представлені всі товари, що було додані для покупки. Кожен з них відображається з повною інформацією, зокрема назва товару, фото, його ціна та кількість, яку можна збільшувати або зменшувати.

Також є можливість видалення виробу зі списку, натиснувши на хрестик зверху картки товару.

Для оформлення замовлення користувач повинен заповнити обов’язкову форму з контактними даними:

- ім’я та прізвище: для персоналізації обслуговування;
- електронна пошта: для відправки підтвердження замовлення та подальшого зв’язку;
- номер телефону: для оперативного зв’язку з консультантом.

Після заповнення всіх необхідних полів для підтвердження замовлення користувач натискає кнопку «Оформити замовлення», що ініціює процес обробки замовлення системою. Одразу з’являється модальне вікно з повідомленням про успішну операцію та інформацією про подальші кроки.

Одночасно з появою модального вікна на вказану електронну пошту надсилається детальне повідомлення, що містить перелік товарів, їх кількість та загальну суму, а також те, що консультант незабаром зв’яжеться з замовником.

4.9 Отримання контактної інформації

Сторінка контактів надає усі шляхи зв’язку з компанією, зокрема електронну пошту, номери телефонів та адрес компанії з годинами роботи. (рисунок 4.19).

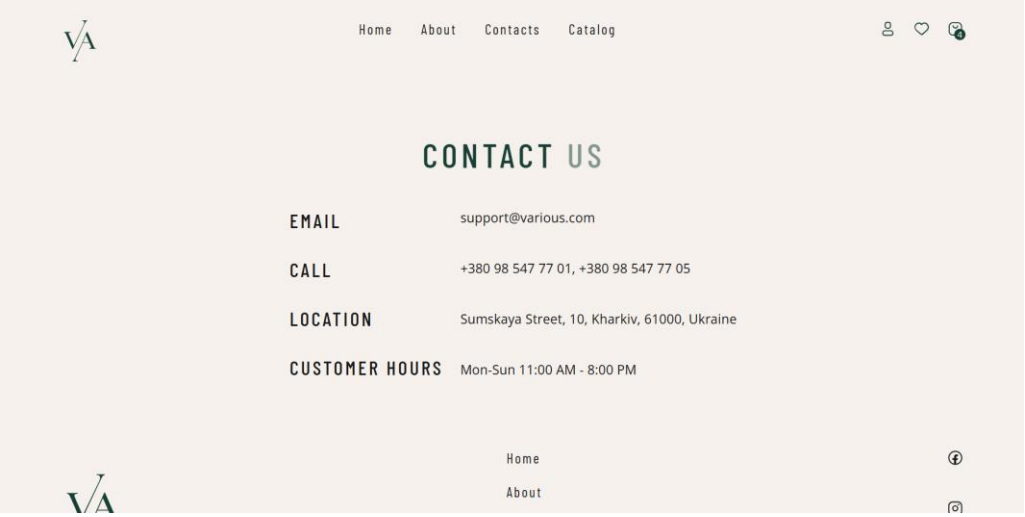

Рисунок 4.19 – Сторінка з контактними даними

За допомогою даної інформації можна здійснювати консультації з технічної підтримки та загальних запитів клієнтів, оперативне вирішення питань та особисті відвідування.

ВИСНОВКИ

Кваліфікаційна робота була присвячена розробці актуального, інтуїтивно зрозумілого для користувача, функціонального та конкурентоспроможного вебзастосунку, що буде реалізований як клієнт-серверний сервіс для онлайн-продажу ювелірних виробів.

Аналіз існуючих платформ, які спеціалізуються на торгівлі прикрасами, показав тенденції у дизайні інтерфейсу, функціональних особливостях та механізмах взаємодії з клієнтами. На основі отриманих висновків сформовано ключові вимоги до майбутнього проєкту та створено логіку взаємодії із застосунком через sitemap і user flow.

У результаті розробки створено повнофункціональний клієнт-серверний застосунок із сучасним дизайном та інтуїтивним інтерфейсом.

Клієнтська частина реалізована на базі React з використанням Redux для того, щоб мати змогу оптимального керування загальним станом програми, що забезпечує швидку роботу з динамічною фільтрацією, сортуванням, пошуком товарів. Система підтримує можливість зміни валюти з автоматичним перерахуванням цін згідно актуального курсу через API стороннього ресурсу, що робить платформу зручною для міжнародних користувачів.

Серверна частина на Node.js та Express забезпечує REST API для ефективної обробки запитів клієнтів. PostgreSQL зберігає деталі про товари, замовлення, вміст кошика та позиції в улюбленому. Реалізована безпечна система авторизації з токенами, які зберігаються у локальному сховищі браузера для підтримки сесій.

Створений клієнт-серверний застосунок поєднує продуктивність, сучасність та функціональність і містить повний цикл онлайн-торгівлі ювелірними виробами. Платформа пропонує зручний інтерфейс для перегляду каталогу, детальної інформації про товари, керування кошиком та улюбленими позиціями, а також підтримку різних грошових одиниць.

Для подальшого розвитку проєкту можна розширити функціональність платформи шляхом додавання більшої кількості валют для міжнародних користувачів та впровадження підтримки кількох мов. Також доцільно розширити можливості сортування виробів, включивши критерії за популярністю серед клієнтів і новизною надходжень до каталогу, для підвищення зручності пошуку та вибору ювелірних виробів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. E-commerce statistics for individuals. URL: https://ec.europa.eu/eurostat/statistics-explained/index.php?title=E-commerce_statistics_for_individuals.

2. Eugene Fedorenko. Designing in Figma: The complete guide to designing with reusable components and styles in Figma, 2020 – 164 с. – ISBN-13 979-8682918232.

3. Andrew Faulkner, Conrad Chavez. Adobe Photoshop Classroom in a Book, 1st Edition, 2020. – 416 с. – ISBN-13 978-0136447993.

4. Kameron Hussain, Frahaan Hussain. Mastering Visual Studio Code: Navigating the Future of Development, 1st Edition, 2024. – 232 с. – ISBN-13 979-8877664975.

5. Michele Bertoli. React Design Patterns and Best Practices: Build easy to scale modular applications using the most powerful components and design patterns, 2017. – 318 с. – ISBN-13 978-1786464538.

6. Marijn Haverbeke. Eloquent JavaScript, 4th Edition: A Modern Introduction to Programming, 2024. – 456 с. – ISBN-13 978-1718504103.

7. Thomas A. Powell. HTML & CSS: The Complete Reference, 5th Edition, 2009. – 864 с. – ISBN-13 978-0071496292.

8. Yamini Panchal, Ravi Kumar Gupta. Building Scalable Web Apps with Node.js and Express: Design and Develop a Robust, Scalable, High-Performance Web Application Using Node.js, Express.js, TypeScript, and Redis, 2024. – 389 с. – ISBN-13 978-8197223815.

9. Leonard Richardson, Amundsen Mike, Sam Ruby. Restful Web APIs: Services for a Changing World, 2013. – 404 с. – ISBN-13 978-1449358068.

10. Henrietta Dombrovskaya, Boris Novikov, Anna Bailliekova. PostgreSQL Query Optimization: The Ultimate Guide to Building Efficient Queries, 1th Edition, 2021. – 344 с. – ISBN-13 978-1484268841.

11. Nodemailer. [Електронний ресурс]. – Режим доступу : https://nodemailer.com.

12. Model-View-Controller. [Електронний ресурс]. – Режим доступу : https://developer.mozilla.org/en-US/docs/Glossary/MVC.

13. Express/Node introduction. [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/Learn_web_development /Extensions/Server-side/Express_Nodejs/Introduction.

14. CRUD (Create, Read, Update, Delete). [Електронний ресурс]. – Режим доступу : https://learn.microsoft.com/en-us/iis-administration/api/crud.

15. Parameterized Queries JavaScript Guide. [Електронний ресурс]. – Режим доступу : https://medium.com/@ajay.monga73/parameterized-queries-javascript-guide-how-to-prevent-sql-injection-with-parameterized-queries-18c5cbbfffe2.