МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНТЕРНЕТ-МАГАЗИНУ В'ЯЗАНИХ ІГРАШОК ЗАСОБАМИ JAVASCRIPT ТА MONGODB»**

Виконала: студентка 4 курсу, групи 6.1210-1пі
спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)
освітньої програми програмна інженерія
(назва освітньої програми)
Д.О. Васильєва
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, PhD, Столярова А.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри комп'ютерних наук, к.т.н. Решевська К.С.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти бакалавр

Спеціальність 121 інженерія програмного забезпечення
(шифр і назва)

Освітня програма програмна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.
(підпис)

“ ____ ” _____ 2023 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТЦІ

Васильєвій Дарії Олексіївні
(прізвище, ім’я та по-батькові)

1. Тема роботи Розробка Інтернет-магазину в’язаних іграшок засобами JavaScript та MongoDB

керівник роботи Столярова Анастасія Валеріївна, PhD
(прізвище, ім’я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 21 » грудня 2023 року № 2180-с

2. Строк подання студентом роботи 03.06.2024 р.

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Проєктування інформаційної системи.
4. Особливості реалізації та результати тестування.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
презентація за темою доповіді

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання 25.12.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану роботи.	09.01.2024	
2.	Збір вихідних даних.	30.01.2024	
3.	Обробка методичних та теоретичних	27.02.2024	
	джерел.		
4.	Розробка першого та другого розділу.	23.04.2024	
5.	Розробка третього розділу.	20.05.2024	
6.	Оформлення та нормоконтроль	27.05.2024	
	кваліфікаційної роботи бакалавра.		
7.	Захист кваліфікаційної роботи.	16.06.2024	

Студент _____ Д.О. Васильєва
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Столярова
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка Інтернет-магазину в’язаних іграшок засобами JavaScript та MongoDB»: 45 с., 25 рис., 22 джерела, 1 додаток.

БАЗА ДАНИХ, ІНТЕРНЕТ-МАГАЗИН, КОРИСТУВАЧ, ТЕСТУВАННЯ, ФРЕЙМВОРК, CSS, EXPRESS, HTML, JAVASCRIPT, MONGODB, NODEJS, REACT.

Об’єкт дослідження – процес розробки Інтернет-магазину.

Предмет дослідження – мова програмування JavaScript, база даних MongoDB, взаємодія між ними.

Мета роботи: спроєктувати та розробити сайт для онлайн-магазину в’язаних іграшок з використанням фреймворків для клієнтської частини React та серверної частини Node.js на основі системи керування базами даних MongoDB.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

При розробці додатку було проведено аналіз предметної області, огляд аналогічних проєктів, а також опис інструментів для розробки інформаційних систем, аналіз використаних технологій. Розроблено структуру програми та діаграми для ілюстрації класів, послідовностей та прецедентів застосунку. Проведено проєктування бази даних і проілюстровано ER-діаграмою.

Реалізовано клієнтську і серверну частини з подальшим тестуванням для завершення процесу розробки. Серверну частину вебсайту було розроблено за допомогою фреймворку Express Node.js, клієнтська частина розроблена з використанням фреймворку React. Проведено ручне і автоматичне тестування інформаційної системи.

SUMMARY

Bachelor's qualifying paper "Development of the Knitted Toys Online Store Using JavaScript and MongoDB": 45 pages, 25 figures, 22 references, 1 supplement.

CSS, DATABASE, EXPRESS, FRAMEWORK, HTML, JAVASCRIPT, MONGODB, NODEJS, ONLINE SHOP, REACT, TESTING, USER.

The object of the study is the process of developing an online store.

The subject of the study is JavaScript programming language, MongoDB database, and their interaction.

The aim of the study is to design and develop a website for an online knitted toys store using the React framework for the client-side and Node.js framework for the server-side, based on the MongoDB database management system.

The methods of research are methods for collecting and analyzing software requirements, methods for modeling, designing, constructing, and testing software.

During the development of the application, an analysis of the domain area, a review of similar projects, and a description of tools for developing information systems were conducted. The technologies used were analyzed. The program structure and diagrams illustrating the classes, sequences, and use cases of the application were developed. The database was designed and illustrated with an ER diagram.

The client-side and server-side were implemented with subsequent testing to complete the development process. The server-side of the website was developed using the Express Node.js framework, and the client-side was developed using the React framework. Manual and automated testing of the information system was carried out.

ЗМІСТ

ВСТУП

Крокуючи вперед, технології значно полегшують життя людства протягом останніх років. Невід'ємним аспектом сучасного суспільства стали онлайн-покупки. Сидячи вдома чи в будь-якому іншому зручному місці, людина може зайти на сайт потрібного їй Інтернет-магазину, обрати товар і оформити замовлення лише в декілька кліків мишею. Значний стрибок популярності онлайн-шопінгу відбувся у 2020 році під час пандемії, але і до цього людство почало надавати перевагу купівлі в декілька кліків мишею замість звичних походів по магазинам. Наведемо переваги Інтернет-покупок [8]:

- здійснення покупок з будь-якої точки Землі в будь-який час доби без потреби фізично перебувати в магазині;
- можливість замовляти товари з-за кордону, що тягне за собою більший асортимент;
- можливість ознайомлення з відгуками інших клієнтів.

Разом із тим є певні недоліки, наприклад:

- неможливість оглянути товар до його безпосередньої доставки, внаслідок чого може не підійти матеріал, розмір чи інші властивості, і це, у свою чергу, призводить до незадоволеності клієнта;
- потенційні проблеми з доставкою, особливо для замовлень з-за кордону.

Для онлайн-шопінгу створюються різноманітні Інтернет-сервіси, як із широким асортиментом, так і більш вузької спеціалізації, до якої належить в'язання. Ручна робота на сьогодні є незамінною жодними технологіями, і окремих сервісів для купівлі в'язаних іграшок практично не існує. Змоделюємо ситуацію, коли користувач шукає в Інтернеті в'язані іграшки. Серед переліку сайтів, що повертає пошук, в основному магазини з фурнітурою та матеріалами для ручної роботи, магазини з в'язаним одягом (і

ті здебільшого для жінок). Відсутністю Інтернет-магазинів із таким асортиментом і обумовлена актуальність даного проєкту. Розробка подібного сервісу може бути вигідною для потенційного клієнта-замовника.

Метою даного проєкту є проєктування та розробка сайту для онлайн-магазину в'язаних іграшок з використанням фреймворків для клієнтської частини React та серверної частини Node.js на основі системи керування базами даних MongoDB.

Досягнення поставленої мети передбачає реалізацію наступних завдань:

- проаналізувати предметну область, схожі проєкти в Інтернет-мережі, інструменти та технології для досягнення поставленої мети;
- спроєктувати базу даних, структуру програми;
- розробити структуру програми, діаграми для ілюстрації потрібних діаграм (а саме діаграми класів, послідовностей і прецедентів), безпосередньо сам код програми;
- протестувати за допомогою ручного тестування розроблену систему.

Об'єкт дослідження – процес розробки Інтернет-магазину.

Предмет дослідження – мова програмування JavaScript, база даних MongoDB, взаємодія між ними.

Методи дослідження – методи збору та аналізу вимог до програмного забезпечення, методи моделювання, проєктування, конструювання та тестування програмного забезпечення.

Для досягнення результату було обрано мову JavaScript, оскільки за допомогою неї можна легко реалізувати серверну і клієнтську частину та взаємодію між ними. Для зберігання інформації про товари було обрано NoSQL систему управління базою даних MongoDB. Дані в ній зберігаються у форматі JSON, а, отже, інформація є не такою строго структурованою як у SQL системах управління базами даних. Це робить роботу з базами даних більш легкою та зручною.

Основні положення і результати кваліфікаційної роботи доповідались, обговорювалися та знайшли схвалення на двох науково-практичних

конференціях: XVII університетській науково-практичній конференції студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» (Запоріжжя, 17-22 квітня 2024 р.) та П’ятнадцятій Всеукраїнській, Двадцять другій регіональній науковій конференції молодих дослідників «Актуальні проблеми математики та інформатики» (Запоріжжя, 25-26 квітня 2024 р.), за результатами яких опубліковано тези.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, переліку посилань та одного додатку. Загальний обсяг роботи становить 45 сторінок. Додатки викладено на 4 сторінках. У переліку посилань 22 використаних джерела.

1 АНАЛІЗ ЗАСОБІВ РЕАЛІЗАЦІЇ ПРОЄКТУ

У цьому розділі описано технології, за допомогою яких реалізовано запланований проєкт. Вибір нижче описаних засобів обумовлений їхньою зручністю у використанні та поєднанням між собою, що було виявлено у ході аналізу та порівняння певних мов програмування, конкретних фреймворків тощо.

1.1 Мова програмування JavaScript: поняття, застосування, переваги та недоліки

JavaScript – це мова програмування, яка забезпечує реалізацію різноманітного функціоналу у вебдодатках. Вона додає вебсторінці динамічності, можливості додавати чи змінювати контент, відтворювати анімації, отримувати дані зі сторонніх сервісів. Так, наприклад, за допомогою програм, написаних на JavaScript, тобто скриптів, можна додати на сторінку багато чого, починаючи від банальної обробки натискань на кнопку, відправки даних із форми до отримання інформації з бази даних чи певного серверу, та і це далеко не всі можливості JavaScript.

Різні дослідження та опитування доводять, що JavaScript займає перші місця серед найпопулярніших мов програмування у світі [14, 18, 19]. Така її позиція має обґрунтування, тому перейдемо до переліку переваг мови JavaScript.

По-перше, JavaScript має легкий синтаксис, порівняно з іншими мовами програмування, що робить її значно простішою для вивчення. По-друге, існує дуже велика кількість фреймворків, які підтримуються даною мовою програмування, що урізноманітнює функціонал і додає багато нових можливостей. По-третє, для виконання коду не потрібно встановлювати

компілятор або що – вистачить лише браузеру. До того ж JavaScript підтримується всіма сучасними браузерами та вона автоматично увімкнена, тож для виконання скриптів не потрібно прикладати додаткових зусиль. Виокремимо той факт, що JavaScript легко інтегрується з HTML та CSS [3].

Серед недоліків можна виділити те, що змінні не є строго типізованими, що може спричинити певні помилки та труднощі. Окрім цього, у різних браузерах встановлені різні рушії JavaScript, тож і скрипт може бути інтерпретований по-різному. Насамкінець, одна помилка в коді може призвести до зупинки всієї сторінки.

На ринку праці великий попит на розробників саме JavaScript, адже цією мовою можна розробити багато різних унікальних проєктів, не тільки вебсайтів, а і мобільних додатків чи серверів.

1.2 Система керування базами даних MongoDB

Системи керування базами даних поділяються на SQL та noSQL. Різниця в тому, яку мову вони використовують для запису даних: SQL використовує однойменну мову SQL (Structured Query Language – тобто мова структурованих запитів), а noSQL використовує JSON, XML, YAML тощо [17].

«MongoDB – найпопулярніша NoSQL документо-орієнтована система управління базами даних» [9]. Ця система управління базами даних (далі – СУБД) зберігає інформацію у json-документах. Тож колекції у базі даних (далі – БД) мають певну структуру, але гнучкіші за реляційну БД.

Принцип роботи досить простий. БД у MongoDB складається з колекцій, які у свою чергу складаються з документів. Кожен документ містить записи за принципом ключ-значення. Документ можна змінювати, додаючи, видаляючи чи редагуючи записи [11].

Із значущих переваг MongoDB можна визначити:

- легкість у вивченні та розумінні;
- документо-орієнтованість;
- швидкість у розробці;
- масштабованість [9].

1.3 Платформа Node.js та фреймворк Express

Для роботи з JavaScript існують різні додаткові засоби з широким функціоналом. Так, наприклад, Node.js – це платформа, яка забезпечує серверне обслуговування сайту, запуск вебсторінок на сервері тощо. Платформа користується великою популярністю серед розробників, і аргументацією цьому є ряд причин.

По-перше, за допомогою Node.js можна реалізувати велику кількість різнопланових додатків: від односторінкових сайтів і мікросервісних додатків до застосунків для спілкування та співпраці в реальному часі. По-друге, можна забезпечити і клієнтську, і серверну частини сайту з використанням JavaScript. По-третє, так як Node.js використовує JavaScript, то і його використання є простим і зрозумілим. По-четверте, платформа є дуже популярною, тож і спільнота користувачів є великою та активною, отже, всі питання можуть активно обговорюватися на форумах і легко можна знайти рішення своєї проблеми [22].

Серед реальних додатків, що розроблені на основі вищезазначеної платформи, можна виділити такі популярні: Netflix, LinkedIn, NASA, Walmart [22].

Для роботи з сервером недостатньо самої платформи Node.js. Для полегшення розробки та забезпечення додаткових функцій знадобиться фреймворк Express. Цей фреймворк використовується для забезпечення зв’язку з БД MongoDB, обробки запитів на зразок get, put, post, delete, update тощо [7].

1.4 Фреймворк React

Стандартний сайт складається з html-сторінок із контентом, css-файлів для стилізації сторінок та js-скриптів для анімацій і примітивних обробок кнопок, введення даних тощо. Як зазначалося вище, для зберігання інформації може використовуватися БД, а для взаємодії з обраною БД і запуском на сервері використовуються специфічні платформи та фреймворки. Аналогічно і з розробкою самого сайту: для структурованості сторінок, їхньої динамічності та інтерактивності можна використовувати фреймворк React.

React – це фреймворк, що базується на створенні компонентів і їхньому поєднанні. Так, скажімо, розробник може створити окремо компонент із навігацією, із шапкою, підвалом сайту та окремо з головним вмістом кожної сторінки [16].

Для проєкту використовуватиметься синтаксис JSX. Його зручність полягає в поєднанні JavaScript та HTML. Тобто він базується на JavaScript, але при цьому дозволяє писати html-теги, які коректно відображатимуться на вебсторінці. Окрім розділення на компоненти, це забезпечує краще розуміння того, як змінюватиметься інтерфейс сайту з обробкою подій [2].

1.5 Висновки до розділу 1

Отже, у результаті проведеного аналізу існуючих технологій для розробки вебсайтів було обрано конкретні засоби, а саме JavaScript, СУБД MongoDB, платформу Node.js та конкретний фреймворк Express для розробки серверної частини, фреймворк React для забезпечення роботи клієнтської частини. Зокрема, було наведено їх переваги, серед яких зазначаються універсальність мови програмування JavaScript та фреймворків, гнучкість MongoDB, зручність у використанні всіх цих засобів. Також було виявлено той факт, що всі зазначені технології бездоганно поєднуються між собою та використовуються саме з метою розробки вебдодатків.

2 ПРОЄКТУВАННЯ ТА РОЗРОБКА ПРОДУКТУ

Для вдалої розробки цілісного продукту важливо спланувати каркас конструкції, що включає в себе базу даних, структуру сайту, взаємодію між користувачем і системою. Розглядатиметься проєктування бази даних та конкретних діаграм взаємодії: діаграми прецедентів, класів і послідовностей.

2.1 Проєктування бази даних

Ключовою частиною сайту є база даних, тому перед початком роботи її потрібно ретельно продумати та спроєктувати.

Основний принцип полягає в наступному: користувач гортає сторінку з асортиментом товарів, додає бажаний у кошик і оформлює замовлення. Тобто у базі даних запланований мінімум – це три таблиці.

Перша таблиця – усі товари. Кожен запис містить опис товару, а саме його ідентифікатор (який є і ключовим полем), назву, кількість у наявності, вартість, зображення та опис. Ідентифікатор у MongoDB є рядковим типом даних (тому що це унікальний набір символів, автоматично згенерований), як і назва товару, зображення та опис. У полі із зображеннями міститься посилання на хмарне сховище, де знаходяться всі потрібні файли, саме через це визначений тип даних – рядок. Кількість у наявності та ціна, очевидно, є числовим типом даних.

Друга таблиця – це «підзамовлення». Тобто це є конкретний товар (а точніше – його ідентифікатор), його кількість і ціна. Аналогічно, ключовим полем є ідентифікатор підзамовлення. В цього поля та в ідентифікатора товару тип даних – рядок. Кількість і ціна – число. З таких підзамовлень у результаті формується цілісне замовлення.

Третя таблиця – це безпосередньо саме замовлення. Воно складається з

наступних полів рядкового типу даних: ідентифікатор замовлення (ключове поле), масив із ідентифікаторами підзамовлень, прізвище, ім’я та по батькові замовника, його електронна пошта, країна, область, місто, адреса. Окрім цього, є ще такі числові поля, як номер телефону, поштовий індекс і фінальна сума замовлення.

Врахувавши всі базові функції та потреби, було розроблено ER-діаграму (рис. 2.1). ER-діаграма (у перекладі з англійської «сутність-зв’язок») – це діаграма, що відображує такі сутності, як об’єкти, користувачі та система, і як вони пов’язані між собою.

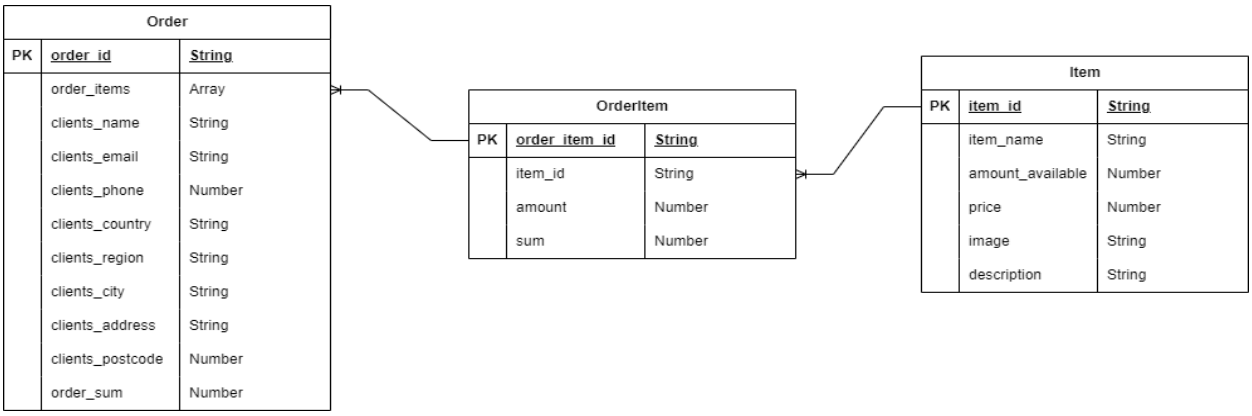

Рисунок 2.1 – ER-діаграма бази даних

2.2 Проєктування сайту та взаємодії клієнта з ним

Проаналізувавши отриману діаграму бази даних і бажані реалізовані функції, для зручнішої реалізації було розроблено діаграму прецедентів, класів і послідовностей.

Діаграма прецедентів (сценарій використання, Use Case) – це діаграма, що дає можливість уявити ролі (актори, себто користувачі або учасники) та їхню взаємодію з системою (конкретні дії – прецеденти) [5]. У проєкті акторами будуть клієнт, база даних. Серед прецедентів будуть наступні дії: перегляд товарів, додавання товару в кошик, оформлення замовлення (див. рис. 2.2).

«Діаграма класів (class diagram) – діаграма, на якій представлена сукупність декларативних або статичних елементів моделі, таких як класи з атрибутами й операціями, а також відношення, що їх з’єднують» [1, с. 34]. Для розроблюваного проєкту планується створити такий базовий набір класів:

- товар із вище визначеним переліком атрибутів (див. елемент Item на рис. 2.1), має метод переглянути товар;
- товар, що додається в кошик (див. елемент OrderItem на рис. 2.1) і має методи додати товар у кошик і видалити;
- замовлення, що містить перелік доданих у кошик товарів, атрибути зазначено на рис. 2.1, метод – оформити замовлення.

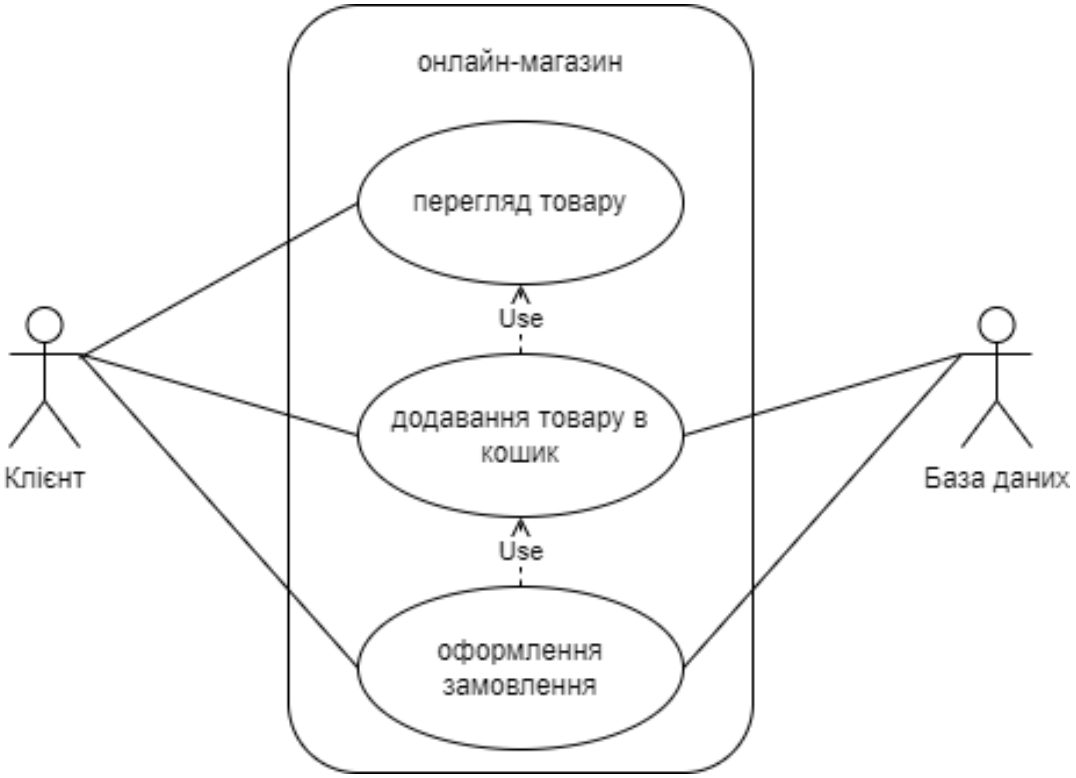

Рисунок 2.2 – Діаграма прецедентів

Між елементами Item та OrderItem, а також між OrderItem та Order встановлено зв’язок композиції, оскільки кожен другий елемент, зазначений у парі, не може існувати без першого [4].

Готова діаграма класів зображена на рисунку 2.3.

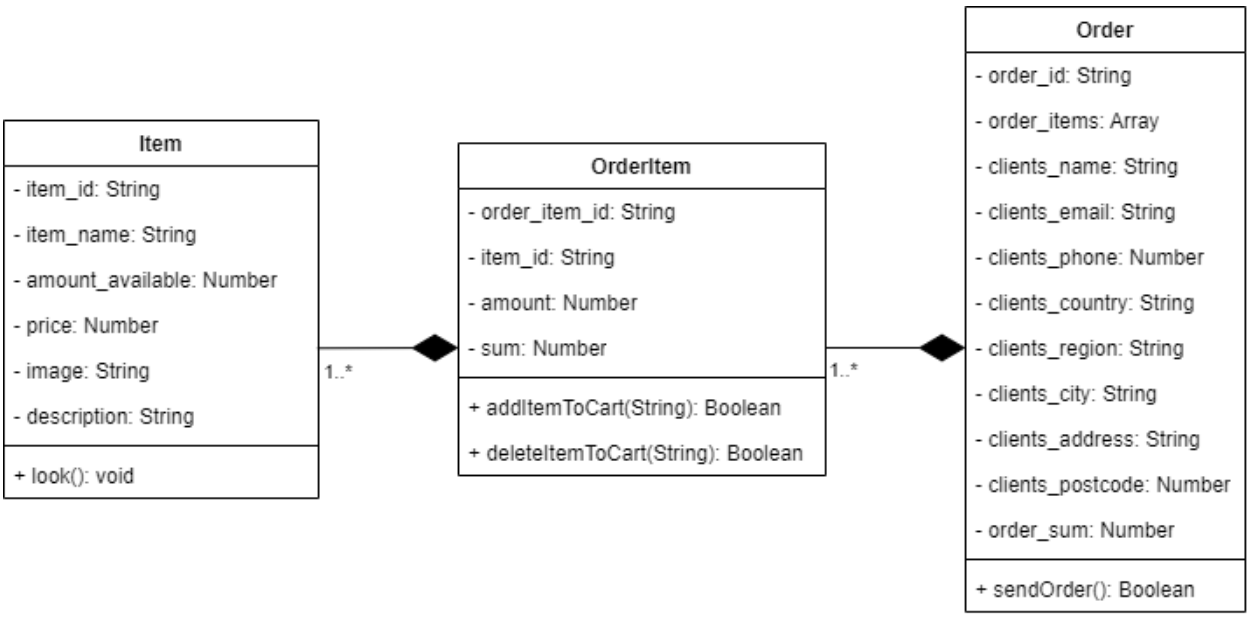

Рисунок 2.3 – Діаграма класів

Діаграма послідовностей відображає обмін повідомленнями між класами в конкретній обмеженій часом ситуації у порядку їхнього відтворення [4].

Для даного проєкту на діаграмі послідовностей будуть використані вже визначені класи та методи (див. діаграму класів на рис. 2.3). Логічною послідовністю буде наступна послідовність методів:

а) переглянути товари;

б) додати товар у кошик;

в) оформити замовлення.

Можливим альтернативним розвитком подій є після кроку № 2 метод «видалити товар із кошику». Тоді користувач може знову додати товар у кошик або ж за умови, що кошик не є порожнім, оформити замовлення.

Для коректного зображення всіх дій на діаграму додано лінію актора (тобто користувача).

Діаграма послідовностей, із урахуванням всіх вище зазначених вимог, матиме вигляд як на рисунку 2.4.

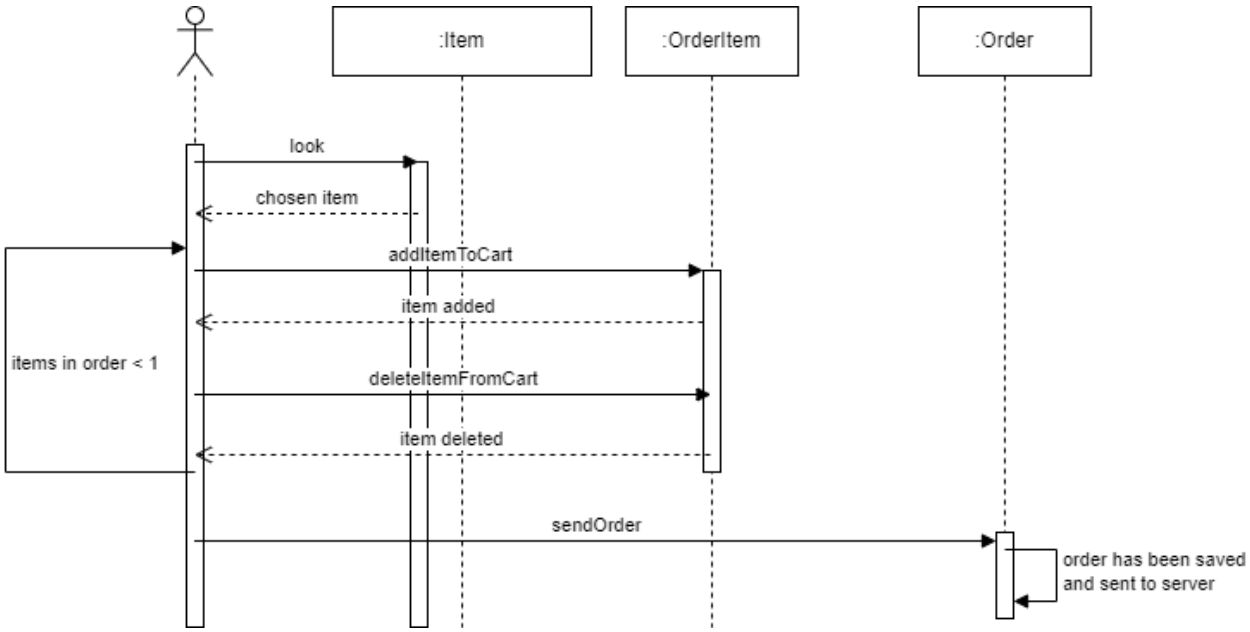

Рисунок 2.4 – Діаграма послідовностей

2.3 Висновки до розділу 2

У ході проведеного планування було розроблено ER-діаграму бази даних, діаграми прецедентів, класів і послідовностей. Окрім планування принципу дії та серверної частини, було продумано клієнтську частину сайту. Структура є базовою: заголовок сайту (header), суміжний із меню, основна частина з головним вмістом і підвал (footer). Серед сторінок сплановано наступні: головна, контакти, про нас, асортимент, кошик.

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

Матеріали опрацьовано в достатній кількості, технології проаналізовано, здійснено вибір найкращих і найзручніших засобів для конкретного проєкту, сплановано всі необхідні основні аспекти. Отже, можна приступати до процесу розробки продукту.

Для перевірки реалізації необхідного базового функціоналу, що запланований для розробки, додаток потребує проведення ручного тестування. Детальний процес буде описано в п. 3.3.

3.1 Реалізація онлайн-магазину

Загальна структура проєкту наступна: дві теки backend і frontend, що відповідають серверній і клієнтській частині відповідно (див. рис. 3.1).

У першій теці містяться потрібні модулі платформи Node, моделі кожного класу та необхідні шляхи, а також основний файл index.js і файли конфігурацій package.json та package-lock.json.

Тека frontend містить стандартні файли (.gitignore, README.md тощо) та теку public. Вони завантажуються автоматично при створенні проєкту із використанням React. Аналогічно до попередньої теки, клієнтська частина також має свої package.json та package-lock.json файли. Тека src наповнена власними сторінками та компонентами.

Шляхи використовуються для отримання доступу до сторінки, для публікації, оновлення чи видалення певної інформації. Загалом найпоширенішими HTTP-запитами, що лежать у основі, є GET, PUT, POST, DELETE [20]. Принцип роботи наступний: додаток надсилає запит, наприклад, типу GET на конкретну адресу. За умови успішно встановленого з’єднання між сервером і клієнтом на сайті відобразиться сторінка, яка прописана у

шляху на цю адресу. При запиті POST додаток зберігає за прописаним раніше шляхом інформацію, що була записана у форму, поле введення для запису тощо. При запиті PUT відбувається аналогічні дії до попереднього, але дані не створюються наново, а оновлюються. Запит DELETE, логічно, видаляє конкретний об’єкт.

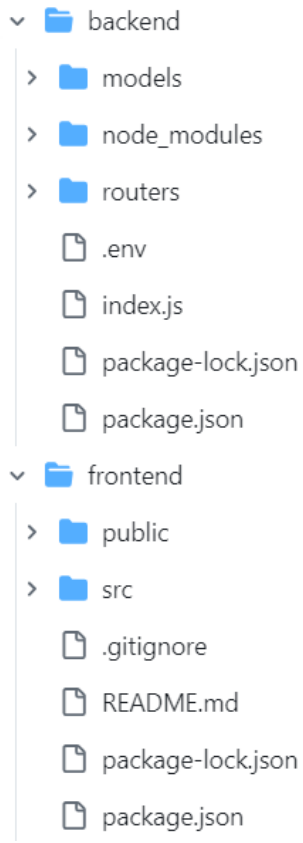

Рисунок 3.1 – Структура проєкту

Моделі – це базова частина проєкту при роботі з MongoDB. Для того, щоб мати змогу отримати, опублікувати чи оновити інформацію, яка має міститися в базі даних, потрібно додати відповідну модель. У кожної колекції бази даних є своя модель із визначеними назвами полів і типами даних. Для прикладу можна звернути увагу на модель об’єкту Order.

На рисунку 3.2 міститься фрагмент коду файлу order.js, що є моделлю об’єкту з колекції Orders. Іншими словами, додати новий об’єкт у колекцію замовлень можна лише відповідно до даної моделі. Всі наведені приклади моделей імпортуються в основний код або у відповідні файли шляхів.

```
const orderSchema = mongoose.Schema({
    client: {
        type: String,
        required: true
    },
    orderItems: [{
        type: mongoose.Schema.Types.ObjectId,
        ref: 'OrderItem',
        required: true
    }],
    phone: {
        type: String,
        required: true
    },
    email: {
        type: String,
        required: true
    },
    country: {
        type: String,
        required: true
    },
    region: {
        type: String,
        required: true
    },
    city: {
        type: String,
        required: true
    },
    address: {
        type: String,
        required: true
    },
    postcode: {
        type: String,
        required: true
    }
});

const Order = mongoose.model('Order', orderSchema);
```

Рисунок 3.2 – Модель Order (замовлення)

Наведемо приклади інших моделей. На рисунку 3.3 – модель товару з файлу item.js. Рисунок 3.4 містить модель підзамовлення (orderItem.js).

```
import mongoose from 'mongoose';

const itemSchema = mongoose.Schema({
   name: {
      type: String,
      required: true
   },
   category: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Category',
      required: true
   },
   price: {
      type: Number,
      required: true,
      min: 0
   },
   amountAvailable: {
      type: Number,
      required: true,
      min: 0
   },
   description: {
      type: String,
      default: ''
   },
   image: {
      type: String,
      default: ''
   },
});

const Item = mongoose.model('Item', itemSchema);

export default Item;
```

Рисунок 3.3 – Модель Item (товар)

```
import mongoose from 'mongoose';
const orderItemSchema = mongoose.Schema({
   item: {
      type: mongoose.Schema.Types.ObjectId,
      ref: 'Item',
      required: true
   },
   amount: {
      type: Number,
      required: true,
      min: 1
   },
   sum: {
      type: Number,
      required: true
   }
});

const OrderItem = mongoose.model('OrderItem', orderItemSchema);

export default OrderItem;
```

Рисунок 3.4 – Модель OrderItem (підзамовлення)

Для того, щоб отримати доступ до бази даних, до неї потрібно під’єднатися. Цьому відповідає невеликий фрагмент коду на рисунку 3.5.

```
mongoose.connect(process.env.DATABASE_CONNECTION)
.then(() => {
   console.log("Database is connected successfully!");
})
.catch((err) => {
   console.log("Something went wrong with database connection...");
});
```

Рисунок 3.5 – Встановлення зв’язку між сервером і базою даних

Окремими компонентами, що були розроблені для проєкту, є меню навігації (рис. 3.6) та підвал сайту (рис. 3.7). Вони є спільними для всіх

сторінок. У цьому перевага використання фреймворків – замість повторювання коду в HTML достатньо розробити компонент, який потім під’єднується в потрібне місце будь-якої іншої сторінки.

Рисунок 3.6 – Меню навігації сайту

Рисунок 3.7 – Підвал сайту

Сайт має декілька сторінок, що вже зазначалися вище. «Головна», «Контакти», «Про нас» не мають ніяких визначних особливостей, це типові сторінки для більшості сайтів, які містять простий текст, посилання, зображення тощо. Для сайту із подібною статичною інформацією достатньо базових засобів HTML та CSS.

Реалізація клієнтської частини – не просто поєднання HTML і CSS. Кожна сторінка є окремим JSX-файлом. Розглянемо будову сторінки Асортимент (див. додаток А).

На початку йде під’єднання потрібних модулів, далі – оголошення компонента Assortment і власне експортування цього компонента. Всередині нього описуються всі необхідні дії: отримання даних із серверу (товарів за адресою localhost:3000/assortment), функція для присвоєння кількості товару, функція додавання до кошику. Компонент повертає HTML-синтаксис із потрібними JavaScript вставками (особливість формату JSX). У результаті отримуємо секцію, що містить усі властивості товару, підтягнуті з відповідної колекції бази даних.

На сторінках «Асортимент» і «Кошик» уже реалізується зв’язок із базою даних, що і потребує окремого написання коду на JavaScript із використанням

Node.js. Так, наприклад, для того, щоб отримати весь перелік товарів із бази, достатньо буде фрагменту коду, який наведено на рисунку 3.8.

```
app.get('/assortment', async (req, res) =>
{
   try {
      const items = await Item.find();
      res.json(items);
   } catch (error) {
      console.error(error);
      res.status(500).json({message: 'Server error'});
   };
});
```

Рисунок 3.8 – Отримання всього асортименту товарів із бази даних

Сторінка «Асортимент» (рис. 3.9) містить перелік усіх об’єктів із таблиці Items, а конкретно відображає назву, зображення, опис, ціну та кількість у наявності для кожного об’єкту. Окрім витягнутих із бази даних значень у кожній секції є поле для введення кількості товару (обмежене мінімальним значенням 0 та максимальним – тим, що є в наявності) та кнопка, що додає товар в обраній кількості в корзину.

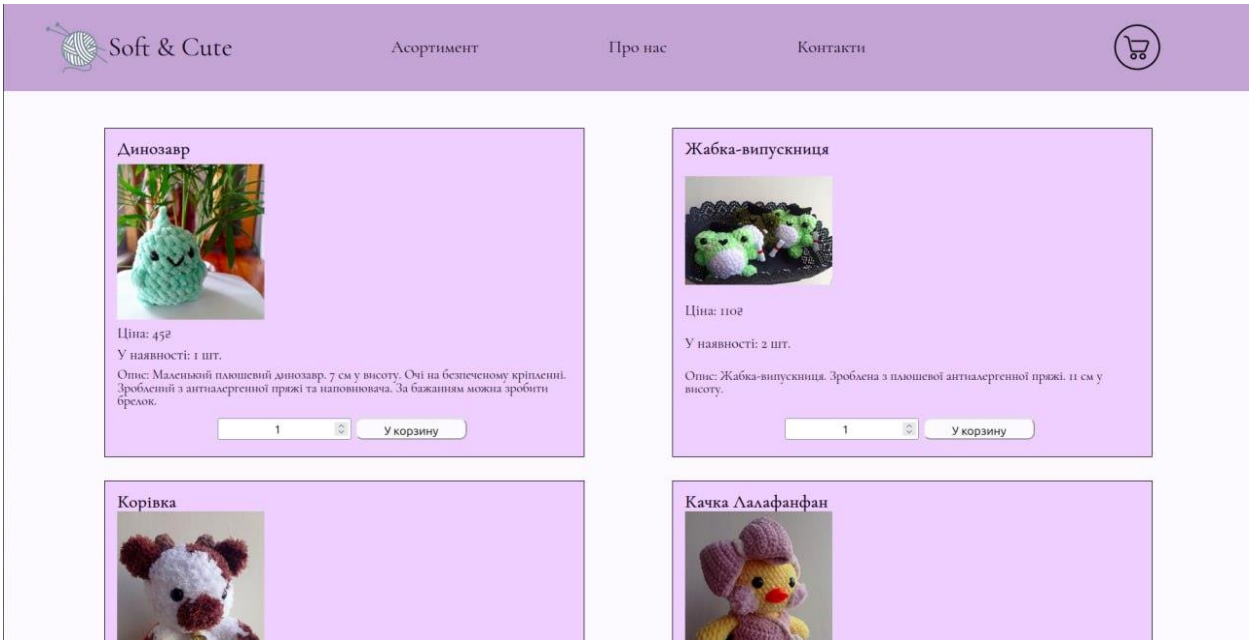

Рисунок 3.9 – Сторінка Асортимент

Кнопка «У корзину» надсилає POST запит (рис. 3.10), щоб додати об’єкт OrderItem до відповідної колекції в базі даних.

```
app.post('/cart/addItem', async (req, res) => {
   try {
      const { item, amount, sum } = req.body;
      const newOrderItem = new OrderItem({
         item,
         amount,
         sum
      });
      await newOrderItem.save();
      res.status(201).json({ message: 'Order item added successfully', newOrderItem});
      console.log(newOrderItem);
   } catch (error) {
      console.error('Error adding order item:', error);
      res.status(500).json({ error: 'Internal server error'});
   };
});
```

Рисунок 3.10 – Фрагмент коду надсилання POST запиту

У результаті додавання (або видалення) певних товарів у кошик кінцевий результат можна переглянути на сторінці «Замовлення» (рис. 3.11).

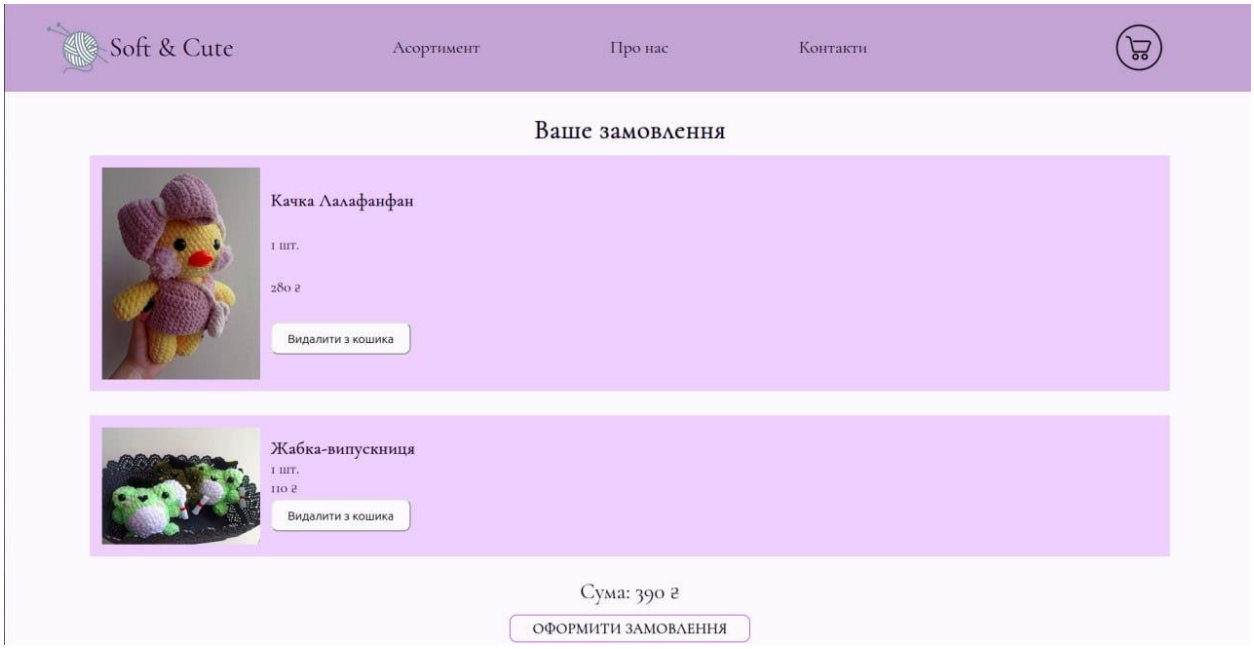

Рисунок 3.11 – Сторінка сайту Замовлення

Товари, додані до замовлення, відображаються в базі даних у відповідній колекції (рис. 3.12) згідно з моделлю, яка була прописана у відповідному файлі.

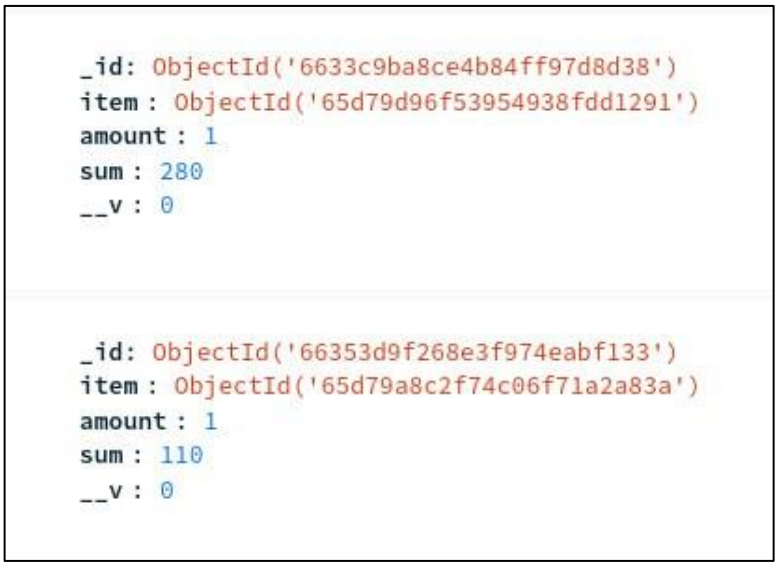

Рисунок 3.12 – Вигляд колекції із товарами, доданими до замовлення

У нижче наведеному фрагменті коду зображено видалення товарів із кошику за конкретним ідентифікатором (рис. 3.13).

```
app.delete('/cart/deleteItem/:id', async (req, res) => {
    try {
        const orderItemId = req.params.id;
        const deletedItem = await OrderItem.findByIdAndDelete(orderItemId);
        if (deletedItem) {
            res.status(200).json({message: 'Order item deleted successfully', deletedItem});
        } else {
            res.status(404).json({error: 'Order item not found'});
        }
    } catch (error) {
        console.error('Error deleting order item:', error);
        res.status(500).json({error: 'Internal server error'});
    };
});
```

Рисунок 3.13 – Видалення конкретного товару з кошику

Фінальним етапом буде заповнення форми клієнта для відправки замовлення. Сторінка має примітивний вигляд: форма з потрібними полями та кнопка відправки (рис. 3.14).

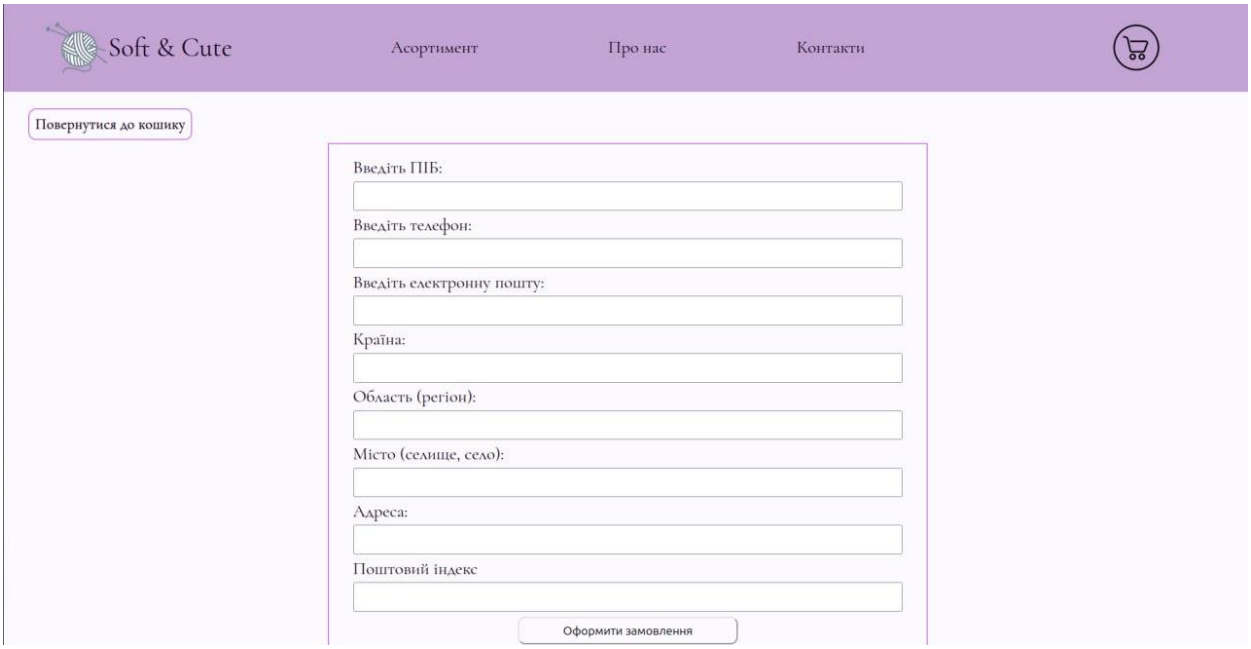

Рисунок 3.14 – Сторінка оформлення замовлення

Повний код додатку опублікований на сервісі GitHub та доступний за посиланням [12].

3.2 Адаптивна верстка

Розробка якісного продукту потребує більше, аніж просте підключення до серверу, бази даних і обробки інформації. Для того, щоб користувач мав змогу зручно переглядати сайт на будь-якому пристрої, існує адаптивна верстка, себто при конкретних значеннях ширини чи висоти екрану розміри конкретних об’єктів змінюються.

Стилізація сайту зроблена таким чином, що стилі для компонентів меню навігації та підвалу містяться в окремих файлах, а для головного вмісту всіх сторінок – у загальному іншому файлі. Властивості адаптивної верстки для

кожного компонента та для окремих об’єктів вказані у відповідних файлах.

Рисунок 3.15 містить фрагмент коду, що відповідає за адаптацію компоненту меню навігації до розмірів екрану.

```
@media screen and (max-width: 1350px) {
   .PageHeaderLogo {
      width: 35%;
   }
}

@media screen and (max-width: 910px) {
   .PageHeader {
      height: 100px;
   }
   .PageHeaderLogo {
      width: 44%;
   }
   .PageHeaderLogoText {
      font-size: 28px;
   }
   .MenuItemLink {
      font-size: 20px;
   }
   .CartMenuIcon {
      height: 90%;
   }
}

@media screen and (max-width: 630px) {
   .PageHeaderLogo {
      width: 20%;
   }
   .PageHeaderLogoText {
      display: none;
   }
}
```

Рисунок 3.15 – Адаптивна верстка компоненту меню навігації

Аналогічно можна розглянути адаптивну верстку для підвалу сайту (рис. 3.16).

```
@media screen and (max-width: 1350px)
{
   .Footer {
      height: 150px;
   }
}

@media screen and (max-width: 910px)
{
   .Footer {
      height: 100px;
   }
   .FooterText {
      font-size: 18px;
   }
}
```

Рисунок 3.16 – Адаптивна верстка компоненту підвал

Так, наприклад, розглянемо вище згадану сторінку «Асортимент». При ширині екрану менше за 1350 пікселів сайт дещо змінює свій зовнішній вигляд, а саме – розмір кожної секції з описом товару та ширину логотипу в меню навігації (рис. 3.17).

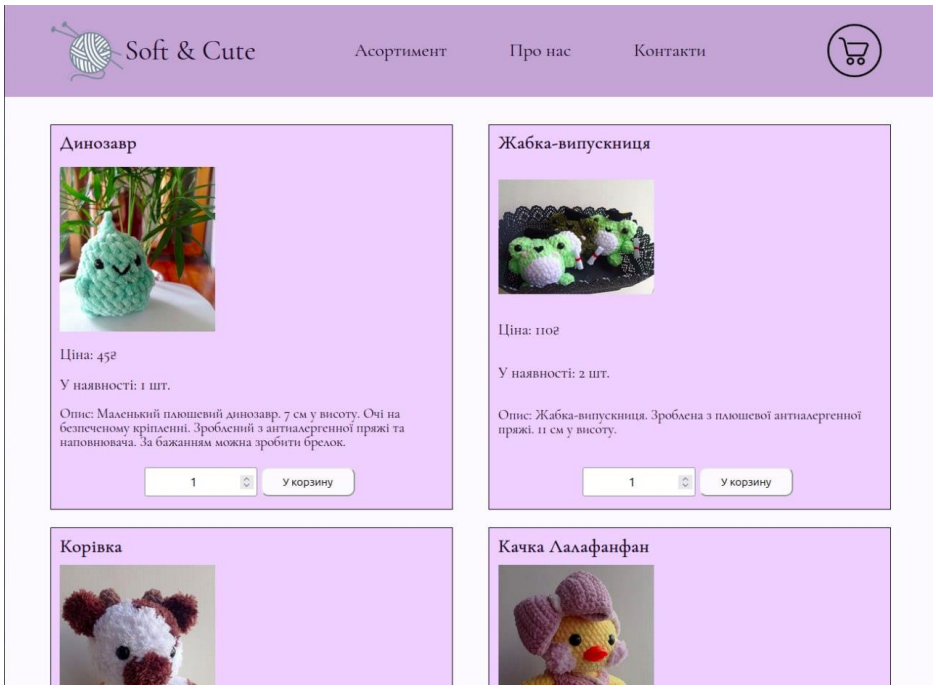

Рисунок 3.17 – Варіант сторінки «Асортимент» при ширині екрану 1200 пікселів

При встановленні ширини екрану менше за 910 пікселів змінюються розміри елементів і шрифтів. При зменшенні ширини екрану менше за 630 пікселів – змінює свій вигляд лише меню навігації, конкретно назва магазину зникає, і залишається лише логотип (рис. 3.18).

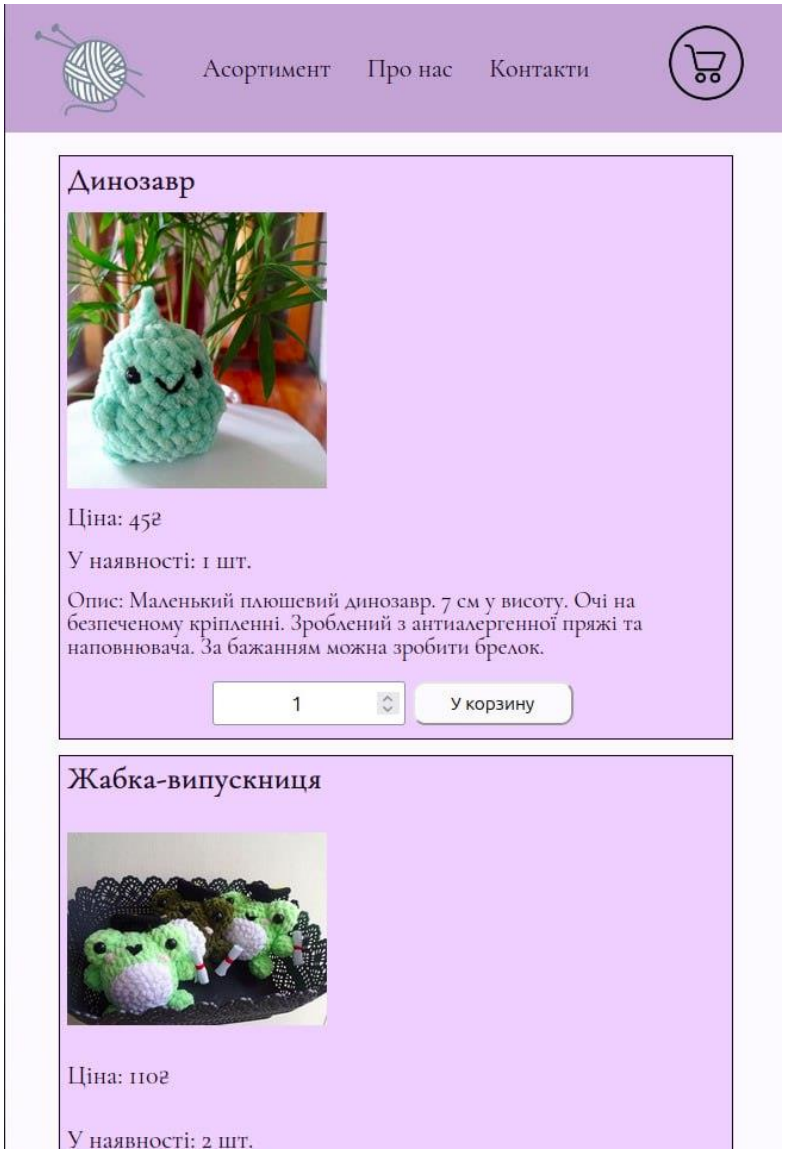

Рисунок 3.18 – Варіант сторінки Асортимент при ширині екрану 600 пікселів

Логічно, що компоненти на всіх сторінках при зменшенні розмірів екрану змінюються однаково. Для основної частини властивості вказуються відповідно до заздалегідь визначених класів. Розгляньмо ще один приклад сторінки – «Кошик» (рис. 3.19 – 3.20).

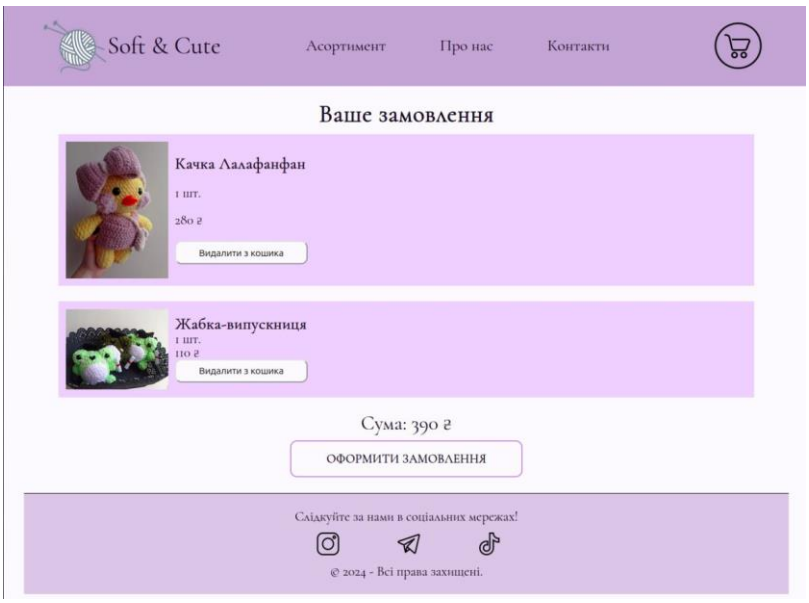

Рисунок 3.19 – Варіант сторінки Кошик при ширині екрану 1200 пікселів

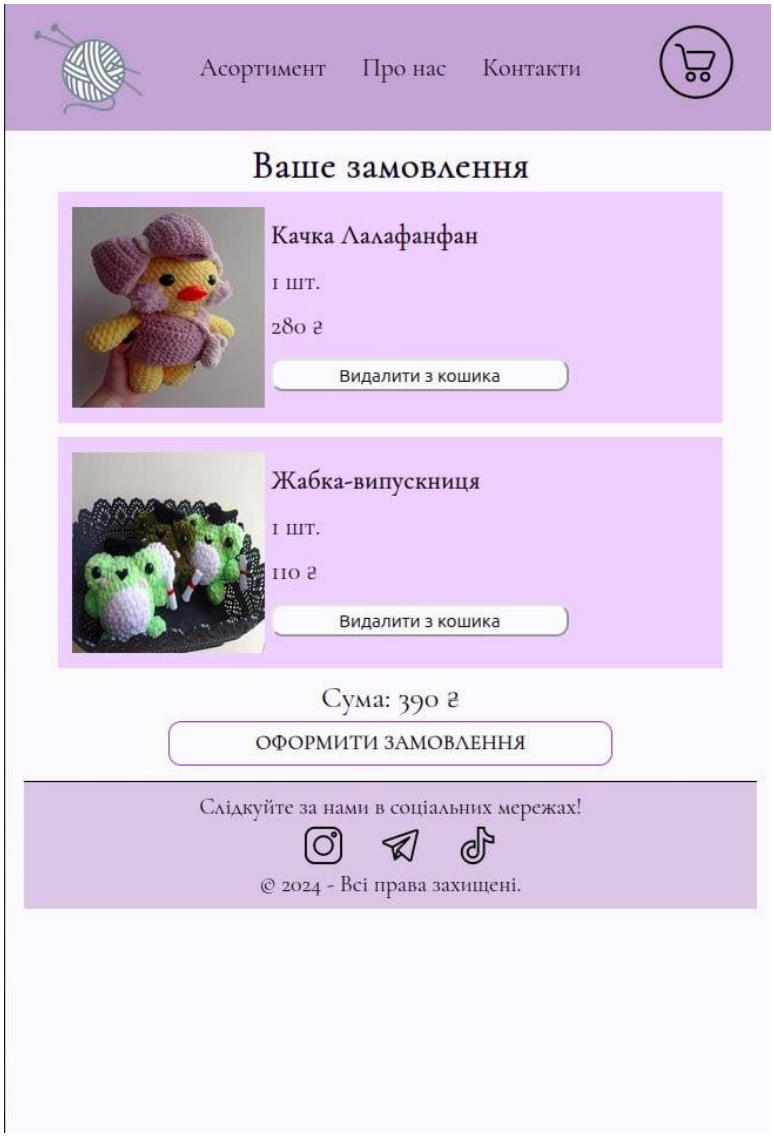

Рисунок 3.20 – Варіант сторінки Кошик при ширині екрану 600 пікселів

3.3 Тестування продукту

Для проведення якісного ручного тестування потрібно дати відповіді на ряд питань та провести декілька видів тестування: тестування верстки, функціональне, юзабіліті тестування тощо [6]. Чи весь визначений заздалегідь функціонал реалізовано? Чи відповідає сайт заданим вимогам? Чи зручний та зрозумілий сайт у використанні? Чи є верстка адаптивною до різних розмірів вікна? Відповіді на ці запитання детальніше описані в підпунктах нижче.

3.3.1 Функціональне тестування

Визначимо конкретні пункти, що стосуються функціональності та потребують перевірки.

У меню навігації працюють всі посилання на інші сторінки. При натисканні на логотип у меню із будь-якої сторінки відкривається «Головна». При натисканні на «Асортимент» відкривається однойменна сторінка. Аналогічно з посиланнями на сторінки «Про нас», «Контакти» та «Кошик» (зображення кошику). Отже, цей пункт виконано.

На сторінці «Асортимент» завантажуються та відображаються всі елементи товару. При переході на сторінку «Асортимент» завантажуються всі товари, що додані в колекцію Items з бази даних. Кожний товар розміщається у відповідній виділеній для нього секції, де міститься назва, зображення, опис, кількість у наявності та ціна. Всі властивості товару відображаються правильно згідно з даними, що занесені в базу даних.

На сторінці «Кошик» правильно відображаються додані товари. При переході на сторінку «Кошик», за умови, що користувач додав товари, вони коректно відображаються. Після переліку всіх товарів зазначається фінальна сума та кнопка переходу на сторінку «Оформити замовлення».

Сторінка Оформити замовлення відображає всі поля та відправляє замовлення. Після успішного додання товарів у кошик переходимо на сторінку «Оформити замовлення». Вводимо свої дані в поля (де електронна пошта – відповідний формат e-mail через @, номер телефону і поштовий індекс – числові типи даних). Після натискання на кнопку «Оформити замовлення» всі дані разом із доданими товарами додаються в колекцію Orders бази даних (рис. 3.21).

Рисунок 3.21 – Вигляд відправленого замовлення в базі даних

Посилання на сторінці «Контакти» працюють коректно. Переходимо на сторінку «Контакти». Бачимо на ній посилання на соціальні мережі. Клацнувши на кожне з них, переходимо на відповідні реальні сторінки, отже, посилання коректні.

Всі кнопки працюють правильно. На сторінці «Асортимент» є кнопки «У кошик» у кожній секції товару. При виборі кількості одиниць і натисканні кнопки товар відправляється у кошик.

На сторінці «Кошик» кнопка «Видалити з кошика» працює коректно: при натисканні товар у попередньо зазначеній кількості видаляється з кошика, при цьому сторінка автоматично перезавантажується.

На цій же сторінці «Кошик» є кнопка «Оформити замовлення». При натисканні користувач переходить на відповідну сторінку, де може заповнити форму. Кнопка «Оформити замовлення» на цій сторінці не відправить дані у базу до тих пір, доки не будуть заповнені всі обов’язкові поля.

Посилання у підвалі сайту працюють правильно. У підвалі сайту є три посилання на соціальні мережі. Вони дублюють посилання зі сторінки «Контакти». Аналогічно, після натискання на кожне з них користувач переходить на вище згадані сторінки у соціальних мережах.

3.3.2 Юзабіліті тестування

Під час юзабіліті тестування слід звернути увагу на зручність у використанні. Для цього потрібно подивитися на продукт із точки зору користувача.

При процесі юзабіліті тестування для розробленого проєкту розглядалися нижче зазначені пункти.

Інтерфейс усього сайту є інтуїтивно зрозумілим. Дійсно, при переході на сайт від початку запускається «Головна сторінка». У верхній частині сторінки розташоване меню навігації, назви елементів якого співпадають із відповідними сторінками (за винятком логотипу, що перенаправляє на головну, та зображення кошику, що направляє на сторінку «Кошик»). Унизу кожної сторінки міститься підвал, типовий для більшості сайтів, що містить посилання на соціальні мережі та копірайт. Головна частина містить певний контент, для кожної сторінки свій. Процес вибору товару полягає у перегляді сторінки «Асортимент». Там завантажуються всі товари. Логічно, що для формування замовлення потрібно додати товар у кошик, і для цього існує кнопка «У кошик» біля кожного товару. Кошик можна знайти, клацнувши на його іконку в меню. Там і буде весь перелік обраних товарів, і впевнившись, що все правильно, клієнт може переходити на сторінку оформлення

замовлення. Там також все зрозуміло – є форма для заповнення даними та кнопка для відправки.

3.3.3 Тестування верстки

Тестування верстки – це перевірка якості верстки сайту. Тобто потрібно оцінити, чи всі елементи відображаються правильно, чи є верстка адаптивною. Так як безпосередньо макету сайту немає, то пункт «Перевірка відповідно до макету» відпадає.

Всі елементи відображаються коректно. Жоден елемент не є вирваним із потоку. Всі об’єкти виглядають належним чином.

Ненагромадження кольорів. Додаток має певну кількість визначених кольорів. Вони всі належать до однієї палітри, тому це цілком відповідає стандартам якісної верстки.

Вирівнювання елементів. Усі елементи вирівняно або по центру, або по краю (у залежності від його призначення та функціоналу), тому ця умова виконана.

Розташування елементів відносно один одного. Елементи розподілено з рівномірними інтервалами за допомогою властивостей display: flex; align-items: center; justify-content: space-between тощо.

3.4 Висновки до розділу 3

У цьому розділі було описано результат розробки продукту з ілюстраціями та фрагментами коду. Також було наведено зміни безпосередньо в самій базі даних.

Окрім цього, було проведено функціональне та юзабіліті тестування, тестування верстки. Основні реалізовані функції відповідають вимогам та очікуванням.

ВИСНОВКИ

У результаті проведеної роботи було виконано ряд завдань, що були визначені на початку роботи.

Було проведено аналіз предметної області, пошук та порівняння аналогічних проєктів. Цим, у свою чергу, певною мірою пояснюється актуальність роботи (конкретно відсутністю онлайн-магазинів вузької спеціалізації в’язаних іграшок ручної роботи).

Окрім визначення тематики, для реалізації проєкту було визначено засоби розробки. Вибір таких технологій, як JavaScript, MongoDB обумовлений їхньою перевіреною взаємодією, існуванням додаткових платформ, що роблять роботу з ними значно зручніше. Фреймворк Express надає можливості забезпечити маршрути в додатку, відслідковувати помилки тощо. Фреймворк React дозволяє розбити проєкт на менші повторювані компоненти, використовувати JSX формат файлів, легко керувати станами.

Також було сплановано проєкт у цілому та окремі його компоненти. Для цього було створено всі потрібні діаграми (а саме ER-діаграма для проєктування бази даних, діаграма прецедентів, послідовностей та класів для відображення взаємодії між користувачем і системою).

Після всіх цих етапів власне і було реалізовано додаток. Він поділяється на клієнтську та серверну частини. Клієнтська частина розроблена за допомогою фреймворку React, який забезпечує більш універсальну роботу над сторінками, а саме комбінація мови розмітки HTML із мовою програмування JavaScript. Окрім цього, значною перевагою було додавання компонентів (меню навігації та підвал) для кожної сторінки не через прописування коду на кожній сторінці, а коротко через підключення окремих файлів JSX, які і містили вище зазначені компоненти.

Для оцінки продукту було проведено ручне функціональне та юзабіліті тестування, а також тестування верстки. У результаті було виявлено, що базові функції продукту вдало реалізовані.

Безперечно, проєкт має багато потенційних функцій, які ще можна додати, наприклад, створення акаунту для керування своїми замовленнями та власною інформацією, фільтрація товарів за певними властивостями, пошук. Щоб можна було просунути додаток і для громадян інших країн, можна реалізувати переклад із перемиканням на бажану мову інтерфейсу.

ПЕРЕЛІК ПОСИЛАНЬ

1. Бухтіяров Ю. В., Дідковська М. В. Основи проектування інформаційних систем. Частина І. С. 34. URL: https://studfile.net/preview/3751299/ (дата звернення: 26.03.2024).
2. Вступ до JSX – React. *React – JavaScript-бібліотека для створення користувацьких інтерфейсів*. URL: https://uk.legacy.reactjs.org/docs/introducing-jsx.html (дата звернення: 15.03.2024).
3. ДС редакція. JavaScript: як і чому варто почати – DSnews.ua. *«Ділова столиця» українською – найсвіжіші новини України та світу*. URL: https://www.dsnews.ua/ukr/future/javascript-kak-i-pochemu-stoit-nachat-23052023-480051#google_vignette (дата звернення: 04.03.2024).
4. Елементи UML. *KDE Documentation*. URL: https://docs.kde.org/trunk5/uk/umbrello/umbrello/uml-elements.html (дата звернення: 28.03.2024).
5. Інструкція, як будувати UML-діаграми. URL: https://dou.ua/forums/topic/40575/ (дата звернення: 24.03.2024).
6. Огляд видів тестування. *Онлайн-курси від компанії QATestLab | Головна сторінка*. URL: https://training.qatestlab.com/blog/technical-articles/review-the-types-of-testing/ (дата звернення: 17.04.2024).
7. Підручник Node.js Express FrameWork – навчіться за 10 хвилин. *Guru99*. URL: https://www.guru99.com/uk/node-js-express.html (дата звернення: 14.03.2024).
8. Плюси та мінуси онлайн-шопінгу: що потрібно знати. *Obozrevatel*. URL: https://life.obozrevatel.com/ukr/section-zhizn/news-plyusyi-i-minusyi-onlajn-shopinga-chto-nuzhno-znat-13-12-2023.html (дата звернення: 05.03.2024).
9. Створення сайтів і додатків з базою даних на MONGO DB: швидко,

професійно, під ключ | студія Brander. *Створення і розробка інтернет-магазинів від Brander*. URL: https://brander.ua/technologies/mongo-db (дата звернення: 16.03.2024).

10. Сучасний підручник з JavaScript. *Сучасний підручник з JavaScript*. URL: https://uk.javascript.info/ (дата звернення: 04.03.2024).
11. Gillis A. S., Botelho B. What is MongoDB? Features and how it works – TechTarget Definition. *Data Management*. URL: https://www.techtarget.com/searchdatamanagement/definition/MongoDB (дата звернення: 09.03.2024).
12. GitHub – daenerysssss/softncute. *GitHub*. URL: https://github.com/daenerysssss/softncute/ (дата звернення: 15.04.2024).
13. MongoDB: The Developer Data Platform. *MongoDB*. URL: https://www.mongodb.com/ (дата звернення: 15.03.2024).
14. Most used languages among software developers globally 2023 | Statista. *Statista*. URL: https://www.statista.com/statistics/793628/worldwide-developer-survey-most-used-languages/ (дата звернення: 09.03.2024).
15. Node.js – Run JavaScript Everywhere. *Node.js – Run JavaScript Everywhere*. URL: https://nodejs.org/en (дата звернення: 12.03.2024).
16. React – JavaScript-бібліотека для створення користувацьких інтерфейсів. *React – JavaScript-бібліотека для створення користувацьких інтерфейсів*. URL: https://uk.legacy.reactjs.org/ (дата звернення: 17.03.2024).
17. SQL vs. NoSQL: The Differences Explained + When to Use Each. *Coursera*. URL: https://www.coursera.org/articles/nosql-vs-sql (дата звернення: 01.03.2024).
18. Superior Codelabs IT Services. Top 10 Best Programming Languages To Learn in 2024. *LinkedIn: Log In or Sign Up*. URL: https://www.linkedin.com/pulse/top-10-best-programming-languages-learn-2024-superiorcodelabs-yx3ec (дата звернення: 09.03.2024).
19. Veeraraghavan S. Top 20 Best Programming Languages To Learn in 2024 |

Simplilearn. *Simplilearn.com*. URL: https://www.simplilearn.com/best-programming-languages-start-learning-today-article (дата звернення: 09.03.2024).

20. What are HTTP Methods? | Postman Blog. *Postman Blog*. URL: https://blog.postman.com/what-are-http-methods/ (дата звернення: 19.03.2024).
21. What is an Entity Relationship Diagram (ERD)?. *Lucidchart*. URL: https://www.lucidchart.com/pages/er-diagrams (дата звернення: 23.03.2024).
22. What Is Node.js? Complex Guide for 2023. *Digital Acceleration Company | Netguru*. URL: https://www.netguru.com/glossary/node-js (дата звернення: 03.03.2024).

ДОДАТОК А

Будова сторінок

А.1 Сторінка Assortment.jsx

```
import React, { useState, useEffect} from 'react';
import Axios from 'axios';

const Assortment = () => {
    const [items, setItems] = useState([]);
    const [cart, setCart] = useState([]);
    const [quantities, setQuantities] = useState({});

    useEffect(() => {
        fetchItems();
    }, []);

    const fetchItems = async () => {
        try {
            const response = await Axios.get('http://localhost:3000/assortment');
            if (response.status === 200)
            {
                const data = response.data;
                setItems(data);
                const initialQuantities = data.reduce((acc, item) => {
                    acc[item._id] = 1;
                    return acc;
                }, {});
                setQuantities(initialQuantities);
```

```
            }
            else {
                console.error('Failed to fetch items. Status:', response.status);
            }
        } catch (error) {
            console.error('Error fetching items: ', error);
        };
    };

    const handleQuantityChange = (itemId, quantity) => {
        setQuantities(prevQuantities => ({
            ...prevQuantities,
            [itemId]: quantity
        }));
    };

    const addToCart = async (itemId) => {
        const amount = quantities[itemId];
        const item = { _id: itemId};
        const sum = amount * items.find(item => item._id === itemId).price;
        const newItem = { item, amount, sum};
        setCart([...cart, newItem]);
        try {
            const response = await Axios.post('http://localhost:3000/cart/addItem',
newItem);
            if (response.status === 201) {
                console.log('Order item added to MongoDB successfully:',
response.data.newItem);
            } else {
                console.error('Failed to add order item to MongoDB. Status:',
```

```
response.status);
          }
      } catch (error) {
          console.error('Error adding item to MongoDB:', error);
      }
  };

  return (
      <div className='Main'>
          <ul className='ItemsMain'>
              {items.map(item => (
                  <li key={item._id} className="ItemSection">
                      <p className="ItemName">{item.name}</p>
                      <img src={item.image} alt="Item" className="ItemImage"/>
                      <p className="Price">Ціна: {item.price}₴</p>
                      <p              className="AmountAvailable">У              наявності:
{item.amountAvailable} шт.</p>
                      <p className="ItemDescription">Опис: {item.description}</p>
                      <section className="SelectAmount">
                          <input
                          type="number"
                          name="ItemAmountSelected"
                          className="InputItemAmount"
                          max={item.amountAvailable}
                          min={0}
                          value={quantities[item._id] || 0}
                          onChange={(e)          =>          handleQuantityChange(item._id,
parseInt(e.target.value))}/>
                          <button className="Button ButtonAddToCart" onClick = { () =>
addToCart(item._id)} >У корзину</button>
```

```
                    </section>
                </li>
            ))}

        </ul>
    </div>
  )
};

export default Assortment;
```