

7SEENG013W Software Development Project

2023/24

E-commerce Application

Student: Oladimeji Oladiti (w2034537/2034537)

Supervisor: Dr David Huang

Date: 15/09/2024

MSc Software Engineering (Conversion)
School of Computer Science and Engineering
College of Design, Creative and Digital Industries
University of Westminster

Abstract

This project aims to develop a sophisticated e-commerce web application to streamline sales and inventory management. The application targets product sales within the UK and US markets. Development was done using JavaScript on the client side and C# on the server side. The server leverages a MySQL database hosted on AWS to handle product and user information efficiently. Server-client communication is facilitated through the HTTP protocol, ensuring seamless interaction. The application's architecture is based on the .NET Web API framework to provide robust and scalable server-side functionality. Key technologies used in the project include the React library for the client interface and ASP.NET for the server-side logic.

List of Figures

List of Tables

Chapter 1 Introduction	7
Introduction	7
Project Aims and Objective	7
Project Aims	7
Project Objectives	8
Project Resources	8
Report Outline	9
Chapter 2 Requirement Analysis	10
Introduction	10
Systems to be Reviewed	10
Review Criteria	10
Criteria addressing Operational Effectiveness	10
Criteria addressing Technical Performance	10
Review of the Systems	11
Amazon	11
Alibaba	14
Jumia	15
Konga	17
Conclusion and Comparison of System	20
Chapter 3 Requirement Analysis	24
Introduction	24
System Stakeholder	24
System Context	26
System Requirements	27
Functional Requirements	27
Non-Functional Requirements	29
User-Interface Requirements	29
Chapter 4 System Design	31
Introduction	31
Use Case Diagram	31
Use Case Description	32
Sequence Diagram	38

Class Diagram	45
Entity Relationship Diagram	48
GUI Design	50
Wireframe Low Fidelity	50
Wireframe High Fidelity	54
Chapter 5 Implementation	57
Introduction	57
System Architecture	59
Review of Technologies	60
System Requirements	60
Implementation Issues and Solution	61
Implemented Class Diagram	62
Overview of Software Components	62
React Frontend Components	62
ASP.NET Backend Controller	65
Conclusion	67
Chapter 6 Testing	68
Introduction	68
Test Conducted	68
Functional Testing	68
Security Testing	68
Chapter 7 Conclusion	71
Introduction	71
Review of Project Aims and Objective	71
Review of Requirements	72
Functional Review of Requirements	72
Non-Functional Review of Requirements	73
User Interface Review of Requirements	74
Further Work and Improvements	75
Conclusion	75

List of Figures

Figure 2.1: Amazon Product Page

Figure 2.2: Amazon Shopping cart Page

Figure 2.3: Alibaba Product Page

Figure 2.4: Alibaba Verified Seller

Figure 2.5: Jumia Product Search

Figure 2.6: Jumia Payment Section

Figure 2.7: Jumia Payment Section

Figure 2.8: Konga Product Search

Figure 2.9: Konga Payment Section

Figure 2.10: Konga Payment Section

Figure 3.1: Stakeholder Diagram

Figure 3.2: Context Diagram

Figure 4.1: Use Case Diagram

Figure 4.2: Sequence Diagram

Figure 4.3: Sequence Diagram

Figure 4.4: Sequence Diagram

Figure 4.5: Sequence Diagram

Figure 4.6: Sequence Diagram

Figure 4.7: Sequence Diagram

Figure 4.8: Class Diagram

Figure 4.9: ERD Diagram

Figure 4.10: Wireframe Low Fidelity

Figure 4.11: Wireframe Low Fidelity

Figure 4.12: Wireframe Low Fidelity

Figure 4.13: Wireframe Low Fidelity

Figure 4.14: Wireframe Low Fidelity

Figure 4.15: Wireframe Low Fidelity

Figure 4.16: Wireframe High Fidelity

Figure 4.17: Wireframe High Fidelity

Figure 4.18: Wireframe High Fidelity

Figure 4.19: Wireframe High Fidelity

Figure 4.20: Wireframe High Fidelity

Figure 5.1 system architecture

Figure 5.2 Implemented Class Diagram

List of Tables

Table 2.1: System Comparison Result

Table 6.1: Security Testcase

Table 6.2: API Testcase

Chapter 1 Introduction

1.1 Introduction

Success in the ever-evolving world of online retail is not only imperative on a business level, but can also have far larger implications for the e-commerce industry. The aim of this project is to generate an innovative e-commerce web application that makes the sale and control product inventory, seeking to adapt more easily according with consumer/business requirements on business environment. The main idea is to enable smooth transactions and strong inventory management of for products. As it pertains to the development process of this project, two main programming languages were implemented: JavaScript on the client side and C# for server-side. It will contain a client-side with React library to create an easy and smooth user experience and interface as well for small screens, And Server side in ASP. NET framework, to process incoming data more reliably and efficiently.

The application is built on an SQLite database, providing high availability and security for product and user data. The architecture of the application provides server-side operations which is both scalable and maintainable. It makes native communication between server and client through the HTTP protocol to exchange data in a safe & efficient way. The platform is aimed at addressing the core challenges of contemporary e-commerce like inventory tracking in real-time, end-end user interactions and handling data comprehensively. The following chapters of this report will explore all the parts that make an application describing design decisions, implementation strategies and technological solutions above mentioned. We go more in depth on chapter 5, where the tools and methodologies used in the Server and client development are explained.

1.2 Project Aims and Objectives

1.2.1 Project Aims

- 1) **Develop a User-Friendly E-Commerce Platform**
Create an intuitive and responsive e-commerce web application that facilitates seamless online shopping experiences.
- 2) **Implement Comprehensive Inventory Management**
Provide robust inventory management functionality to ensure real-time tracking of product availability and stock levels.
- 3) **Enhance User Experience with Modern Web Technologies**
Leverage modern web technologies to deliver a visually appealing and highly performant user interface.
- 4) **Ensure Secure and Scalable Backend Infrastructure**
Develop a secure, scalable backend to efficiently handle user data, transactions, and product information.

1.2.2 Project Objectives

- 1) **Aim 1: Develop a User-Friendly E-Commerce Platform**
 - a) Design and implement a user-friendly interface using React, ensuring ease of navigation and accessibility.
 - b) Integrate payment gateways to facilitate secure and convenient transactions.
- 2) **Aim 2: Implement Comprehensive Inventory Management**
 - a) Develop backend logic using C# and SQLite to manage product inventories, including adding, updating, and removing products.
 - b) Implement real-time inventory tracking to update stock levels automatically based on user purchases.
- 3) **Aim 3: Enhance User Experience with Modern Web Technologies**
 - a) Utilize HTML, CSS, JavaScript, and SASS to create a responsive and visually appealing user interface.
 - b) Implement Redux for state management to improve application performance and user experience.
- 4) **Aim 4: Ensure Secure and Scalable Backend Infrastructure**
 - a) Use ASP.NET Core Web API for the backend to handle server-side operations, ensuring security and scalability.
 - b) Implement user authentication and authorization to protect sensitive user information and control access.

1.3 Project Resources

A 10th generation Intel i7 HP laptop with enough processing power was used to complete the e-commerce application development. Also, an MSI monitor with great resolution was used with readability greatly increased and it was easier to work in terms of large UI components refactoring and two or even three code windows being open at the same time. The backend development was done using visual studio which is most preferable integrated development environment (IDE). The main reason I chose Visual Studio was for its fine support for Entity Framework and a robust ORM (Object-Relational Mapping) tool that enables simplified database-environment interactions. The IDE has a wealth of tools at its disposal from the integrated package management by NuGet, good debugging tools and full support for ASP.NET core for development. On the front end, Visual Studio Code was used. As it is lightweight and yet powerful, which made it favorite among the React developers. I used Visual Studio Code for the coding the frontend because it was very handy with its little extensions that extended the language support, which made me possible to code fast and correct in any of JavaScript, HTML, CSS and SASS.

1.4 Report Outline

Chapter 2 is the review of four e-commerce systems comparable to our planned development project. Chapter 2 starts by explicitly defining the system review criteria and rating each of them at the end in terms of, again based on those same parameters. Chapter 3 introduces the stakeholder and context diagrams, which help to analyze the potential needs of the users of this system, followed by clear definitions of the different requirement types.

Chapter 4 explains the system design by providing Use Case diagrams and also identifies all Use Case scenarios. It also includes sequence diagrams for these Use Cases, as well as a Class Diagram that represents the application's design. Additionally, this chapter presents the entity-relationship diagram and user interface designs. Chapter 5 explains the system implementation by showing the system architecture, the technology stack used, and the Class Diagram for the implemented software. Chapter 6 focuses on performance testing to test the application for different scenarios. Finally, we finish off the report in Chapter 7 by overviewing main achievements of our project and mentioning limitations as well as outlining directions for future work.

Chapter 2 Requirement Analysis

2.1 Introduction

In today's world, developing an e-commerce application needs a clear understanding of the system's functional and non-functional requirements. This chapter contains the process of analyzing the necessary requirements for the e-commerce application. This involves analyzing e-commerce applications that already exist.

2.2 Systems to be Reviewed

This section will thoroughly compare four big eCommerce platforms in the world which are: Amazon, Alibaba, Jumia (Nigeria), and Konga (Nigeria). The aim is to examine how this platform is related to our project and also how it differs from our e-commerce application. By doing this analysis, we find out the efficiency, performance and user experience of these similar systems. This comparative study will also enable us to know the best practices and proper techniques, which will, in turn, contribute to the development of our e-commerce application. We will focus on critical aspects such as business functionality, scalability, customer outreach, and regional adaptability to understand how these platforms excel in the global and Nigerian eCommerce markets.

2.3 Review Criteria

This project's review criteria are structured to provide insight into the assessment of both the operational aspect and the technical aspects. It is divided into two areas:

- 1) **Operational Effectiveness:** The operational effectiveness criteria are concerned with how well the e-commerce system supports and improves important business operations. It explains things like the user experience, order management and the processing of payments, which therefore ensures that the e-commerce system meets its operational goals
- 2) **Technical Performance:** The technical performance contains criteria that analyze the technicality and the design quality. The main areas of focus are how well the system performs, like the speed of loading a page, the security, the quality of the code used in developing the system, mobile responsiveness and lastly, the cross-browser compatibility of the application

Criteria for Evaluating Operational Effectiveness:

- 1) **User Interface Design:** Visual design, layout consistency, and aesthetic appeal.

- 2) **Navigation:** Ease of navigating the site, finding products, and accessing various sections.
- 3) **Search and Filtering:** Effectiveness and accuracy of the search function and product filters.

Criteria for Evaluating Technical Performance:

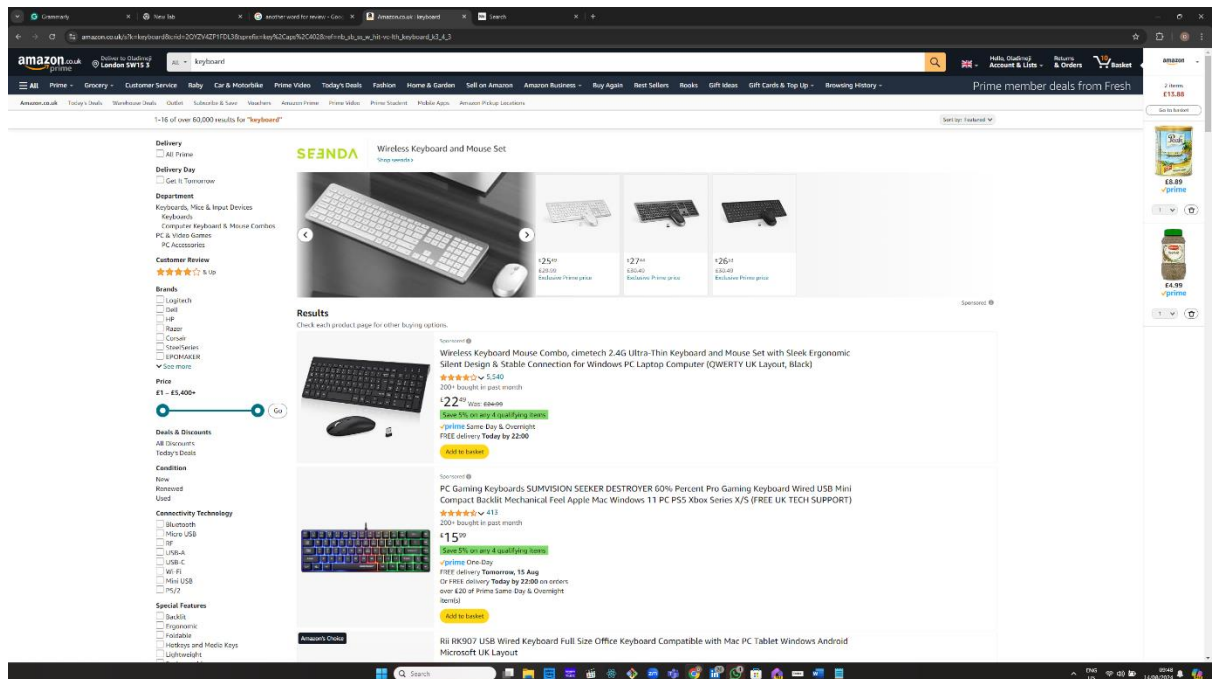
- 1) **Product Management:** Features for adding, editing, and managing product listings.
- 2) **Order Management:** Capabilities for processing and tracking orders.
- 3) **Payment Processing:** Integration with payment gateways and transaction security

[2.4 Review of the Systems](#)

2.4.1 Amazon

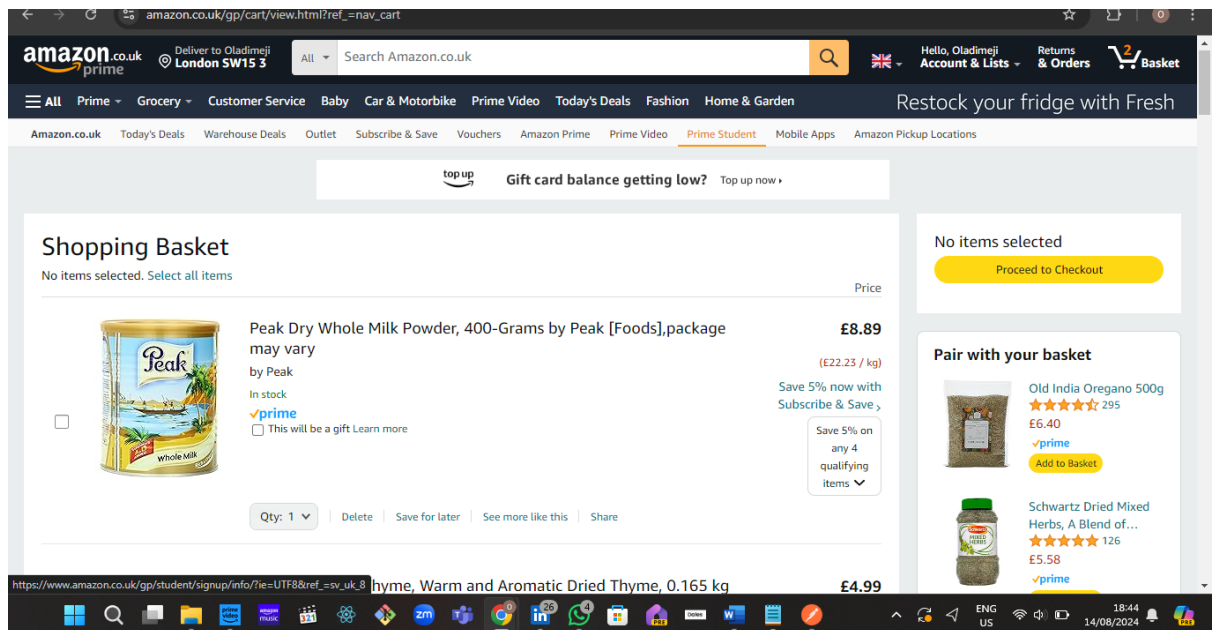
The first similar system to be reviewed, which is one of the world's largest and also one of the world's most influential, is Amazon. Amazon is well known for its vast number of products, advanced features and customer-focused approach. Amazon was founded in 1994, and it has improved by offering a wide range of services, from retail to cloud computing, and some of its latest services, such as Amazon Music and Amazon Video. This review will focus on two key functionalities: Product Search and Filtering and Add to Cart and Wishlist. Below is the product search page where a search for keyboards was performed, resulting in a list of keyboard options.

Figure 2.1: Amazon Products Page



The Amazon search functionality is a very important aspect of the Amazon's platform; this functionality is designed to deliver relevant products quickly to the users when they search for a product using its keyword. The efficiency of Amazon's search algorithm is one of the reasons why the platform is so successful, which makes it an important area for sellers to understand and optimize. Amazon's algorithm seeks to return to the user the best possible match between the user's query and the products available in the e-commerce system. Unlike traditional search engines prioritizing web pages, Amazon's search engine, often referred to as A9 (now part of Amazon's broader AI and machine learning efforts), focuses exclusively on product listings. The primary goal is to increase product sales by showing the users the products they would be interested in and are most likely to buy. When a user enters a search query, Amazon's algorithm uses this input to determine the most relevant results.

Figure 2.2: Amazon Shopping Cart Page



The “Add to Cart” and “Add to Wishlist” are very important features on Amazon, and they are designed to improve the shopping experience and increase user engagement on the platform. The “Add to Cart” functionality allows the users to easily add products they would like to buy to their shopping cart. The button is placed on every simple product on the platform, making it easier and more convenient for the user to use. Once a product is added to cart, the cart is automatically updated increasing the number of products in the cart, and on the amazon shopping cart page, the product is displayed with its price, number of quantity and image. Also, it must be noted that the quantity of the product can also be increased and decreased. The cart also summarizes and displays the total price of the items in the cart, which is essential, especially if the cart contains multiple different items.

With the "Add to Wishlist" feature customers can leave products they like in the wish list section without buying. You can create multiple different Wish lists and sort them by what they are, i.e. gifts you want soon or just in the future. This means it helps users to remember items that they would like to buy later or also watch for price changes. You can also share your Wishlist with others, such as during the holidays or birthdays. Amazon also offers customers built-in features for controlling their Wish lists, such as adding comments to products and starring them or organizing long lists into separate sections. It also has an integration with Amazon's recommendation engine, which will alert users to discounts or if items get back in stock. Together they provide a hassle-free customer shopping experience on Amazon helping them to manage their purchase process in perfect control.

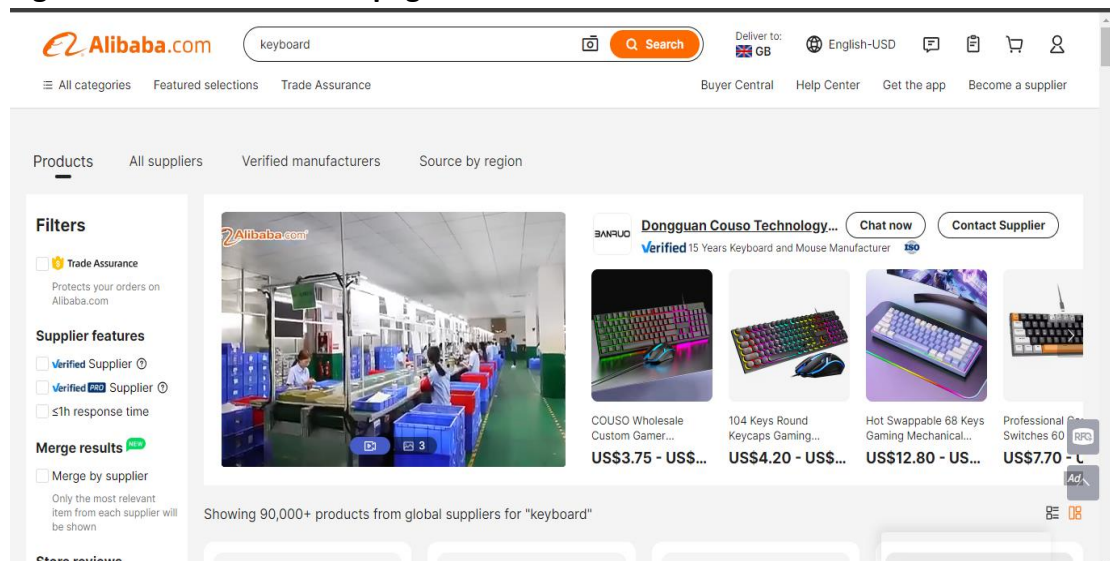
The Amazon e-commerce platform is responsive and the pages search results loads very quickly with high traffic. It has a nice layout design and simple to navigate

around it. Implementing add-to-cart makes sure the users can save items in one place and at the same time estimate of what they are more likely to buy which therefore enhances shopping experience.

Alibaba

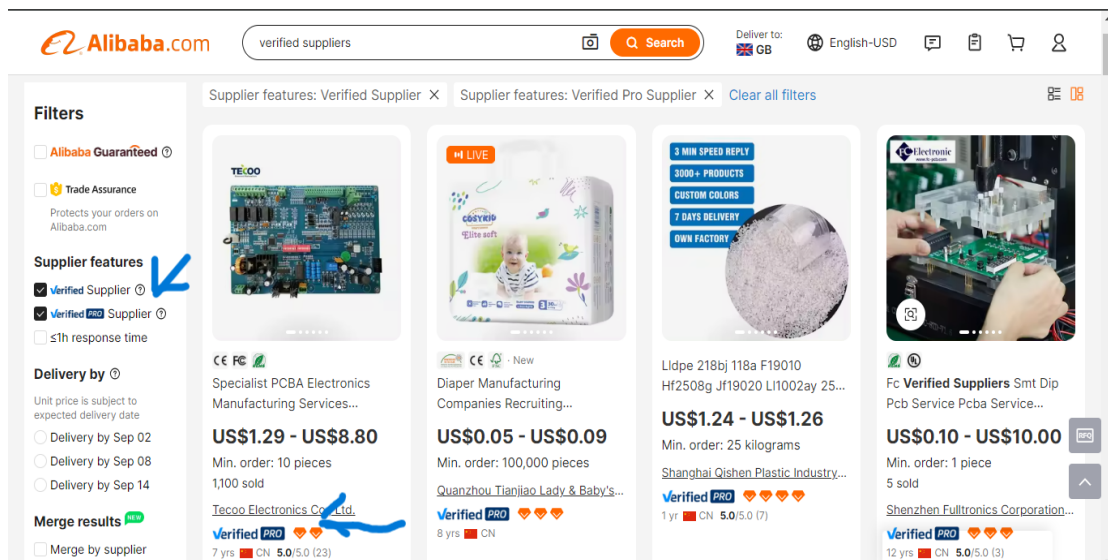
Like Amazon, Alibaba is also one of the largest e-commerce companies in the world but mainly focuses on business-to-business transactions. Alibaba has everything to offer through its e-commerce platform — ranging from consumer electronics, household things to large industrial machinery. Similarly, the Amazon review is focussed on two features; Product Search and Supplier Verification.

Figure2.3 Alibaba Products page



Th Alibaba search is available to help businesses instantly locate suppliers and products. The search bar, like the Amazon one set prominently on it's homepage is placed right in front of your eager fingers ready to type away what most closely resembles (or has anything at all related) to that product you're looking for. When a user performs their search operation, the Alibaba algorithm conducts an in-depth search for suppliers and products that match what this user requested. It allows the users to narrow down their search filters based on product type, minimum order quantity and supplier location

Figure2.4 Alibaba Verified Seller



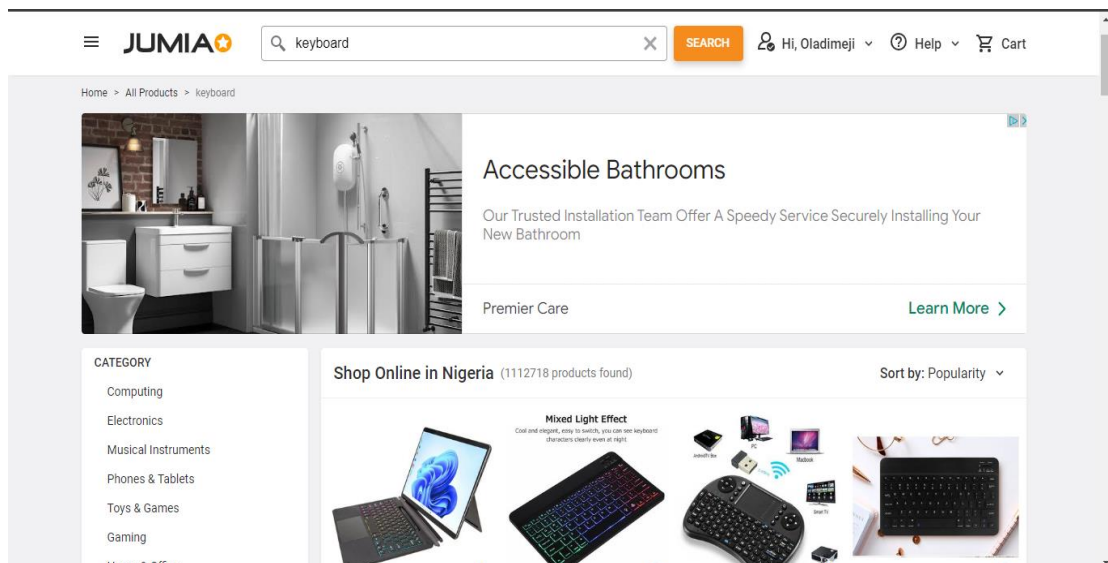
The next feature to look at is Alibaba's supplier verification system. It helps users identify much more trustworthy suppliers on Alibaba, goes through a rigorous vetting process, and lets you see their profile display badges, such as verified or verified pro. These badges appear on product listings to assure buyers. Alibaba also provides trade assurance, safeguarding buyers from quality problems with the products they order and payment errors.

Alibaba was founded with a focus on business users, and the platform is designed for that target audience, making the platform more system than fashionable elegance. The Alibaba website is very user-friendly, and search results return fast, as do supplier profiles. This design is pretty simple, showcasing the products and suppliers. The user interface is perhaps a little less pretty than say, Amazon, but it does the job nicely for business users. In sum, Alibaba is a robust and absolutely complete system for b2b e-commerce.

Jumia (Africa)

Jumia is one of Africa's leading e-commerce websites; Jumia offers various products from fashion to electronics and household goods. Unlike Amazon and Alibaba, founded many years ago, Jumia is a much younger e-commerce platform that was founded in 2012, around 12 years ago. Jumia primarily operates in Africa and focuses on the African market, where it is often referred to as the "Amazon of Africa". It serves over a dozen African countries, including Nigeria, Kenya and Egypt. It also includes services like Jumia Pay for payments and Jumia Food for food delivery, tailored to the needs of the African market. This review focuses on two main functionalities: product or search and payment options

Figure 2.5: Jumia Product Search



Alibaba product searches, Jumia users can also carry out an easy and transparent shopping process. The search feature of the Jumia application has been customized for the African markets with features that enable users and visitors to get products located around them. The Jumia's search bar is user-friendly because users can find and buy products easily. The search engine is related in relevance scores to the product titles, descriptions and categories. Users can then filter the results by price, brand, and seller location. The search query is designed to display products that can be delivered in the user's area, making it easier for customers to find readily available items.

Figure 2.6: Jumia Payment Section

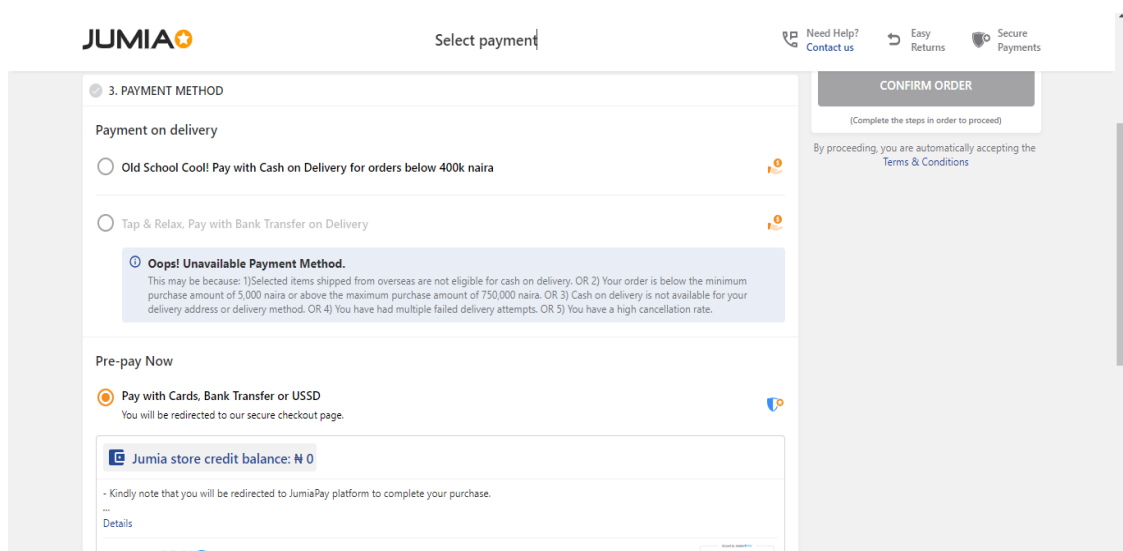
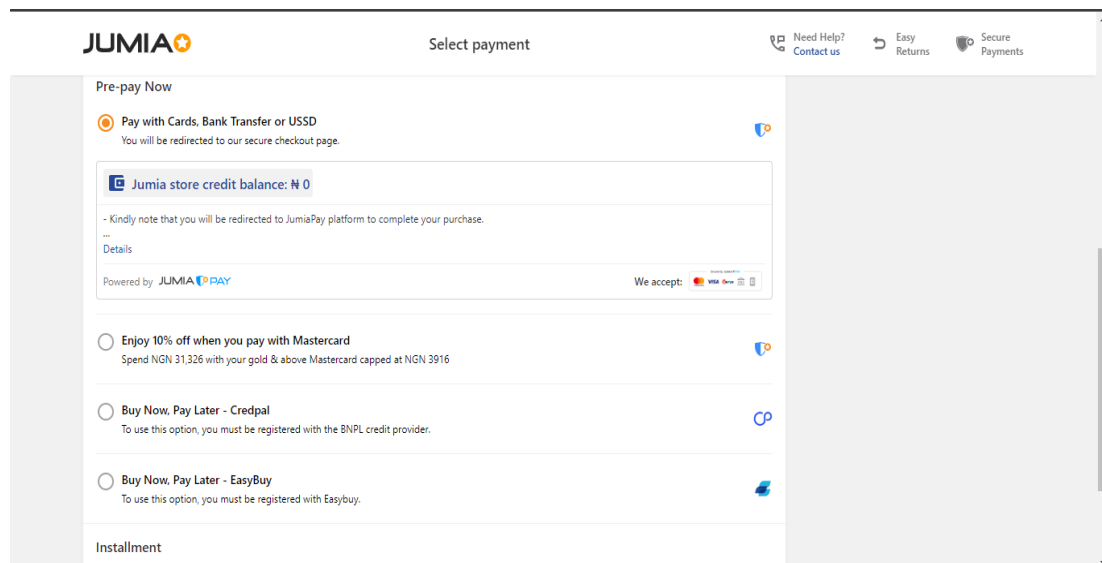


Figure 2.7: Jumia Payment Section



Jumia is unique compared to big e-commerce platforms like Amazon and Alibaba in its various payment options, which are tailored to match the many faces of finance across Africa. Jumia supports mobile money platforms which are popular in most countries on the continent, as well as traditional cash payments.

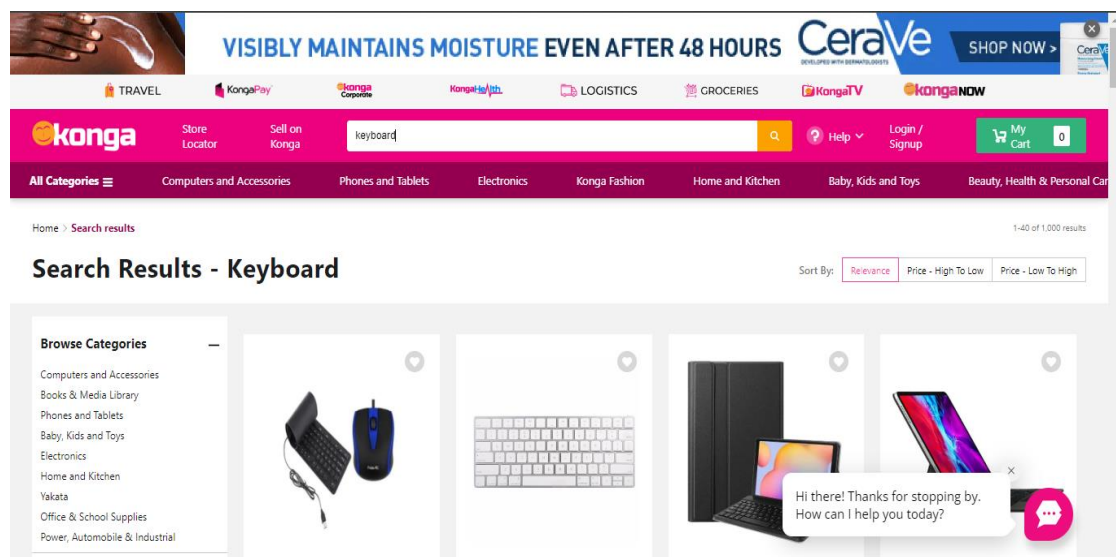
It also has a cash on delivery feature where all the items ordered are inspected by customers and then payment is done. That payment flexibility is vital to build trust with customers in places where eCommerce penetration remains low

Jumia e-commerce platform is adapted to be used by Africans and it loads side quickly for even those with poor network. It has a nice design with a user-friendly interface that makes navigation as easy as Amazon or Alibaba. Jumia is a trustworthy and convenient online shopping website that caters to the African lifestyle.

Konga (Nigeria)

Konga, a Nigerian e-commerce platform, set itself apart from other e-commerce platforms like Jumia, amazon and Alibaba through its strong focus on the Nigerian and broader West African markets and offering services and products that meets the needs of this region. Unlike Amazon's global reach and Alibaba's emphasis on business-to-business trade, Konga integrates online shopping with offline experiences through its vast network of physical stores, which Jumia and Amazon do not offer. Also, Konga prioritizes the cash-on-delivery options just like Jumia, catering to the cash-based economy prevalent in Nigeria. Konga also operates Kong Pay which is a payment system designed to facilitate secure transactions within its ecosystem, which addresses local challenges in digital payments. This local market focus, hybrid online and onsite model and tailored payment solutions set Konga apart from its global counterparts. This review focuses on two key functionalities: Product Search and Recommendations and Konga Pay Payment Integration.

Figure 2.8: Konga Product Search



Konga search functionality, while similar in basic design to Jumia, Amazon and Alibaba, is tailored to the specific needs of its Nigerian and West African users. Like other e-commerce platforms, Konga features a straightforward search bar, filtering and sorting options, allowing users to find products based easily on the price and customer ratings. However, Konga's search results often concentrates on local products and sellers, which shows that it is regional based, whereas Amazon and Alibaba operate globally. Contrary to Amazon's advanced search functionality, the Konga search system is more straightforward, catering to the local market demand for essential goods and popular items. Unlike Alibaba, which prioritizes business-to-business search and purchasing in bulk, Konga's search system is more concerned with individual users and their everyday purchases. In addition to Konga's search capabilities, it also offers product recommendations which is based on the user's browsing history and their previous Orders. These recommendations are shown in the homepage and product pages, which helps the user to discover new products that match their interest.

Figure 2.9: Konga Payment Section

The screenshot displays the '2. PAYMENT OPTIONS' section of the Konga checkout process. It features three radio button options: 'Pay Now' (selected), 'Buy Now Pay Later', and 'Pay on Delivery'. The 'Pay Now' option includes logos for VISA, Mastercard, and Verve. Below these options is a section for 'Add Voucher Code' with a text input field and an 'Apply' button. A 'Continue to Payment' button is at the bottom. On the right, a summary sidebar shows: Sub total: ₦173,000; Delivery Fee: ₦2,652; Total: ₦175,652. A promotional banner for 'KongaNOW' offers same-day delivery before 3PM in Lagos and Abuja. A chat bubble at the bottom right says 'Hi there! Thanks for stopping by. How can I help you today?'.

Figure 2.10: Konga Payment Section

This screenshot shows the same Konga payment section as Figure 2.9, but with a 'SELECT PAYMENT METHOD' modal open in the center. The modal lists three options: 'KongaPay' (Pay With Your KongaPay Wallet), 'Bank Transfer' (Make Payments Using Bank Transfer), and 'Card' (Make Payments With Your Credit Or Debit Card). The background is dimmed, showing the 'Pay Now' option and the summary sidebar. A security notice at the bottom of the modal states: 'This is an encrypted payment, your details are 100% secure and safe'.

Konga pays payment functionality, when compared to Jumia pay, have similarities and is specifically developed based on the need of the Nigerian market. Like Jumia Pay, Konga Pay offers a range of payment options, including credit cards, debit cards and bank transfers, which allow users to complete transactions securely and conveniently.

Also, Konga and Jumia supports cash-on-delivery, which is essential in a market where lots of customers prefer to verify products before paying. However, Konga Pay is more focused on facilitating transactions within the Konga ecosystem, offering a streamlined and straightforward experience that is tightly integrated with the Konga online marketplace. In contrast, Jumia pay has a broader application in multiple countries in Africa, and this extends

beyond e-commerce with more features like bill payments, mobile top-ups and financial services such as mini loans. Furthermore, one of the standout features of Konga Pay is its integration with the platform order management system, allowing users to track their payments and orders in real time.

Konga's e-commerce platform is well-designed and focuses on providing a smooth user experience. The e-commerce search platform search functionality is responsive, with quick loading time and really accurate search results. The platform UI design is modern and intuitive, with a layout that makes it easy for users to navigate and find products. Also, it must be noted that integration of Konga pay in the system makes payment easy and secured.

2.5 Conclusion and Comparisons of Systems

From the review of these four e-commerce systems, we can see that each system has its own strengths and weaknesses. Amazon's e-commerce platform stands out for its good search system and customer review systems. Alibaba has an edge over Amazon and other e-commerce platforms with its business-to-business transactions, with strong verifications of suppliers. Jumia, on the other hand, is based on a regional shopping experience with several payment methods, making it convenient to many customers in Africa. Lastly, Konga integrates a good payment system within its e-commerce platform, which, in return, provides a fantastic shopping experience for Nigerian users.

The table below summarizes the performance of each system based on the review criteria above, with each system ranked on a scale of 1 to 5

Table 2.1: System Comparison Result

Review Criteria	Amazon	Alibaba	Jumia	Konga
The Searching of Products and Filtering Quality	5	4.5	4	4
The Customer/User Products Reviews and Ratings	5	3	4	3.5
The Supplier of Products Verification for Customer Satisfaction	N/A	5	3.5	3.5

The Payment Options available for purchasing products	4.5	4	5	4.5
The Performance and the Design Quality of the e-commerce platform	5	4	4.5	4
The User Interface design and the Navigation process of the e-commerce platform	5	4	4.5	4
Mobile Responsiveness	5	4.5	4	4
The Range of Products and Products Availability	5	5	4.5	4
The Security and Privacy Measures of the E-commerce Platform	5	4.5	4	4

The Integration of the e-commerce platform with Third-Party Services	5	4	5	5
The Marketing and Promotion Features on the e-commerce platform	5	5	4	3.5
Total Score	49.5	47	46	43

Based on the evaluation and the analysis of the 4 e-commerce platforms above, several information and conclusions can be drawn. Each of these e-commerce platforms has been evaluated based on the platform's ability to provide the users with quality essential functionalities while using the platform.

Amazon: The Market Leader

Amazon definitely emerges as the leader among these platforms with an overall score of 49.5, which shows its dominance in nearly all aspects of the e-commerce experience. The product search and filtering system is unparalleled, providing users with accurate product search results. Also, Amazon does a great job with its system design and performance section which they score high in. These makes sure that the customer has a good experience. And Lastly, Amazon scored high in the integration with third-party services like Amazon Prime and Amazon Fresh.

Alibaba: Strong in Supplier Verification

Alibaba follows closely behind Amazon, with a score of 47. The e-commerce platform's distinct functionality is its supplier verification process, where it achieved a perfect score. This feature is a very important aspect of Alibaba's system operation which is because it operates a business-to-business marketplace which ensures the credibility of the suppliers. While Alibaba scored well in most areas, it is still slightly behind Amazon in areas like user interface and the integration of the platform with third-party services, but it still provides a solid user experience with a highly functional platform that provides high-quality services to the individual customers and businesses

Jumia: A Strong Contender in Payment Options and User Interface

According to this analysis, Jumia performs well in Africa where it is a major eCommerce platform player with an overall score of 46. Jumia is a cut above the rest of those ecommerce platforms with its payment methods. This is a huge driver in regions such as Nigeria, where paying for items online is fraught with uncertainty and Jumia has been going deep on payment security to make sure that users of the service trust it. The user interface and navigation at jumia.com also compares well in order to provide an effective offering of a platform that is easy enough for people from diverse backgrounds aren't confused or find it difficult.

Konga: Competitive but with Room for improvement

Konga with total score of 43 not in Last place but behind the other 3 top players. The solid integration the platform has with third-party services and payment options have proven to be a standard in running an e-commerce website that will find success among Nigerians. Though the platform function well, there are areas of improvement in terms of intuitive and aesthetically pleasing designs that can be more engaging for user satisfaction.

Conclusion

That said, they are all good to some extent Amazon slightly more so than the rest. For its business-to-business model, Alibaba is effective in supplier verification — a key area in which the B2B company stands out — while Jumia benefits from banks of payment choices provided to users and enticing UI design, making it as accessible to their market space. Konga is competitive as well though it can improve slightly in certain areas to be able to compete with the bigger e-commerce platforms. Ultimately, As the world of e-commerce progresses further forward these platforms will undoubtedly need to adapt and evolve in order to sustain their place and cater for new demand coming from all other customers

Chapter 3 System Requirements

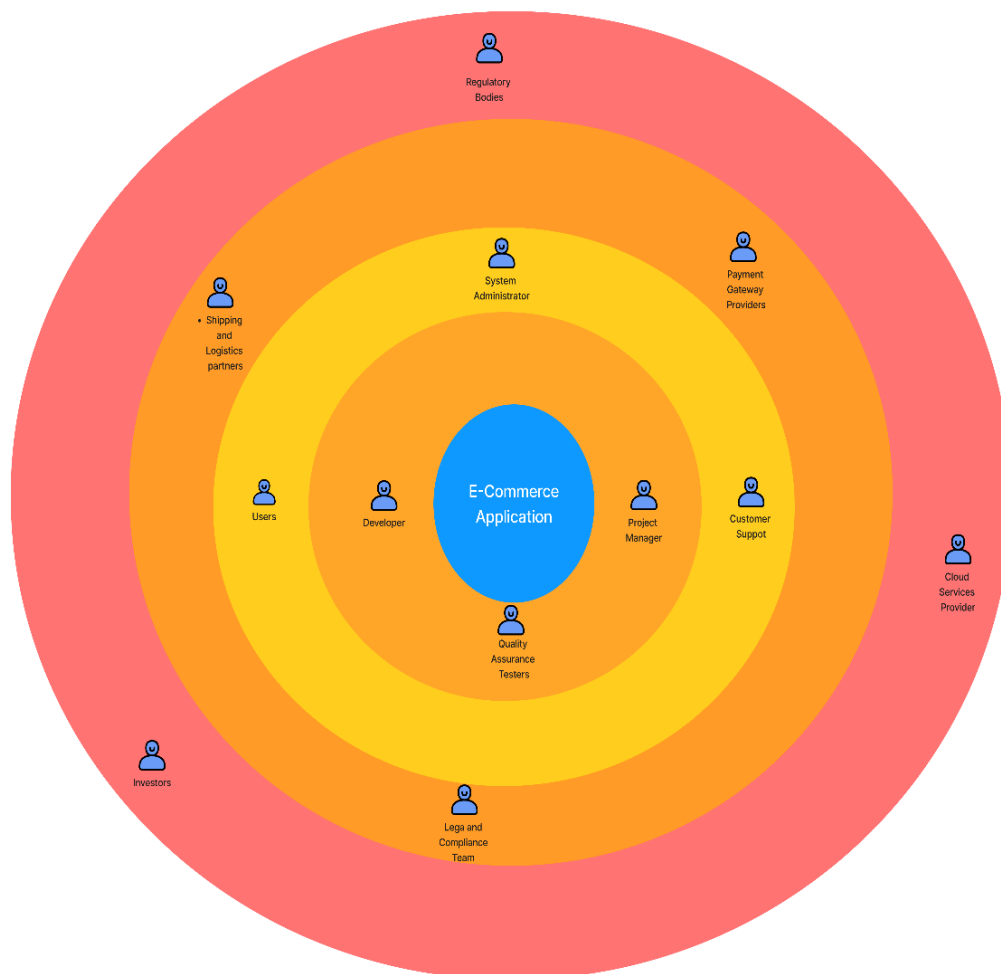
3.1 Introduction

Before we go into the system requirement section of Chapter 3, it is very important to have a clear understanding of the multiple stakeholders that are involved and interactions between the entities and the e-commerce platform. Also, it is important to identify and illustrate the two tools that facilitate this understanding are the Stakeholder Diagram and the Context Diagram

3.2 System Stakeholder

The stakeholder onion diagram, or simply the stakeholder diagram, is a diagram that identifies the entities or groups that are interested in or are in some way affected by the system and puts them into different categories. This visual representation helps in showing the layers of the stakeholders, which ranges from those that are directly involved in building the system to the group that are indirectly affected by the system operations.

Figure 3.1: Stakeholder's Diagram



Here is a breakdown of all the entities/Stakeholders identified in the diagram stakeholder diagram above:

1) Core Layer (System/Project Team)

- **Developers:** The developers are in charge of coding and deploying the e-commerce system
- **Project Manager:** The project manager is the one that oversees the project, ensuring that the project meets the set timeline, the proposed budget and other business requirements
- **Quality Assurance Testers:** The quality assurance tester makes sure that the system is functional and does not have any bug

2) Inner Layer (Direct Interaction with the system)

- **Users:** These are the customers that makes use of the system by browsing and purchasing products
- **Customer Support Team.** They provide help to the user who have complaints about the system and also solve queries
- **System Administrator:** The system administrator manages the everyday operations of the system like account management and troubleshooting

3) Middle Layer (Support and Integration)

- **Payment Gateway Providers:** The payment gateway providers are the external services outside the system that handle payment processing after the user makes a purchase
- **Shipping and Logistics Partners:** The shipping and logistics are responsible for delivering the products ordered by the user
- **Legal and Compliance Team:** The legal and compliance team ensures that the e-commerce system abides by the rules and regulations, especially in data protection, e-commerce law and the rights that protect the user.

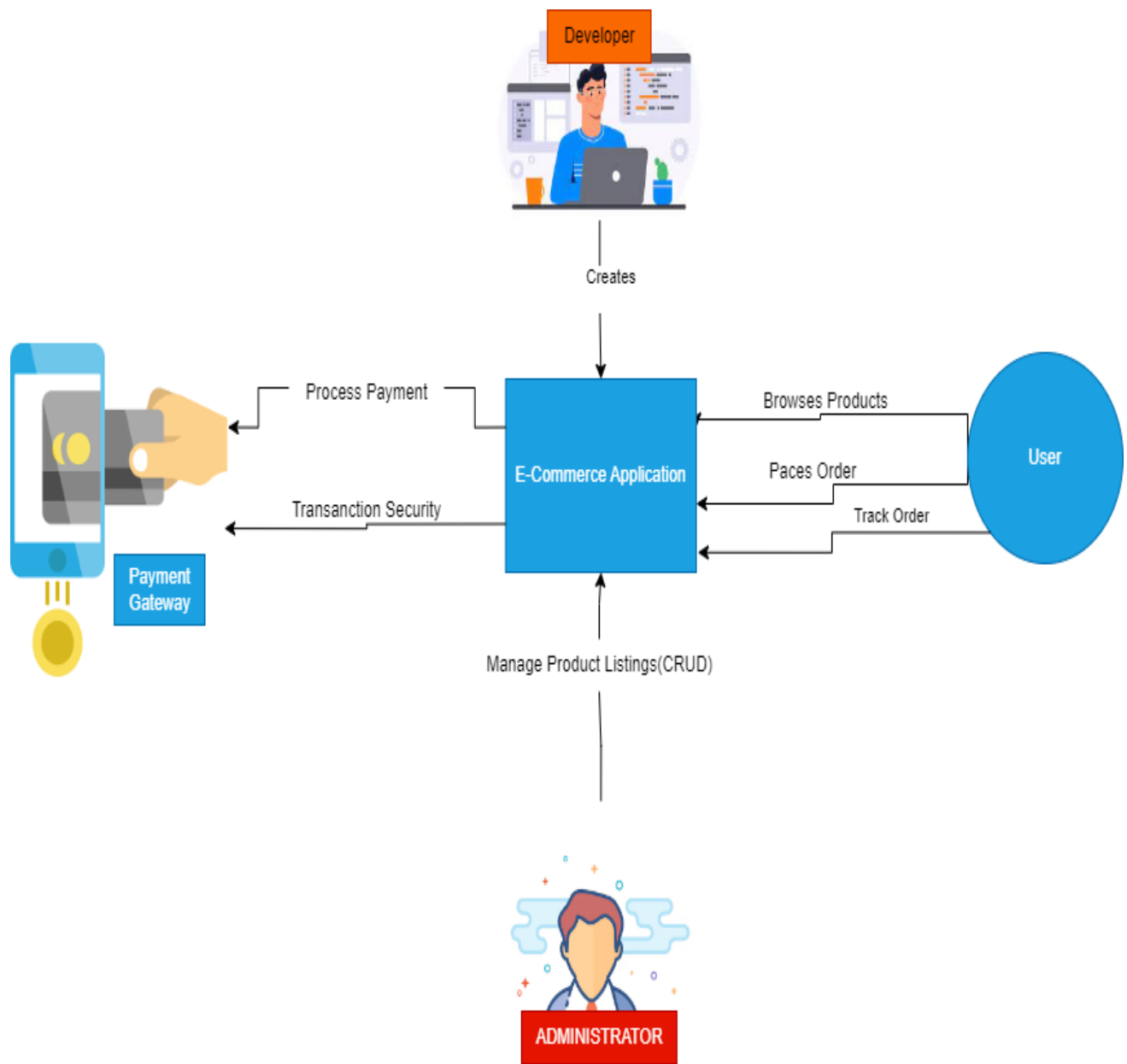
4) Outer Layer (Indirectly Affected Stakeholders)

- **Investors:** These are entities with financial interest in the success of the e-commerce system

- Regulatory bodies: These are government bodies or organizations that set the rules and regulations on data privacy and the platform user protection
- Cloud Service Providers: These are companies that offer cloud technology which can be used to host the e-commerce platform

3.3 System Context

Figure 3.2: Context Diagram



The context diagram above provides a detailed explanation of the key external entities that are involved in this e-commerce system and their interactions. These primary entities are the User, Payment Gateway, Administrator and the Developer.

The Users are the customers that use the e-commerce platform by browsing products, make purchases and manage their account. When a user selects the product of their choice and then proceeds to the checkout, the payment gateway, which is a third-party service, processes the payment and ensures the payment security.

This payment gateway is the third-party service responsible for verifying and completing transactions and then after communicates the outcome back to the e-commerce system. The Administrator plays an important role in managing the e-commerce system activities and operations, which include product listings. The Administrator can perform the CRUD operations, which are to create, read, update and delete within the e-commerce system

The developer is responsible for building and constantly maintaining this e-commerce system and also ensures that it functions efficiently. The developer also manages the backend of the system, which is the database and implements new technologies to ensure that the system has the latest features and is secured

3.4 System Requirements

This section provides a clear overview of the ecommerce system requirements essential in the development of the system. This requirement has three categories, and they are functional, non-functional and the user interface (UI) requirements in which each of these addresses aspects of the system's performance and functionality. It must be noted that each of the categories listed earlier is further divided and categorized into essential, desirable and luxury requirements, which are prioritized in that order.

3.4.1 Functional Requirements

The functional requirements entail the specific behaviors and functionalities that the e-commerce system must have to be able to meet the user needs, and they are:

3.4.1.1 Essential Functional Requirements:

- a. **Product Search:** The e-commerce system must have a search functionality which allows user to find products of their choice easily based on their keyword and the categories of these products. The search functionality should be able to return accurate and relevant results based on the keyword.
- b. **Product Management:** The e-commerce system must have a comprehensive product card that includes the names of the products, descriptions, prices, images and categorization. The admin should be able to perform the crud operation.

- c. **User Account Management:** The e-commerce system should allow the users to create and manage their accounts, including secure authentication.
- d. **Shopping Carts and Checkout:** Users must be able to browse products and also add them to the shopping cart, increase the quantity and also remove the item from the cart. The checkout process should allow the user to review their order information and then complete the payment through the payment gateways
- e. **Payment Gateway:** The application must have a payment gateway for a successful and secured payment by the user of the system

3.4.1.2 Desirable Functional Requirements

- a. **Order Management:** The system must provide the functionality to manage orders, which allows the users to keep track of their product from the moment they place an order.
- b. **Wishlist Functionality:** The system should have the “add to Wishlist” functionality where the user can save the product they are interested in for future reference
- c. **Review and Ratings:** The system should have a feature where the user can leave the reviews and ratings for the products they have ordered and their experience

3.4.1.3 Luxury Functional Requirements

- a. **Product Recommendation:** This is the implementation of a recommendation system that automatically recommends products to the user
- b. **Automated Emails:** The system can have a system that automatically sends email notifications to the user for scenarios like order tracking

3.4.2 Non-Functional Requirements

The non-functional requirements section focuses on the system's performance, security, and user experience.

3.4.2.1 Essential Non-Functional Requirements

- a. **Security:** The system must have reliable security measures in place to protect the data and information of the users
- b. **Performance:** The e-commerce system must have a strong performance percentage that ensures fast loading times and a smooth experience for the user even during high traffic on the platform
- c. **Scalability:** The system should be able to scale easily by being able to handle an increasing number of users and an increasing number of products

3.4.2.2 Desirable Non-Functional Requirements

- a. **Multiple Browser Compatibility:** The system architecture should be compatible with all kinds of browsers as users utilize multiple different browsers
- b. **Responsiveness:** The system must be fully responsive on all kinds of devices

3.4.2.3 Luxury Non-Functional Requirements

- a. **Cloud Deployment:** The system should be deployed on a cloud platform like Microsoft Azure or Vercel to benefit from things like loss of data recovery
- b. **Advanced Security:** The implementation of advanced security measures in the system, such as two-factor authentication for the user's login

3.4.3 User Interface Requirements

The UI requirement ensures that the system is responsive and easy to navigate

3.4.3.1 Essential User Interface Requirements

- a. **Clear Navigation:** The system should have a clear navigation structure which allows the user to easily find products and easily navigate the system.
- b. **Responsive Design:** The user interface design must be responsive, that is, it must be compatible with all devices of various types
- c. **Consistent Design:** The application must have a consistent design across all pages, like uniform fonts and uniform colors

- d. **Error Handling:** Implementation of user-friendly error messages, such as errors when trying to submit a form or when a payment fails

3.4.3.2 Desirable User Interface Requirements

- a) **Quick Access Menu:** These are the shortcuts usually on the homepage, which makes it possible for users to access popular sections, such as new arrivals, easily
- b) **Enhanced Design:** Incorporation of high-resolution images, animation and other appealing visual elements that catches the attention of the user

3.4.3.3 Luxury User Interface Requirements

- a. **Advanced Search System:** Implementation of advance search feature like a voice search and searching with images.
- b. **Display of Interactive products:** This is the use of interactive display elements like 360-degree product visual and zoom features in order for the users to have a better visual of the product.

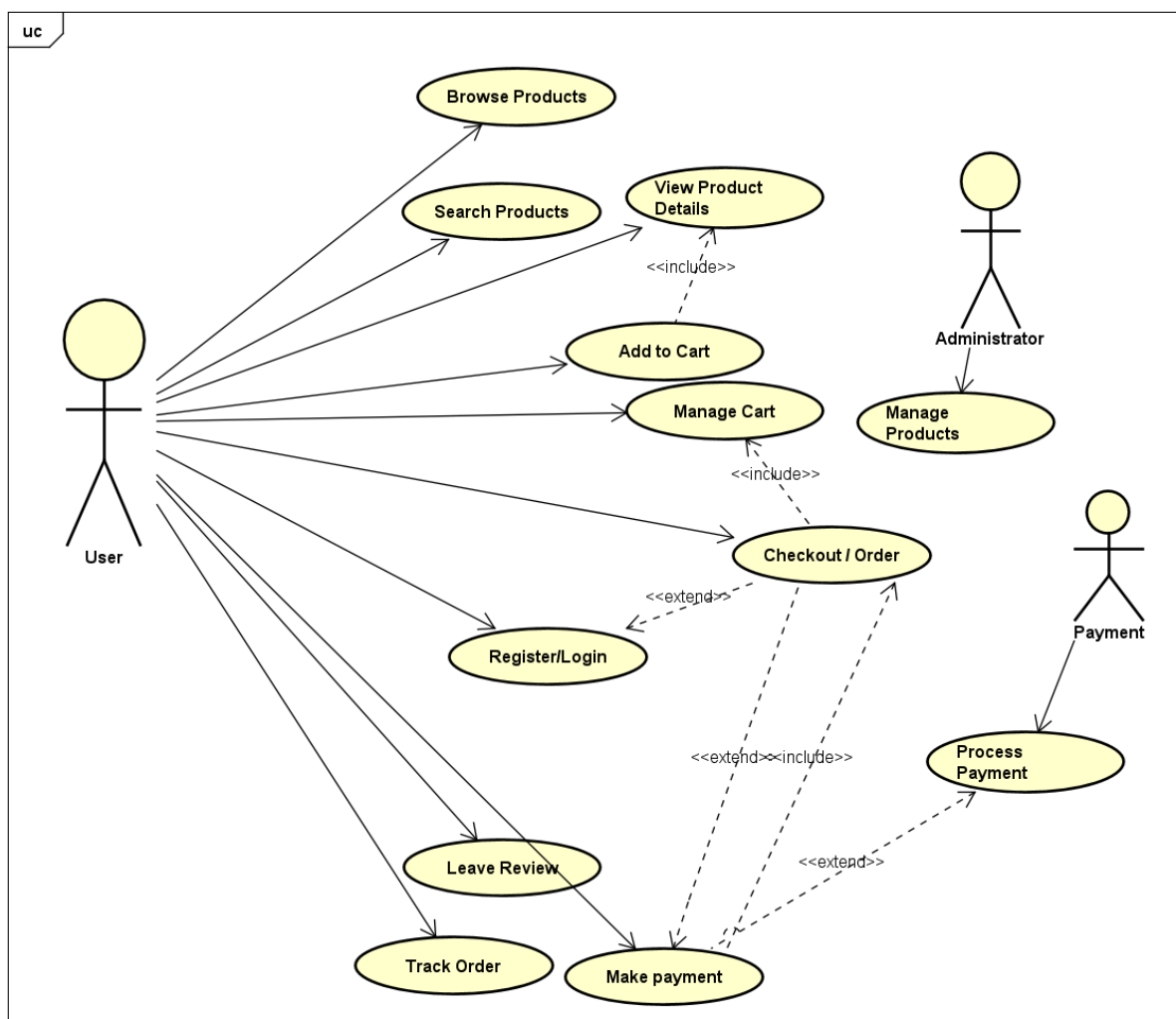
Chapter 4 System Design

4.1 Introduction

This chapter details the e-commerce application system design beginning with a use case diagram which shows examples of how users can engage along with the internal mechanisms at play for each action point. This is then extended to include a more in-depth listing of specific sequence diagrams that explain how each use case can be implemented at the finer-grained level and what system components are responsible for such actions. Furthermore, the class diagram will be illustrated to represent top-level classes that will use in forming connection among those. There you will also learn how the entity-relationship diagram is to be implemented and discussed, relating it somewhat back to existing requirements. Finally, we will develop and study UI frames to show what kind of user interface flow can be expected and used by the users in using this application.

4.2 Use Case Diagram

Figure 4.1: Use Case Diagram



4.3 Use Case Description

A) Use Case: Browse Products

Principal Actor: The User/ Customer

Precondition: The User is on the homepage or category page

Trigger: The User wants to view available products.

Main Success Scenario:

1. The user navigates to the homepage or a specific page in the category
2. The Client-Side sends an http request to the server to retrieve all products
3. The server processes the request and sends it to the database
4. The database returns the list of products to the server.
5. The server sends a list of products back to the client-side
6. The client displays the products to the user.

Alternative Scenarios:

1. **The server fails to connect to the database:**
 - i. The client side will not display any product
 - ii. Return to step 1

B) Use Case: Search Product

Principal Actor: The User/ Customer

Precondition: The User knows the product they want to search for

Trigger: The User enters their search keyword in the search bar.

Main Success Scenario:

1. The user enters a product name or keyword in the search bar and clicks the search button
2. The Client-Side sends an http request to the server with the search term
3. The server processes the request and searches the database for matching products
4. The database returns matching products to the server
5. The server sends the result to the client-side
6. The client displays the search result to the user.

Alternative Scenarios:

- 1 **No matching products are found:**
 - i. The client side displays a message "No Products Found"
 - ii. Return to step 1

C) Use Case: View Product Details

Principal Actor: The User/ Customer

Precondition: The User has found a product they want and they want to learn more about it

Trigger: The User clicks on a product from the product listings or search results

Main Success Scenario:

1. The user clicks on a product
2. The Client-Side sends an http request to the server to retrieve detailed product information
3. The server retrieves the product details from the database
4. The server sends the product details to the Client-Side
5. The server sends a list of products back to the client-side
6. The Client displays the product details to the user, including images, description and price

Alternative Scenarios:

1. The Product details cannot be retrieved from the database:

- i. The client side displays an error message “Failed to load Product details”
- ii. Return to step 1

D) Use Case: Add to Cart

Principal Actor: The User/ Customer

Precondition: The User has viewed the details of a product and wants to purchase it

Trigger: The User clicks the “Add to cart” button

Main Success Scenario:

1. The user clicks the “Add to Cart” button
2. The Client-Side sends an http request to the server with the product details and quantity
3. The server verifies that the product is available and adds it to the user’s cart in the database
4. The server sends a confirmation back to the client
5. The client-side updates the cart user interface to show the added product

Alternative Scenarios:

1. The product is out of stock:

- i. The server sends an error message to the client "Product is out of Stock"
- ii. The client side displays the error message to the user
- iii. Returns to Step 1

E) Use Case: Manage Cart

Principal Actor: The User/ Customer

Precondition: The User added products to the cart

Trigger: The User wants to view, update or remove items in the cart

Main Success Scenario:

1. The user navigates to the cart page
2. The Client-Side sends an http request for the current contents of the cart from the server
3. The server retrieves the cart items from the database and sends them to the client side
4. The client side displays the cart contents to the user
5. The user can update quantities, remove items or proceed to checkout
6. The Client sends updates to the server, which updates the cart in the database

Alternative Scenarios:

1. The user tries to update the quantity of the item beyond the available stock:

- i. The server sends an error message to the client "Request quantity exceeds available stock"
- ii. The client side displays the error message to the user

F) Use Case: Checkout/Order

Principal Actor: The User/ Customer

Precondition: The User has finished managing the cart and is ready to purchase

Trigger: The User clicks the "Checkout" button

Main Success Scenario:

1. The user clicks " Checkout" button.
2. The client sends a request to the server to initiate the checkout process
3. The server retrieves the cart items, calculates the total price, and prepares the order summary.
4. The client displays the order summary and payment options to the user.
5. The user confirms the order and selects a payment method.
6. The server processes the payment and confirms the order.

7. The client displays an order confirmation message to the user.

Alternative Scenarios:

1. **Payment processing fails:**

- i. The client shows an error message: "Payment failed. Please try again."The client side displays the error message to the user
- ii. Return to step 5

G) Use Case: Register/Login

Principal Actor: The User/ Customer

Precondition: The User is not logged in

Trigger: The user wants to access account-specific features like managing the cart or placing an order.

Main Success Scenario:

1. The user is navigated to the login and registration page
2. The user enters their login credentials or registration details.
3. The client side sends the credentials to the server for validation
4. The server verifies the credentials or creates a new account and sends back a response
5. The client side logs the user in and redirects to the homepage or previous page

Alternative Scenarios:

1. **The user enters incorrect credentials:**

- i. The server sends an error message to the client: "Invalid username or password."
- ii. The client displays the error message to the user and returns to step 2.

H) Use Case: Make Payment

Principal Actor: The User/ Customer

Supporting Actor: Payment Gateway, Server

Precondition: The user has confirmed the order and selected a payment method.

Trigger: The user clicks the "Pay Now" button to initiate payment.

Main Success Scenario:

1. The user reviews the order summary and selects "Pay Now" to proceed with payment.
2. The server receives the payment initiation request and prepares the payment details.
3. The server sends the payment request, including order details and payment method, to the Payment Gateway.

4. The Payment Gateway processes the payment and sends a success or failure message back to the server.
5. The client side logs the user in and redirects to the homepage or previous page
6. The server sends a payment confirmation or failure message to the client (user interface).
7. The client displays the payment confirmation and order details to the user.

Alternative Scenarios:

1. Payment Fails Due to Insufficient Funds or Network Issues:

- i. The Payment Gateway returns a failure message to the server.
- ii. The server records the failure and sends an error message to the client.

2. Payment Gateway Times Out:

- i. The Payment Gateway does not respond within the expected time frame.
- ii. The server treats the request as a timeout and logs the error.
- iii. The user is given the option to retry the payment or cancel the order.

I) Use Case: Leave Review

Principal Actor: The User/ Customer

Precondition: The User is not logged in

Trigger: The user wants to access account-specific features like managing the cart or placing an order.

Main Success Scenario:

6. The user is navigated to the login and registration page
7. The user enters their login credentials or registration details.
8. The client side sends the credentials to the server for validation
9. The server verifies the credentials or creates a new account and sends back a response
10. The client side logs the user in and redirects to the homepage or previous page

Alternative Scenarios:

2. The user enters incorrect credentials:

- iii. The server sends an error message to the client: "Invalid username or password."
- iv. The client displays the error message to the user and returns to step 2.

J) Use Case: Manage Products

Principal Actor: Administrator

Precondition: The Administrator is logged in

Trigger: The administrator wants to add, update, or remove products from the catalog.

Main Success Scenario:

1. The administrator navigates to the product management page.
2. The administrator adds, updates, or deletes a product.
3. The client sends the changes to the server.
4. The server validates the changes and updates the database.
5. The client displays a confirmation message to the administrator page

Alternative Scenarios:

1. **The server fails to update the database:**
 - i. The client displays an error message: "Failed to update product information. Please try again later."

K) Use Case: Process Payment

Principal Actor: Payment Gateway

Supporting Actor: User, Server

Precondition: The user has initiated the payment process by submitting the payment request.

Trigger: The server sends a payment request to the Payment Gateway with order details.

Main Success Scenario:

1. The server sends a payment request to the Payment Gateway, including the order details and payment method.
2. The Payment Gateway processes the payment by verifying the payment method and deducting the funds.
3. The Payment Gateway returns a success message along with a transaction ID to the server.
4. The server sends a payment confirmation to the client side (user interface).
5. The client displays the payment confirmation and order details to the user.

Alternative Scenarios:

1. **Payment Declined by Payment Gateway**
 - i. The Payment Gateway returns a failure message with the reason for the decline
 - ii. The server logs the failure and sends an error message to the client.

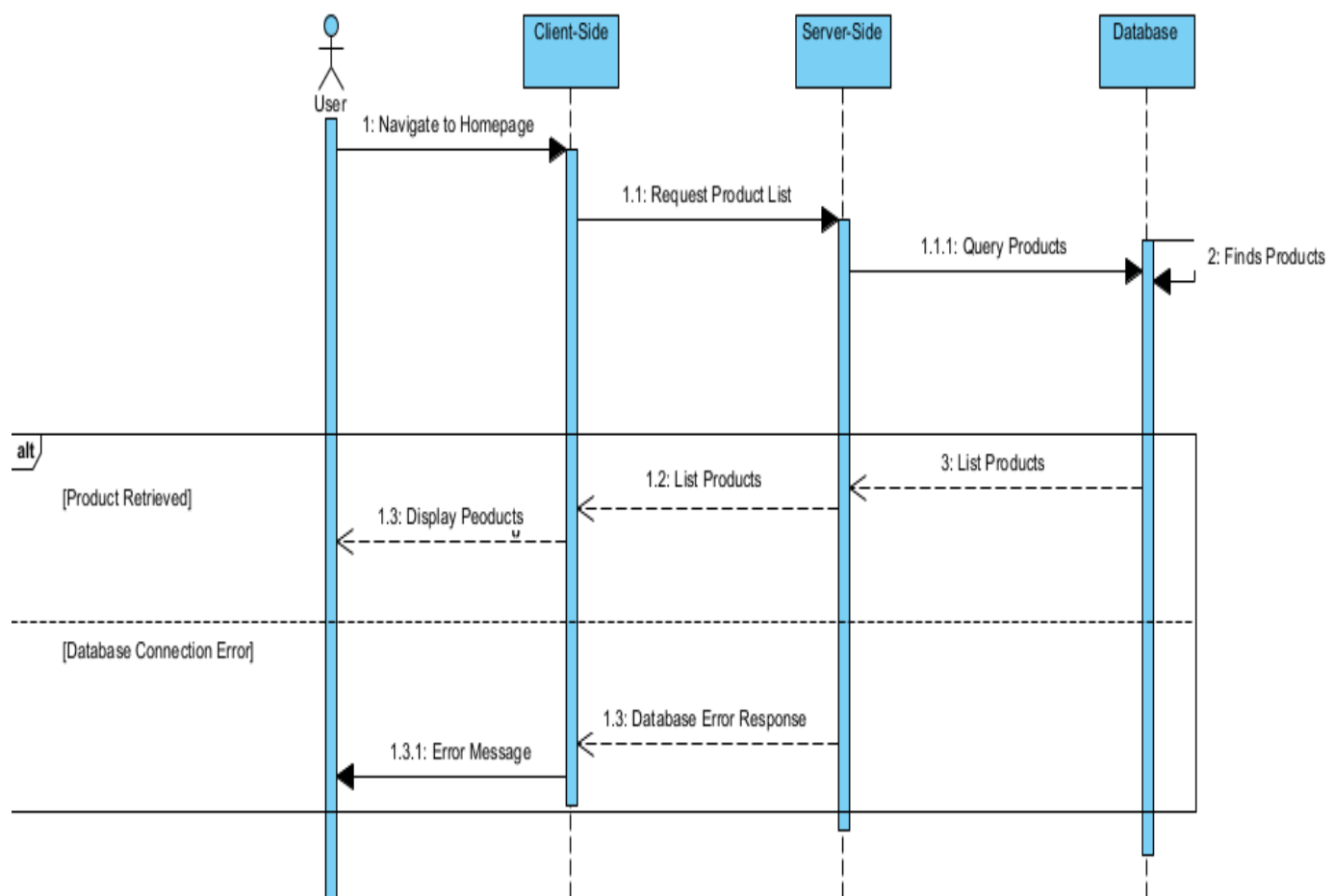
2. Payment Gateway Times Out:

- i. The Payment Gateway does not respond within the expected time frame.
- ii. The server treats the request as a timeout and logs the error.
- iii. The user is given the option to retry the payment or cancel the order.

4.4 Sequence Diagram

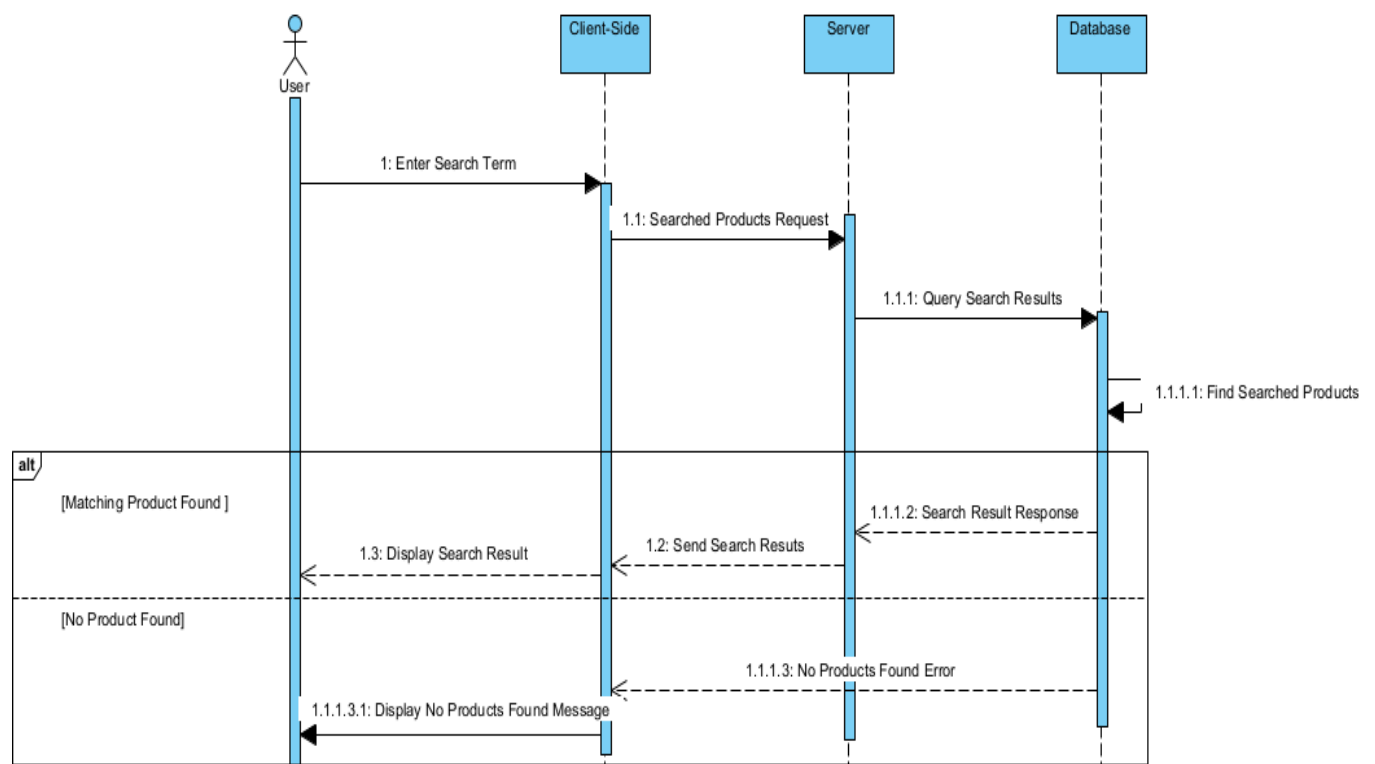
Case 1: Browse Products

Figure 4.2: Sequence Diagram

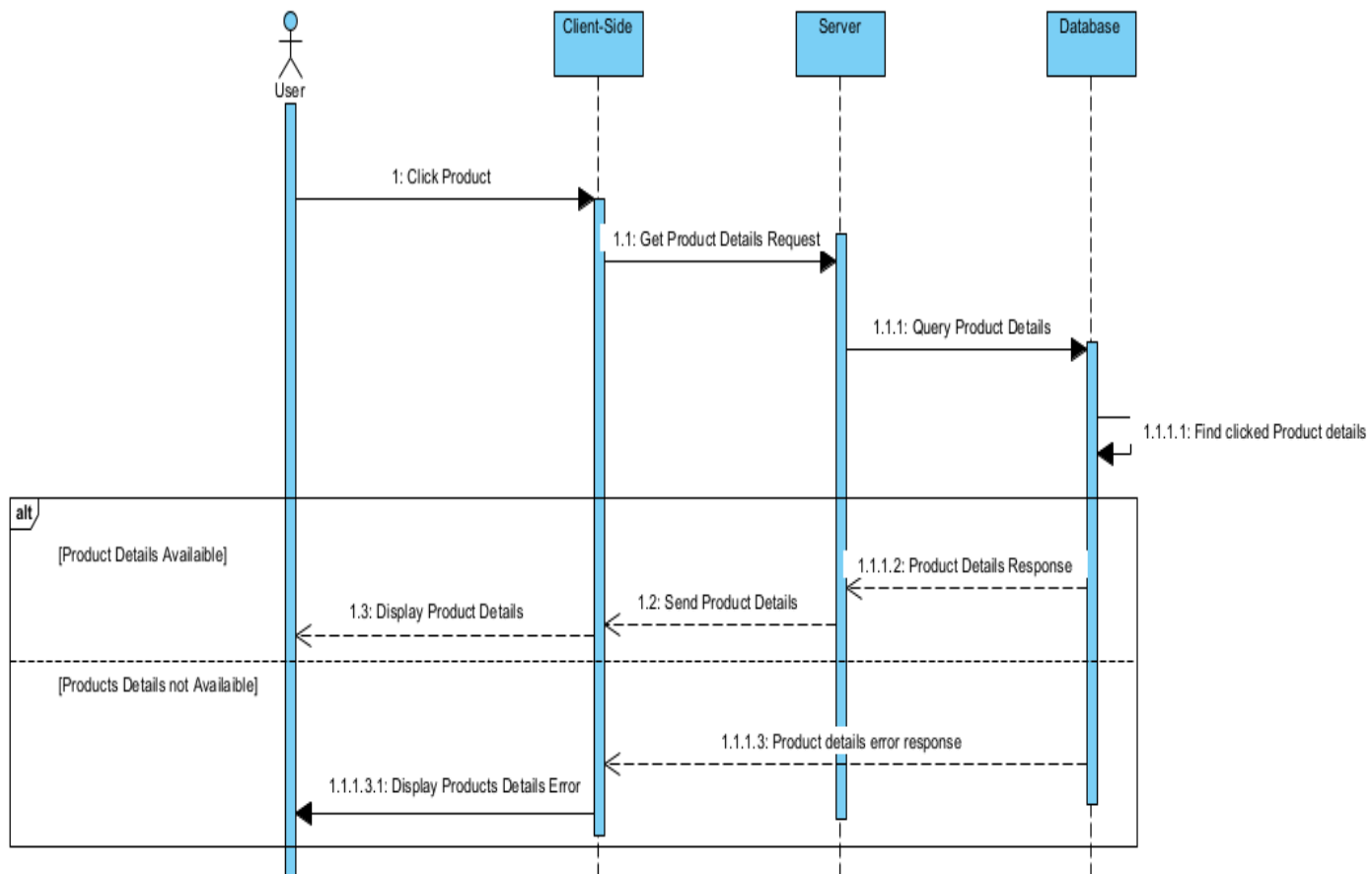


Case 2: Search Products

Figure 4.3: Sequence Diagram

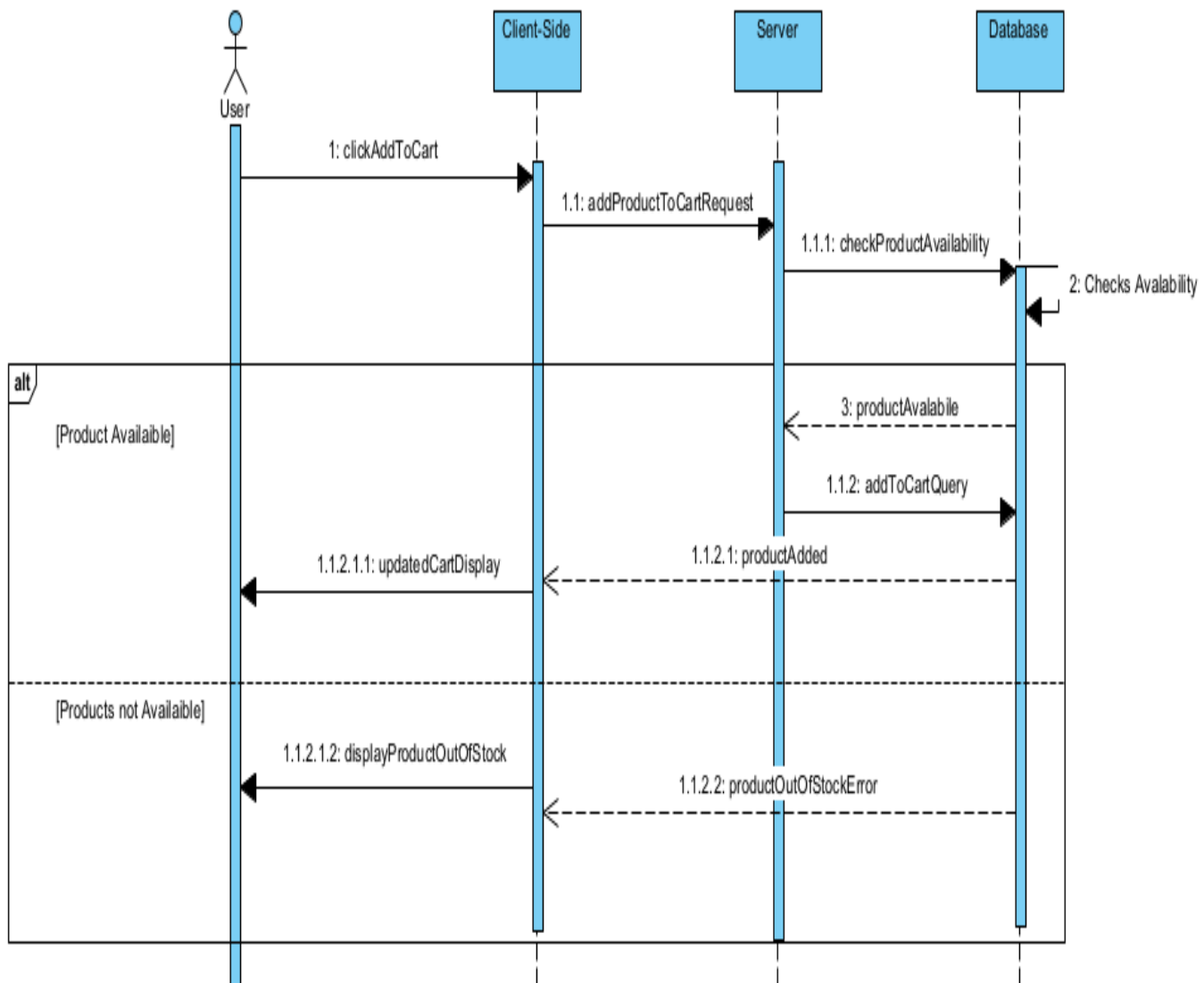


Case 3: View Product Details Figure 4.3: Sequence Diagram



Case 4: Add to Cart

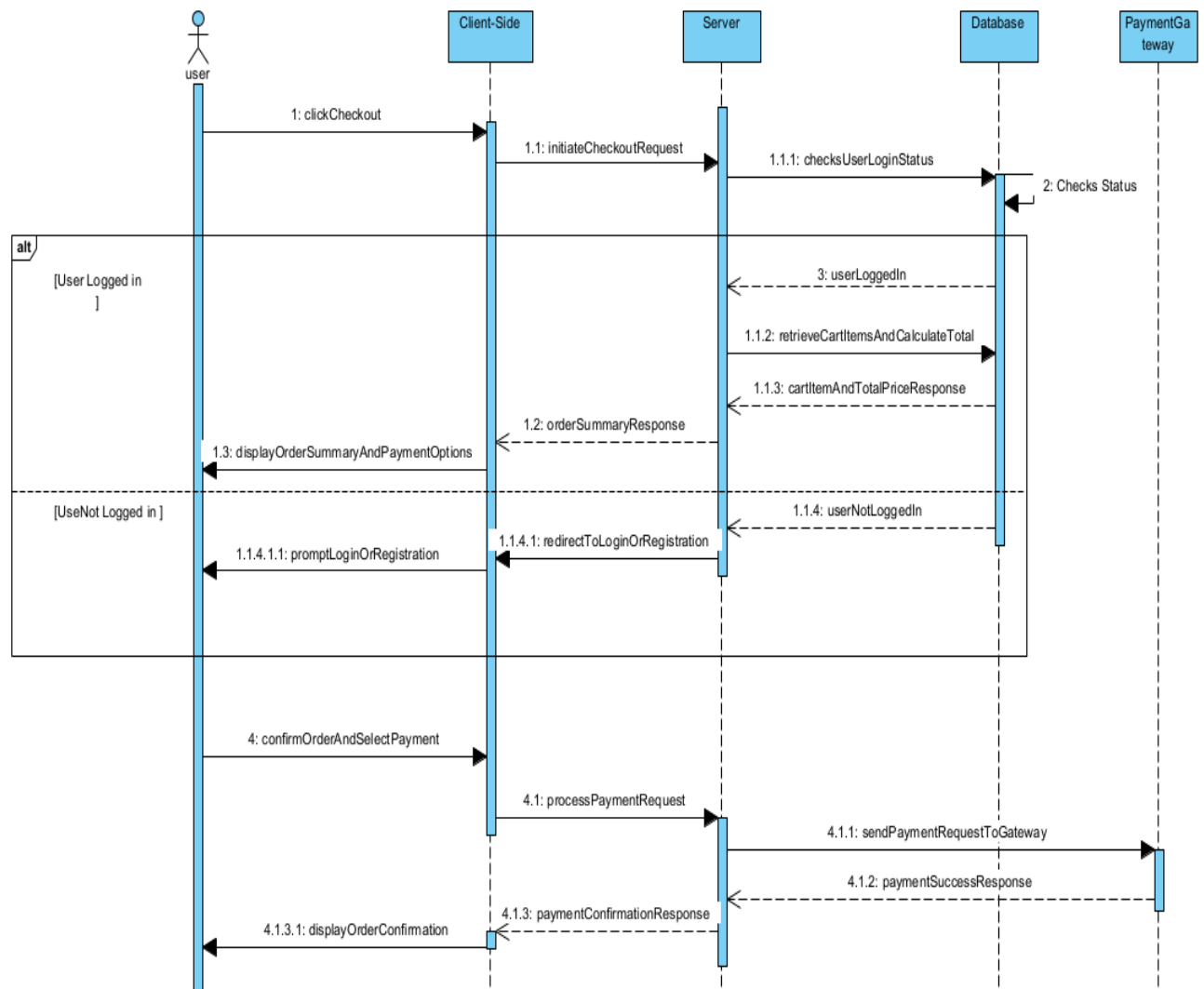
Figure 4.4: Sequence Diagram



Case 5: Check Out/Order

Figure 4.5: Sequence Diagram

sd [Sequence Diagram1]



Case: Register/Login

Figure 4.6: Sequence Diagram

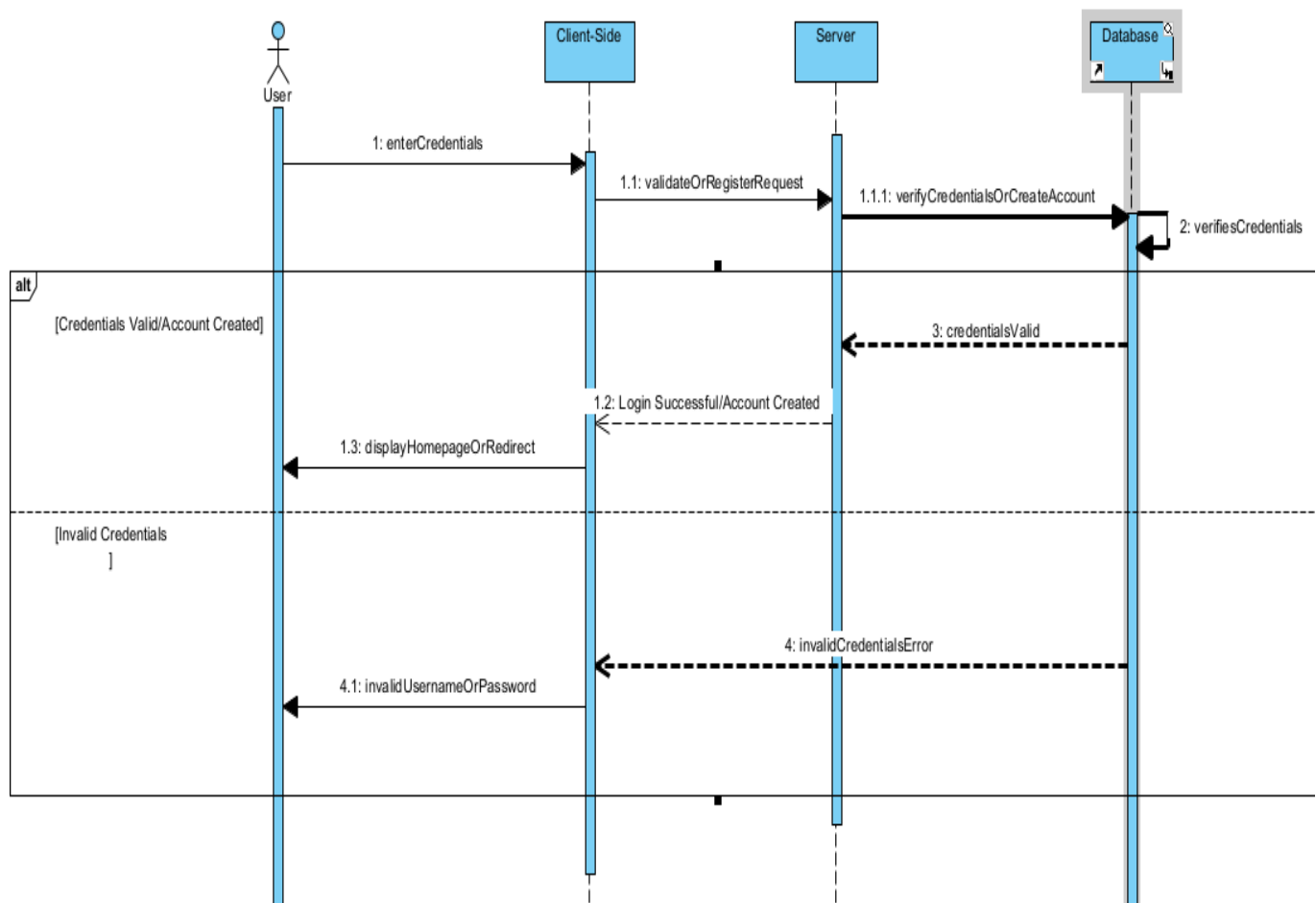
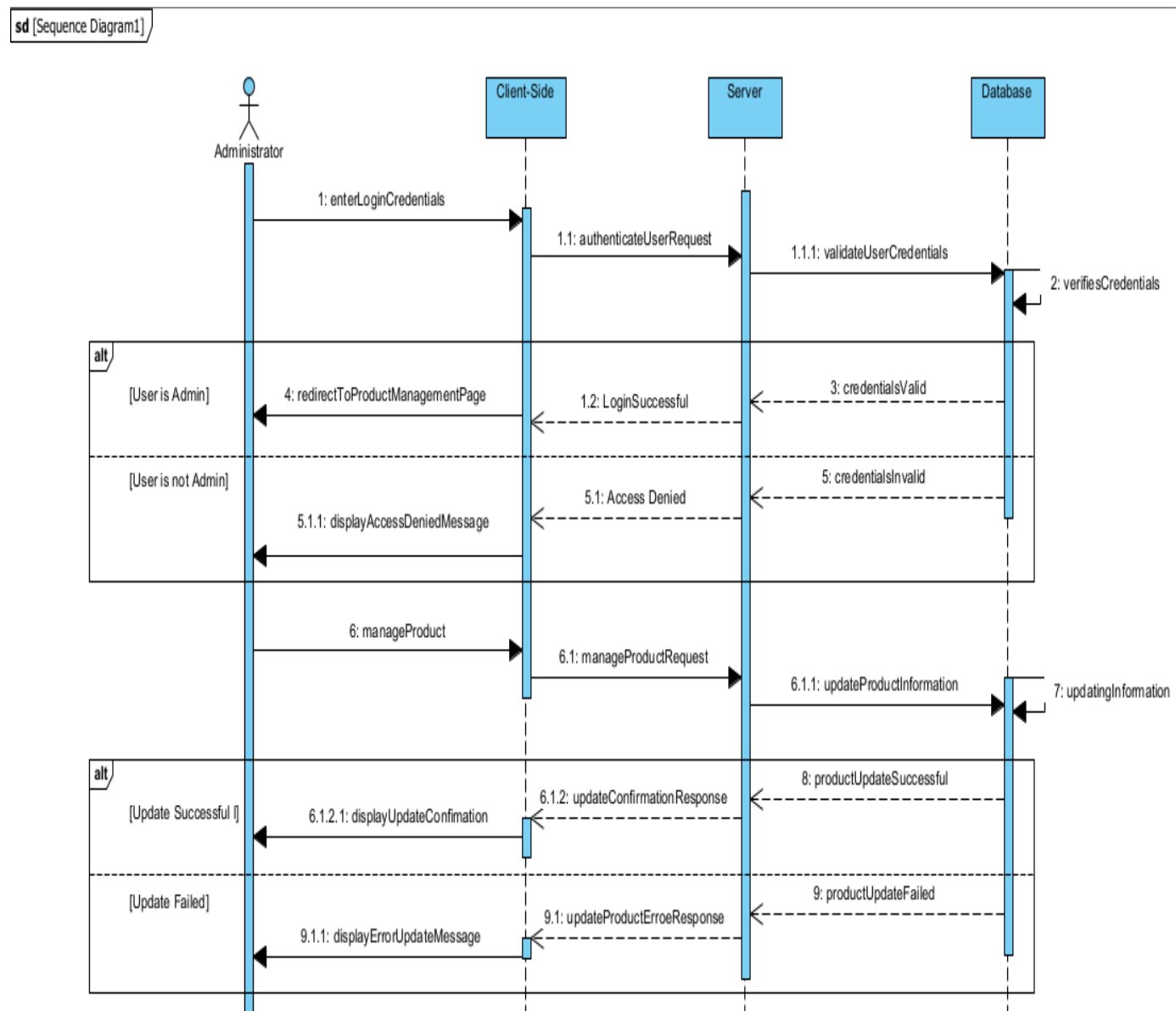
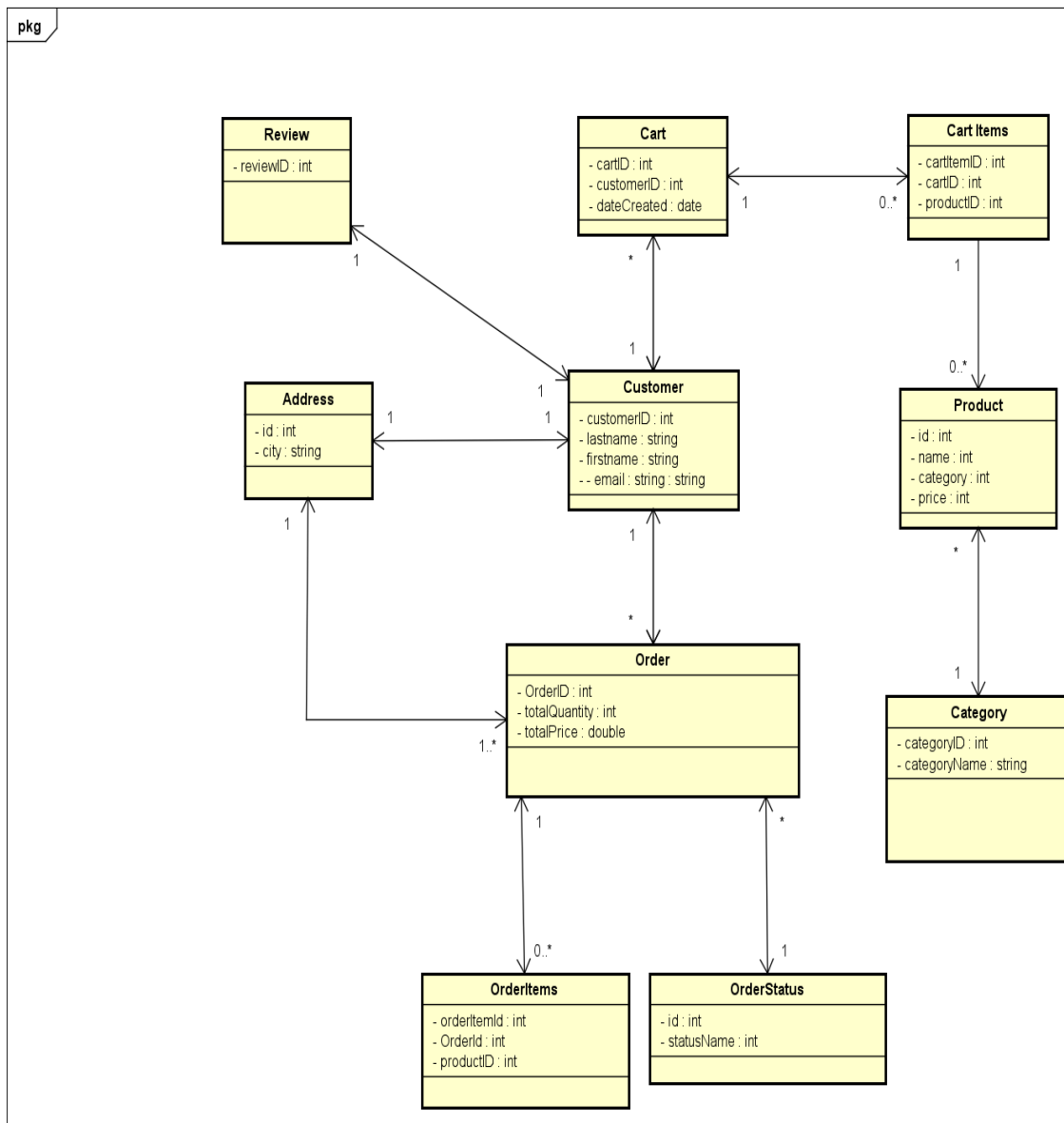


Figure 4.7: Sequence Diagram



4.5 Class Diagram

Figure 4.8: Class Diagram



Cart and Cart Item: There is a one-to-many relationship between Cart and Cart Item. In practical usage, each Cart will have one-to-many amount of Cart Item objects for every product that is added into cart. There is a one-to-one relationship between each Cart Item and its parent Cart, identified by the foreign key annotated in the cartItemID attribute of this bi-directional primary key join column. Using this relationship, we can track multiple products into a single shopping cart and one Cart Item for each product.

Cart and Customer: Cart and Customer is many-to-one relationship. A cart is associated with a unique customer and also one customer can have multiple carts. This kind of setup makes things like tracking shopping history or persistent carts for customers easier.

Product and Cart Item (Many-To-One Relationship): Cart Item and Product have a many-to-one relationship. The Cart Item entity has one ProductId foreign key which references a single related Product and represents the type of product added to cart. A single product can be added multiple times to different carts or even the same cart in this case, and hence, correspondingly have more than one Cart Item entry reflecting these additions.

Category and Product (One to Many): Category to Product is a One-to-many relationship. A Category can have zero or Many Products and this relationship is mapped using the foreign key called as categoryId inside Product entity. It does this by relationship, which means that the same product is organized into categories so users can filter products by category on this ecommerce application.

Customer and order (One-to-Many Relationship): Customer and Order have a one-to-many relationship between them. Many Order entities belong to a single Customer, as shown in each of the Order entity by foreign key column CustomerId. This relationship represents the purchasing history of a customer with each order.

Customer and Cart (One-to-Many Relationship): The relationship between Customer and Cart is a one-to-many relationship. Each Customer can have multiple Carts, which is represented by the CustomerId foreign key in the Cart entity. This relationship allows the customers to manage more than one shopping carts, such as an active cart and saved carts, enabling them to keep track of different shopping sessions or lists.

Customer and Address (One-to-One relationship): Here Customer and Address has one to one relationship A Customer can have one and only one Address referenced by the address id in the customer entity

Orders and Customer (Many-to-One relationship): Order to customer is many-to-one relation. A single Customer can have several Order entities related to it which reflects by adding the foreign key CustomerId in my Order entity.

Order and Order Status (many-to-one relationship): Order is many to one with Order Status. For example, multiple Order entities can be associated with a single Order Status entity through the use of foreign key 'OrderStatusId'. This lets us keep track of an order which is important for customer service or to manage the orders.

Order and Order Item (One-to-Many Relationship): The relationship between Order and Order Item is a one-to-many relationship. Each Order can have multiple Order Item entities, which represent individual products purchased in that order. This relationship is indicated by the OrderId foreign key in the Order Item entity. It allows for detailed tracking of all items within an order, including quantities and individual prices, supporting accurate billing and inventory management

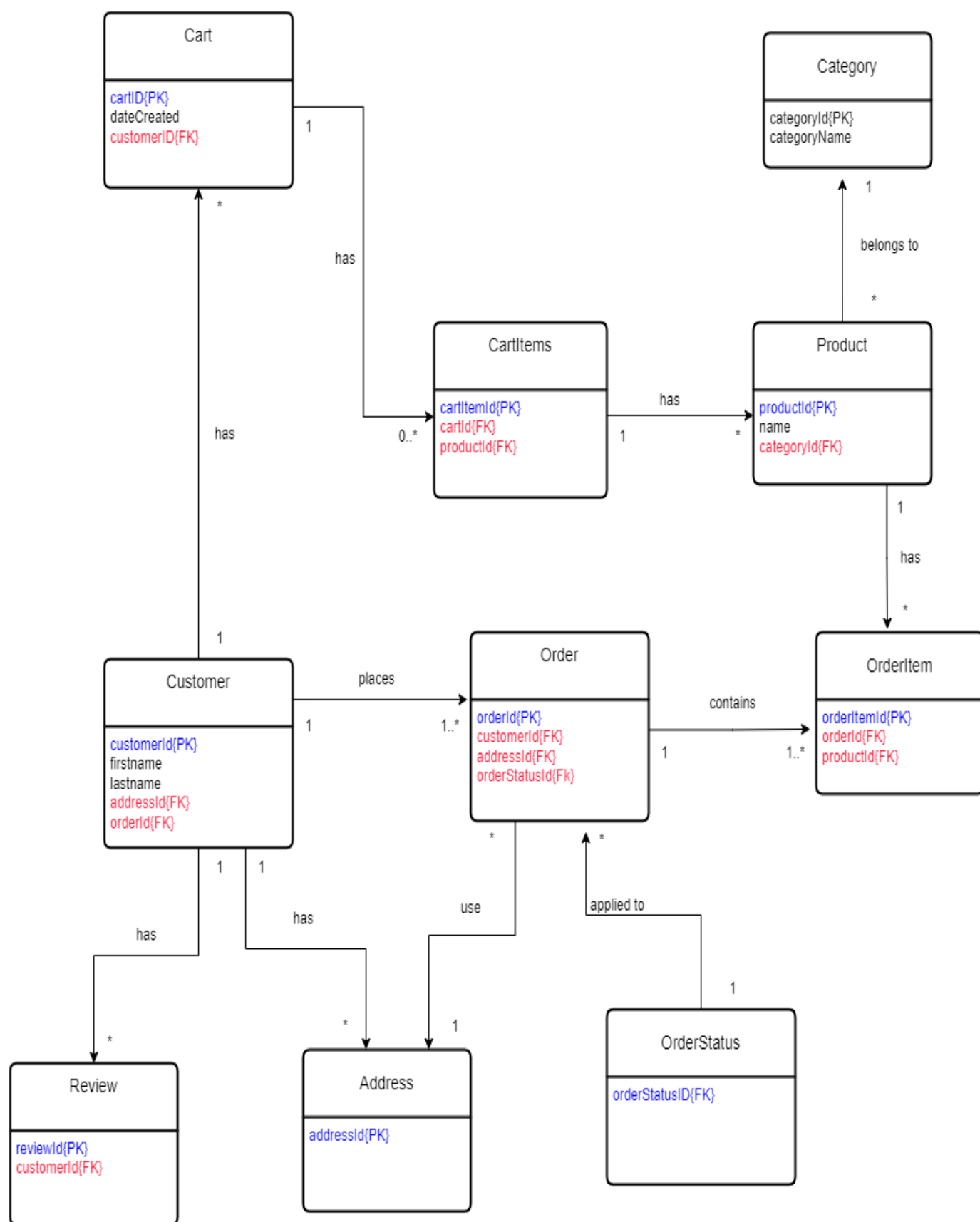
Order Item and Product (Many-to-One Relationship): The relationship between Order Item and Product is a many-to-one relationship. Each Order Item is linked to a single Product through the ProductId foreign key, representing the product being purchased. Multiple Order Item entries can refer to the same Product, reflecting its sale in different orders or multiple instances within the same order.

Order Status and Order (One-to-Many Relationship): The relationship between Order Status and Order is a one-to-many relationship. Each Order Status can be associated with multiple Order entities, representing the status of these orders at a given time. This relationship allows for the management and tracking of order progress through various stages, from placement to delivery.

Product and Cart Item (One-to-Many Relationship): The relationship between Product and Cart Item is a one-to-many relationship. Each Product can be associated with multiple Cart Item entities, which shows that a product can be added to more than one carts. This way a single product to appears in the shopping cart of different customers, which shows a real-world scenario where many customers are interested in purchasing the same item.

Product and Order Item (One-to-Many Relationship): The relationship between Product and Order Item is a one-to-many relationship. which shows that each Product can have more than one Order Item entities associated with it, which represents instances of the product being purchased in different orders.

4.6 Entity Relationship Diagram



Cart: The cart entity represents a shopping cart where customers add their products before purchasing it. It has a primary key cart Id which uniquely identifies each cart. Each cart is associated with a specific customer through a foreign key Customer Id which is a primary key on the customer table, establishes a many-to-one relationship with the customer entity. Also, there is a one-to-many between the cart and the cart item entity which implies that a cart can contain multiple cart items and the relationship is illustrated by adding the foreign key cart Id to the Cart Item table

Cart item: The cart item entity represents the individual item that the customer adds to their cart while shopping. The cart item entity is related with the cart entity through a foreign key cart id which indicates a many-to-one relationship. Also, cart item has a many-to-one relationship with the product entity which is also represented by a product id foreign key on the cart item table which indicates that each cart item is a reference to a specific product

Category: The category entity represents the various product categories that are available on the ecommerce application. It has a primary key category Id which uniquely identify each category and there is a one-to-many relationship between the category and the product which means that a category can have lots of products and also this relationship established by adding foreign key category Id in the product table and linking it back to category Id in category

Customer: The customer entity stores the information about the user that uses the application and it has a customer id primary key which identifies each customer. There is a one-to-many relationship between the customer and order with the foreign key customer Id in the Order table which allows for a customer to have multiple orders. Also, there is a one-to-one relationship between the customer and the address entity, represented by the foreign key address id on the customer table which implies that a customer can have one default address

Order: The Order entity represents a customer's purchase transaction. It has a primary key Order Id to uniquely identify each order. There is a many-to-one relationship between Order and Customer, with the foreign key Customer Id in Order linking back to Customer Id in Customer, indicating that multiple orders can belong to one customer. The Order entity also has a many-to-one relationship with the Address entity and there is a many-to-one relationship with the Order Status entity, using the foreign key Order Status Id, indicating that multiple orders can share the same order status. Furthermore, Order has a one-to-many relationship with Order Item, with the foreign key Order Id in Order Item linking back to Order Id in Order, indicating that one order can contain multiple items.

Order item: The Order Item entity represents individual products included in an order. It has a primary key Order Item Id to uniquely identify each order item. Order Item is associated with the Order entity through a foreign key Order Id, establishing a many-to-one relationship where many order items can belong to one order. Additionally, Order Item has a many-to-one relationship with the Product entity, using the foreign key Product Id, signifying that each order item is a reference to a specific product.

Order status: The order status entity represents the different status of the orders placed by the customers such as pending or delivered. It has a primary key Id to identify each of the status and there is a one-to-many relationship between order status and order with the foreign key order status Id in the order table linking back to in the order status which shows that one status can be assigned multiple orders

4.7 GUI Design

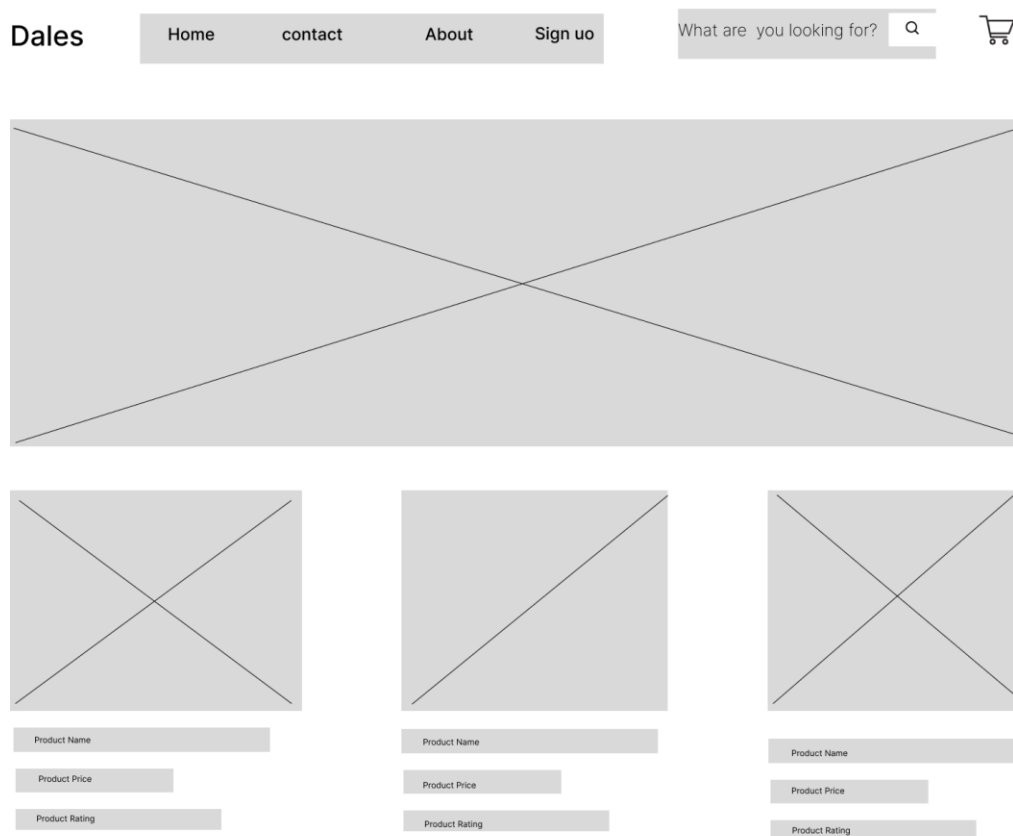
The implementation of this UI design was all about having a clear, functional and pleasant design that will allow users to click around the application. The design aimed to meet the needs of the users by offering a simple visually engaging experience. Below are the principles and guidelines adhered to:

1. **Usability:** Ensuring that the design of the application makes the application easy to use and to understand
2. **Consistency:** This is maintaining a uniform design across all the pages of the application
3. **Responsiveness:** It was very important to make sure that the application functions well across different devices and screen sizes.

Wireframes Low Fidelity

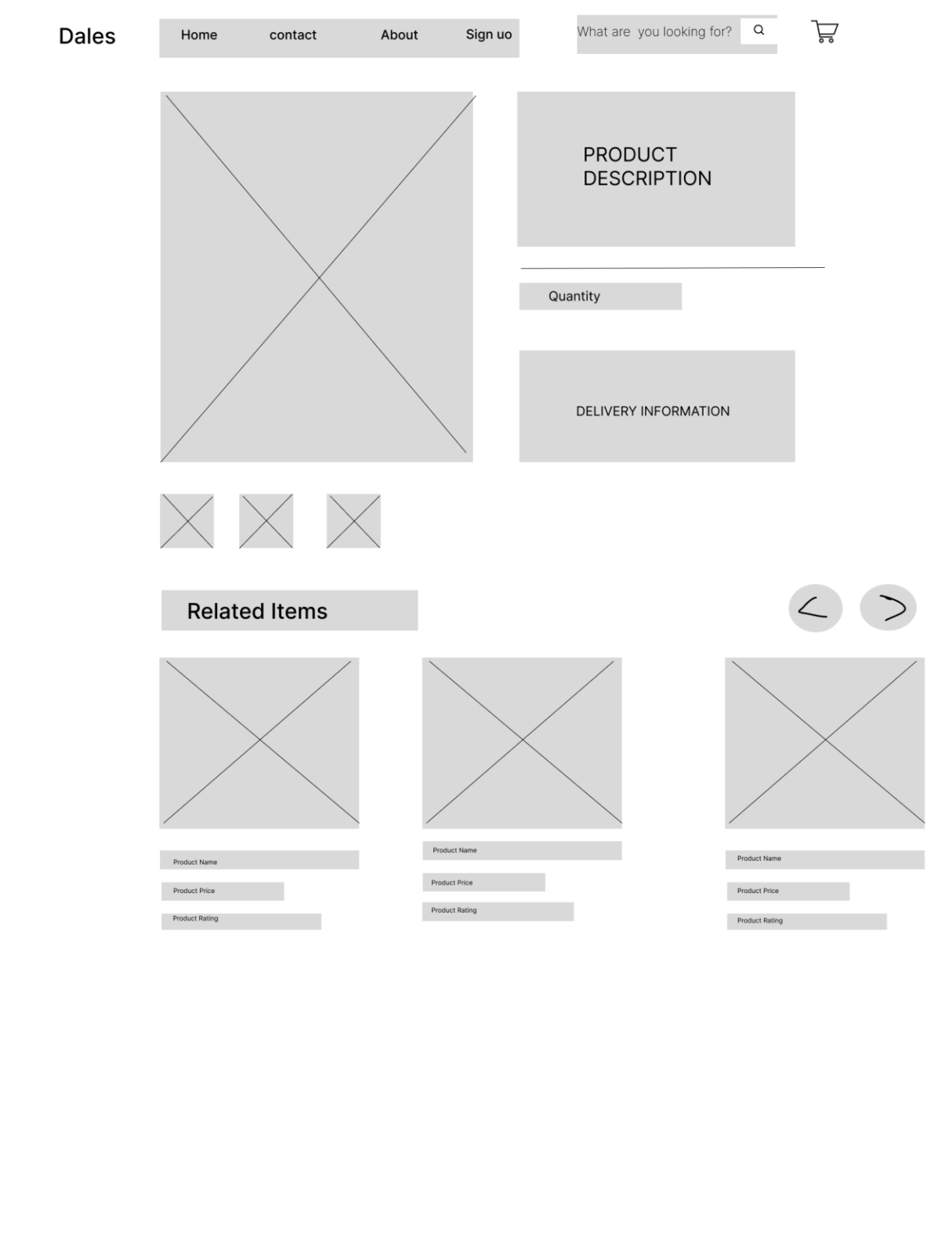
Homepage

Figure 4.10: Wireframe Low Fidelity



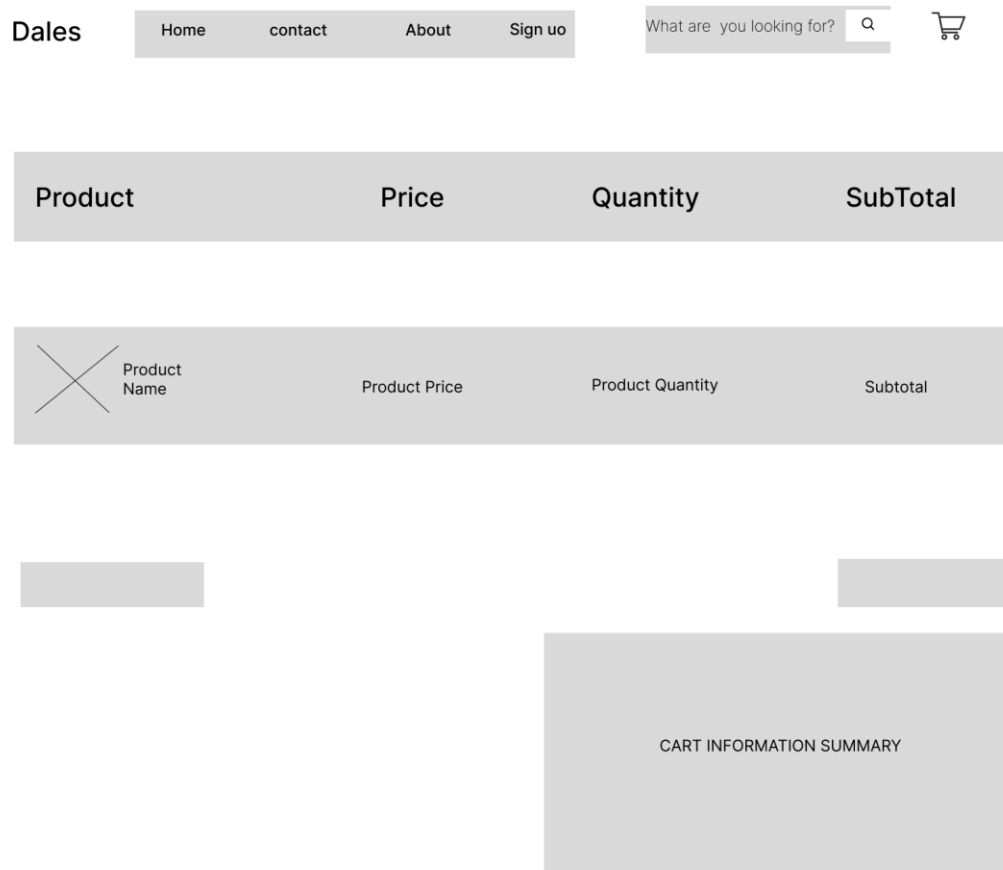
Product Description

Figure 4.11: Wireframe Low Fidelity



Cart

Figure 4.12: Wireframe Low Fidelity





Checkout

Figure 4.13: Wireframe Low Fidelity

Dales

HomecontactAboutSign uo

What are you looking for? 



Address

First Name

Last Name

Street Address

Phone Number

Email Address

SubTotal

Shipping


Total


Place Order

Login Figure 4.15: Wireframe Low Fidelity

Dales

HomecontactAboutSign uo

What are you looking for? 



Login

Enter Your Details Below

Email Address


Password


Login

Sign Up

Figure 4.14: Wireframe Low Fidelity

Dales Home contact About Sign up

What are you looking for? 



Create an Account

Enter Your Details Below

First Name

Last Name

Email Address

Password

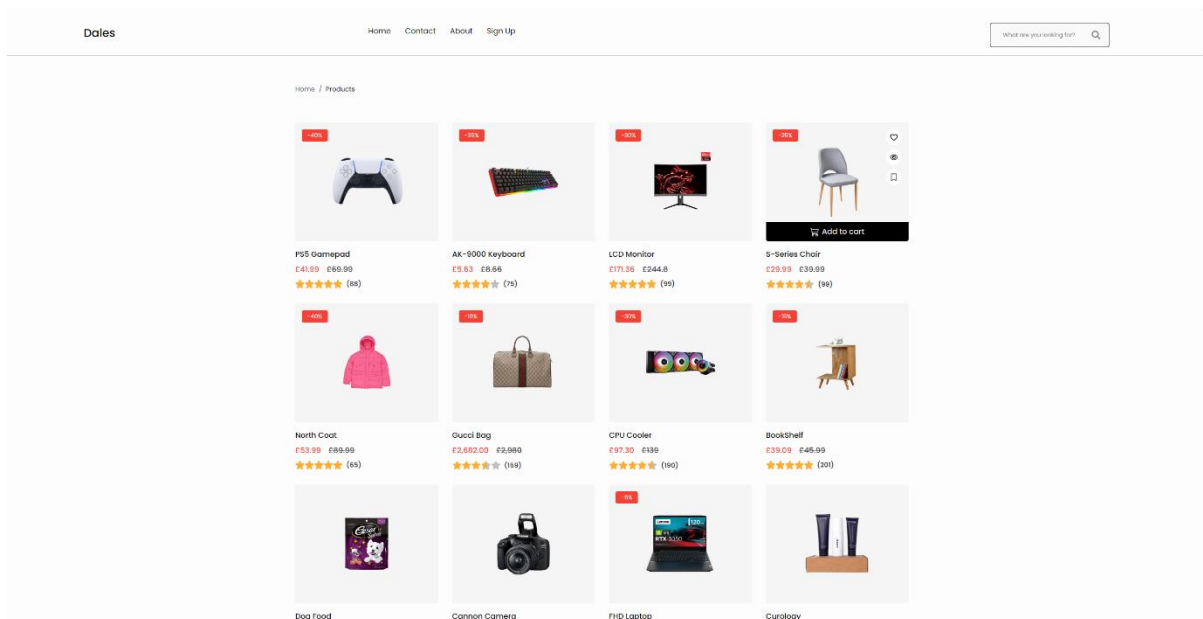
Sign Up

Wireframes High Fidelity

Design Pages based on Use cases above:

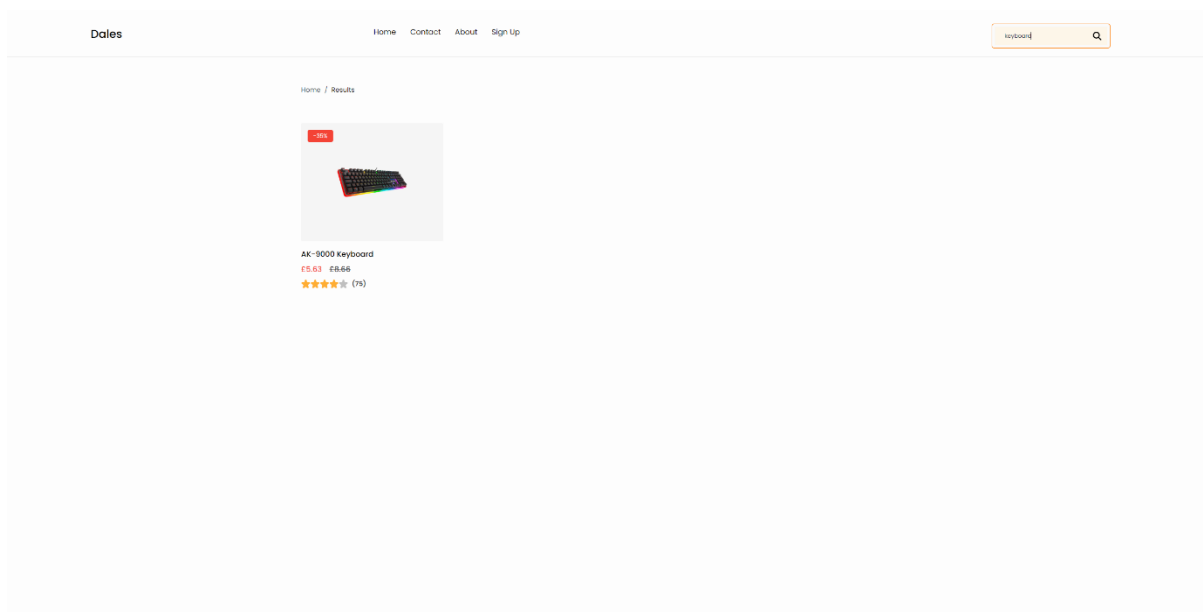
Use Case1: Browse Products

Figure 4.16: Wireframe High Fidelity



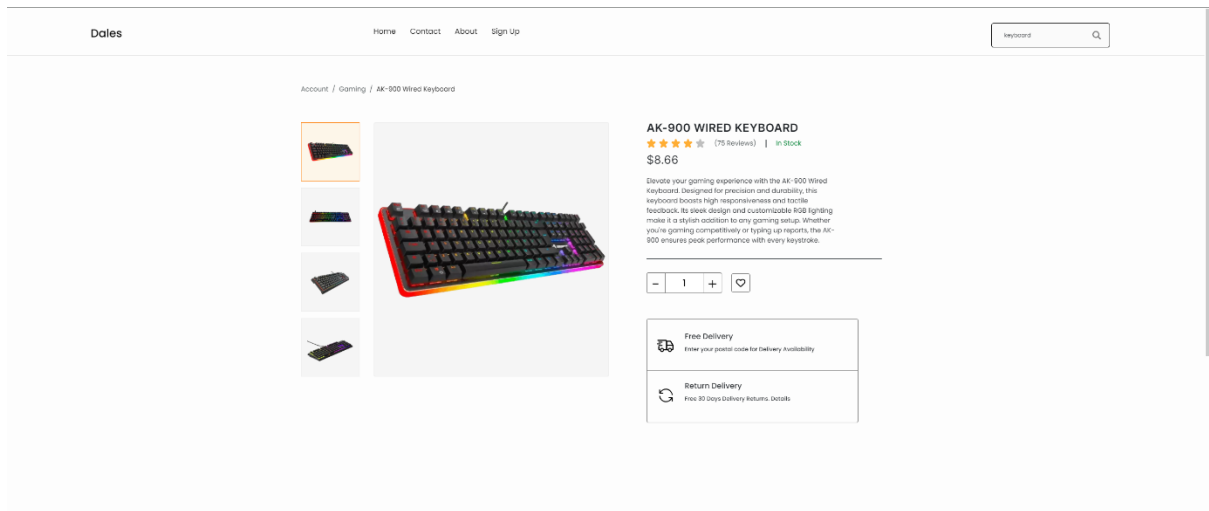
Use Case 4: Search Product

Figure 4.17: Wireframe High Fidelity



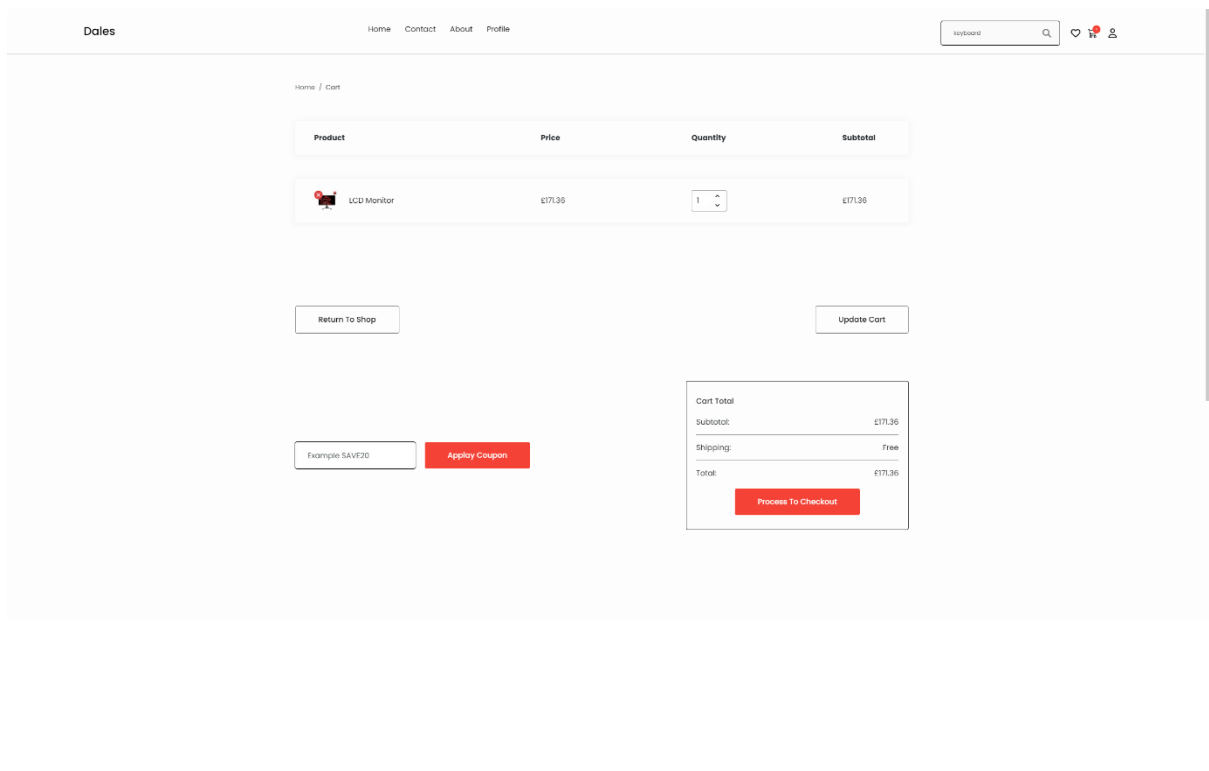
Use Case 5: View Product details

Figure 4.18: Wireframe High Fidelity



Use Case2: Add Product to cart

Figure 4.19: Wireframe High Fidelity



Use Case 3: Checkout/Order

Figure 4.20: Wireframe High Fidelity

Dales

HomeContactAboutProfile

keyword

Account / Checkout

Billing Details

First Name*

Company Name

Street Address*

Apartment, floor, etc. (optional)

Town/City*

Phone Number*

Email*

☐ Save this information for faster check-out next time

LCD Monitor

\$171.38

Subtotal

\$171.38

Shipping

Free

Total

\$171.38

☐ Cash on Delivery

Place Order

57 | Page

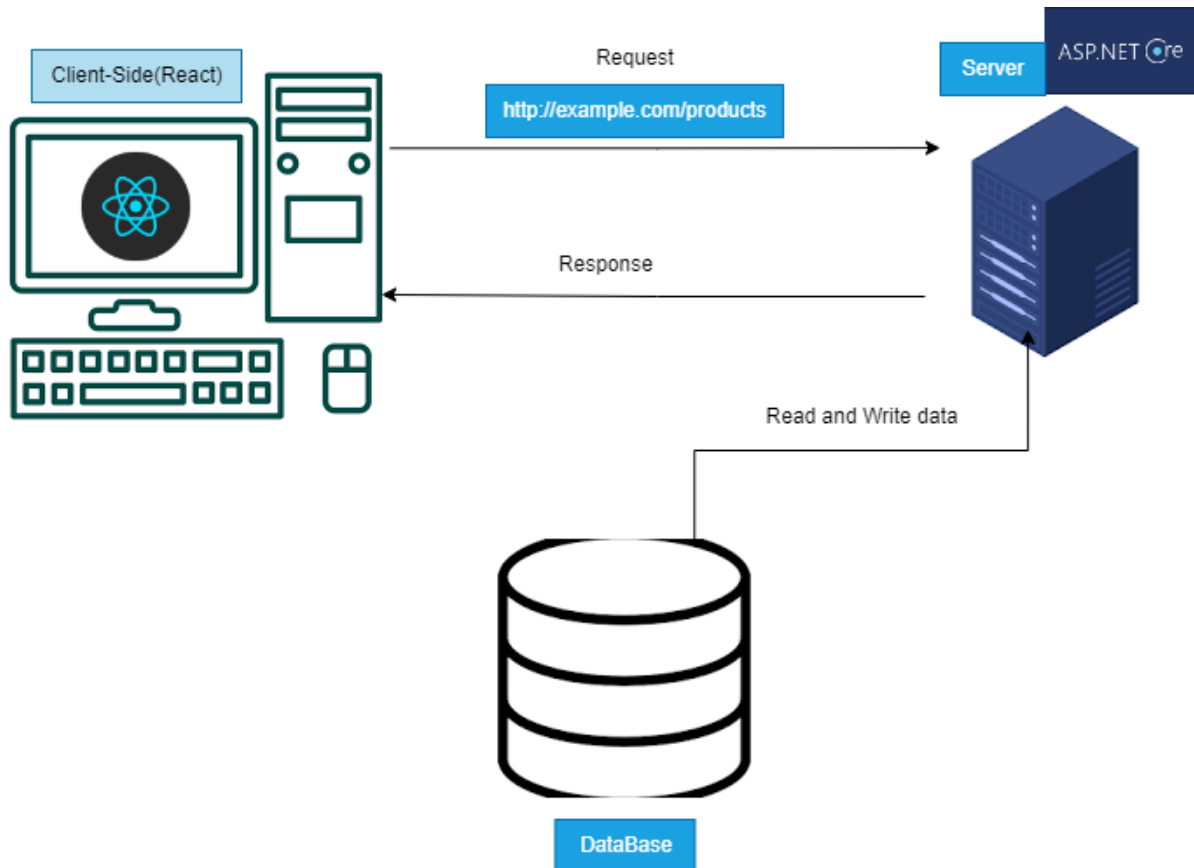
Chapter 5 Implementation

5.1 Introduction

This chapter is concerned with the implementation of the e-commerce system and more information about how the system has been built. To begin with, the first analysis will be on the system architecture giving an overview of how the application is designed and structured generally. In this, we will analyze each part along with their functioning in the overall course of action as a whole providing an uninterrupted user interface. Then we will discuss the technologies that were used in this project. This part will cover some of the possible technologies explored, the major advantages and constraints they presented as well as outline our thought process guiding into a final decision on what to use. This part should help explain why those technologies were used instead of and how they contribute to the app depth or speed. Finally, we will talk over how the application is being part actualized on server-side and client side. Here you will find some design decisions, coding practices and integrations strategies that helped in developing a native application. The aim is to provide a systematic breakdown of all components and how they are function together for an efficient e-commerce platform.

5.2 System Architecture

Figure 5.1 system architecture



The architecture of the whole system includes a client-side application which was built using react and also the server side based on Asp. NET Core Web API. The main front-end of this system is an application built with React, which has the responsibility to render user interface React is built around a component-based architecture which means the user interface can be broken down into reusable pieces components. These components are self-contained and work together to create a rich user experience. .NET Core Web API is running in this system as the backend. A RESTful API for this would deal with HTTP requests driven by the React application and return relevant responses. There are a wide variety of endpoints that the React application interacts with through HTTP methods such as GET, POST, PUT and DELETE. Each and every Endpoint represent the specific business function to retrieve data, creating new records in a table and update existing data. There is also the creation of models in the backend, these models will be stored in our database and on which we can interact with. For example, since this system works with users, products or orders then each of these entities has been represented by a model. These models are often classes that describe characteristics of the entity and how it is related to other entities. Communication between the React frontend and ASP. NET Core Web API has a request-response pattern. When a user does an action on the frontend like submitting a form or clicking any button then react triggers API which results into an HTTP request, which may contain the necessary data. The ASP. NET Core Web API takes the request and interacts with database to get

some data or store state and sends HTTP response. It can then respond with data and confirmation that the operation succeeded. In summary, the system architecture uses React to form a responsive and reactive interface for the users. On the server side, all logic and data processing are done accordingly with NET Core Web API using it partially also to interact with a database. The request-response pattern ensures that user actions on the frontend are accurately reflected in the backend, providing a seamless user experience.

5.3 Review of Technologies

During the development of my eCommerce application, I decided to use a set of technologies that would give me powerful, responsive and scalable system. For the front end I used HTML, CSS, JavaScript and sass.

I used HTML as the structural markup language to structure my application by defining elements such as headers, paragraphs, forms and more. I used CSS to style these HTML elements in order for the application itself to look good and uniform on all devices as well as across different modern browsers. I enhanced the CSS by using Sass, which is a CSS preprocessor that makes writing more maintainable and reusable styles easier. As the app grew, I felt much more capable of handling large stylesheets due to features like variables and nesting in Sass. It was the main script I used to incorporate interactivity and dynamic behavior into my application. JavaScript was the core scripting language, it allowed me manage user events, handle the DOM and interact with backend services. Actually, instead of pure JavaScript I chose React which is a popular library for developing UI. With React I could utilize the component-based architecture to break user interfaces into self-contained, reusable components. This not only modularized and organized my codebase, but also improved performance by ensuring changes were committed to the DOM in just one render through React's virtual dom. I used Redux in order to manage the global state of my application. Redux was especially helpful with managing all the state interactions across multiple components. Centralizing my state and logic made my app a lot more maintainable as well since there are less chances for bugs.

On the backend, I used C# and ASP.NET Core to build a robust API that serves as the backbone of my application. C# is the most popular programming language with full data type support and OOP functionality. C# is a versatile and powerful programming language that integrates seamlessly with the .NET ecosystem, making it an excellent choice for this project. ASP.NET Core which is a truly cross-platform framework and it let me build web API from scratch in no time, with high performance to process business logics, for serving client requests and communicating database as well. I appreciate ASP.NET Core has a good combination of modularity and lightweight, which made it my preferred choice.

To store data, I used the SQLite which is a small and lightweight, serverless database engine that allows access to other databases or RDBMS in development. I mean, it was really simple to setup and use SQLite for the database in my project while building. I interacted with the database via Entity Framework, an object-relational mapper (ORM) which provided me to work with my database using C# objects and simplified access to data reducing boilerplate code. During the process of developing this I used Visual Studio Code for front-end, and visual studio itself for back-end development. I also loved working with JavaScript, React or Sass in Visual Studio Code thanks to its flexibility and huge list of extensions. While Visual Studio provided a powerful, integrated environment for ASP.NET Core and C# development, helping me maintain a productive workflow and ensuring the project was built with best practices in mind.

5.4 Implementation Issues and Solution

There were important implementation challenges during the development of this e-commerce application. These challenges ranged from data security to the choices made in design, integrations with external services and performance optimization. In the following section we will discuss these issues in more detail and how to overcome them by various strategies used.

5.4.1 Data Security and Encryption

Issue: Data security was one of the biggest concerns, especially because of the sensitive information being handled in the application, such as user credentials, payment details and the customer's personal data. Additionally, securely storing passwords and ensuring safe authentication practices were critical components of this aspect.

Solution: To address these concerns, several measures were implemented:

1. **SSL/TLS Encryption:** All the data that is transferred between the client-side and server-side were transmitted using SSL/TLS encryption which makes sure the data is safe while transferring. The ASP.NET Core Web API framework was configured to use HTTPS connections which means that all the communication between the client-side and the server-side are encrypted.
2. **Hashing and Salting Passwords:** Passwords of the users were not saved in plain text into the database. Instead, passwords were hashed using a strong algorithm. By implementing it this way, even if the database is compromised or attacked the actual password would still remain secured
3. **Token-Based Authentication:** To handle the user sessions securely, I used a token-based authentication which is JSON Web Tokens. When the user logs in successfully, a token is created and will be used for making further requests in the application. It is more secure than session management which is the traditional way and provides defense against a lot of types of attacks including session hijacking.
4. **Data Encryption at Rest:** All confidential information kept in the database, like payment info, was encrypted by powerful encryption methods (AES-256). This step ensures that if the attacker still manages to access my database, then also he cannot get anything from it.

These strategies made the application very secure and eliminated data breaches or unauthorized access.

5.4.2 Design and Choice of Algorithms and Data Structure

Issue: Picking the best algorithms and data structures were necessary to make my app performant especially for dealing with potentially huge datasets like product catalogs or the users information. The challenge was to balance the readability and maintainability with the performance of the application, especially in areas such as search functionality and state management.

Solutions: Several key decisions were made to address these challenges

1. **Efficient Search Algorithms:** Optimized search is done in product search by indexing. The backend API used an indexed search on the database, ensuring quick retrieval of results even with a large number of records. Also, a mechanism was implemented on the frontend to reduce the number of API calls when users type in the search bar rapidly.
2. **State Management with Redux:** We used Redux for controlling the Application State, because it is predictable and yields easy debugging. However, to avoid performance problems, state

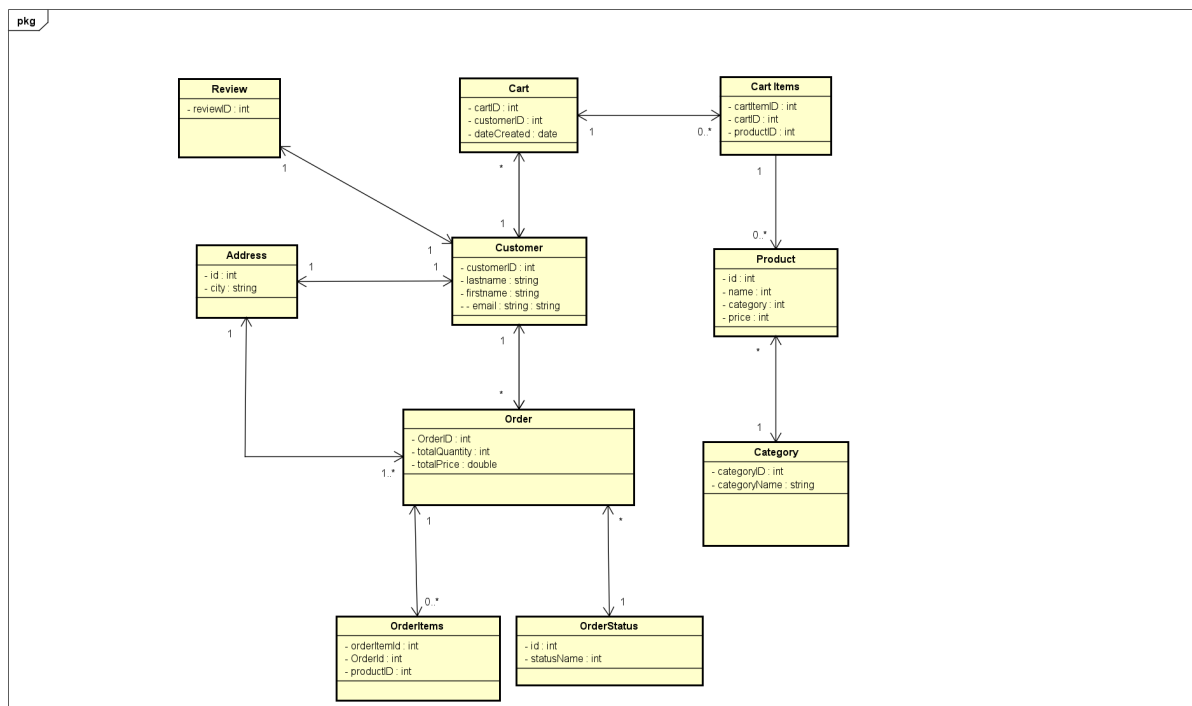
slices were designed to make sure that only necessary components re-render when the state changes.

3. **Data Structures for Fast Access and Updates:** The application used data structures like maps and sets to ensure fast access and updates within the application. For instance, the product data was stored in a map structure to allow O (1) time complexity for lookups, which is very important for a responsive user experience.

By selecting algorithms and data structures, the application maintained high performance, and also easy to manage and improve in the future

5.5 Implemented Class Diagram

Figure 5.2 Implemented Class Diagram



5.6 Overview of Software Components

This e-commerce application used lots of React components for the frontend and ASP.NET Core controllers for the backend. All of these components and controllers played a important role such as handling products, authentication and the shopping cart. The client-side components made use of React hooks and Redux for the management of the application state, while the server-side made use of Entity Framework Core to carry out database operations and ASP.NET Identity for authentication of the users. This design allows for easy maintenance and scalability of the application.

React Frontend Component

1. **Products Page Component:** The Products Page component fetches and displays a list of products. It handles the loading state and controls the display of products or loading indicators based on the fetch status.
 - **Data Structures & Methods:**

State:

- products: Array to store fetched product data.
- Loading Products Page: Boolean to indicate if the products page is loading.

Methods:

- fetch Products: Asynchronously fetches product data from the API.
- use Effect: Triggers fetch Products on component mount.
- use Dispatch, use Selector: Hooks from Redux for state management.
- use Translation: Hook for internationalization.
- Use Update Loading State: Custom hook for updating loading state.

Pseudo Code

```
• function ProductsPage () {  
•   state: products = [], loadingProductsPage = false  
•  
•   onComponentMount () {  
•     setLoadingState(true)  
•     fetchProductsFromAPI ()  
•   }  
•  
•   function fetchProductsFromAPI () {  
•     try {  
•       response = fetch (apiUrl + "Products")  
•       if response is not OK then throw error  
•       data = response. Json ()  
•       setProducts(data)  
•     } catch (error) {  
•       logError(error)  
•     } finally {  
•       setLoadingState(false)  
•     }  
•   }  
•  
•   return (  
•     Render Helmet with title and meta description  
•     Render main container with:  
•       PagesHistory component  
•       Conditional rendering based on  
loadingProductsPage:  
•       if not loading:  
•         ExploreProducts component with products  
•       else:  
•         SkeletonCards component  
•     )  
•   }
```

1. **Cart Component:** The Cart component provides an interface for users to review and manage the items in their shopping cart. It includes options for checkout and applying coupons.

Data Structures & Methods:

Methods:

- Use Translation: Hook for handling translations.

Pseudo Code

```
• function Cart () {  
•   translateFunction = useTranslation ()  
•  
•   return (  
•     Render Helmet with title and meta description  
•     Render main container with:  
•       PagesHistory component  
•       CartProducts component  
•       CartButtons component  
•       Wrapper containing:  
•         AddCoupon component  
•         CartInfoMenu component  
•   )  
• }
```

2. **Cart Product Component:** Purpose: The Cart Product component displays detailed information about a single product in the cart, including an image, price, quantity, and subtotal.

- Data Structures & Methods:

State:

- Product Image: URL for the product image.

Methods:

- Use Translation: Hook for translations.
- Use Effect: Hook for fetching product image.
- Fetch Product Image From API: Fetches and sets the product image URL.

Pseudo Code

```
function CartProduct ({data}) {  
  
  productImage = null  
  
  productData = data  
  
  calculateSubTotal ()  
  
  onComponentMount () {
```



```

    if (data.id exists)
        fetchProductImageFromAPI ()
    }
}

function fetchProductImageFromAPI () {
    try {
        response = fetch (apiUrl + "imagedata/product/" + data.id + "/main-image")
        imageBlob = response. blob ()
        productImage = createObjectURL(imageBlob)
    } catch (error) {
        logError(error)
    }
}

return (
    Render table row with:
        Product image and link
        Product price
        Quantity input
        Subtotal
)
}

```

ASP.NET Backend Controller

1. Account Controller: The Account Controller manages user authentication tasks, including registration, login, and email verification. It handles JWT token generation and user sign-in/sign-out processes.
 - **Data Structures & Methods:**

Methods:

 - **Register:** Registers a new user and sends an email verification link.
 - **Verify Email:** Confirms the user's email address using a token.
 - **Login:** Authenticates the user and returns a JWT token.
 - **Logout:** Signs out the user.
 - **Generate Jwt Token:** Creates a JWT token for authenticated users

Pseudo Code:

```

function Register(model) {
    user = createNewUser (model. Email)
    result = saveUser (user, model. Password)

    if result is successful:
        token = generateEmailVerificationToken(user)
        sendVerificationEmail (user. Email, token)
        return SuccessMessage
    else:
        return ErrorMessage
}

function VerifyEmail (userId, token) {
    user = findUserById(userId)
    if user exists:
        result = confirmUserEmail (user, token)
        return result? SuccessMessage: ErrorMessage
    else:
        return NotFoundError
}

function Login(model) {
    result = signInUser (model. Email, model. Password)
    if result is successful:
        token = generateJwtToken(user)
        return {Token: token}
    else:
        return Unauthorized Error
}

function Logout () {

```

```
    signOutUser ()  
    return SuccessMessage  
}  
  
function GenerateJwtToken (user, roles) {  
    claims = create Claims (user, roles)  
    token = createJwtToken(claims)  
    return token  
}
```

5.6 Conclusion

This chapter examined the implementation of some critical components within the application, like the Products Page, Cart, Cart Product, and the Account Controller. The Products Page component manages the display and loading of product data, using React hooks for state management and dynamic updates. The Cart component provides an interface for users to manage their cart items, including options for checkout. The Cart Product component handles individual cart item displays and use image fetching to enhance the user experience. On the backend, the Account Controller manages essential authentication functions like user registration, login, and email verification, employing the JWT token functionality too. Together, these components illustrate a well-integrated system where frontend and backend functionalities work together to provide a good experience to the user.

Chapter 6 Testing

6.1 Introduction

This chapter will cover all the testing strategies which i opted for in order maintain quality and reliability of this e-commerce web application. The Testing phase is an important part of a software development life-cycle, which includes finding & rectifying defects and checks whether application/solution works as designed or not.

6.2 Tests Conducted:

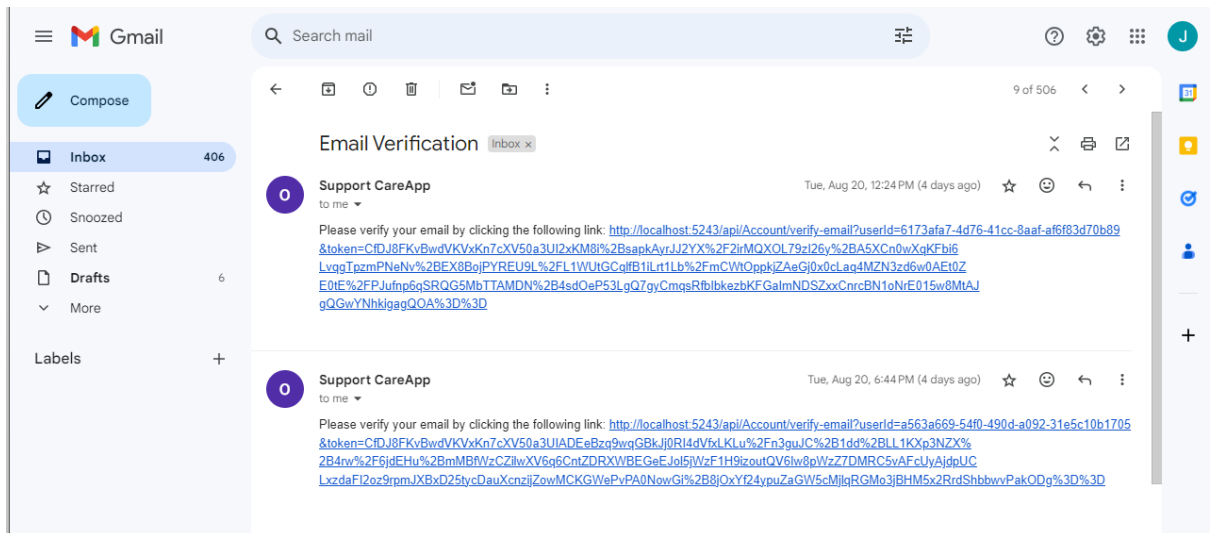
- 1. Functional Testing:** The functional tests of this ecommerce application done to check whether the application works as expected with no malfunctions.

Test Case ID	Description	Functionality	Expected Output	Status
F1_TestCase	Test User Registration	Registration Form	New User Account Created	Pass
F2_TestCase	Test Product Filtering	Category Filter Applied	Products filtered by category	Pass

Functional Testing Result: The main functionalities like browsing products, searching, adding to cart, and checking out were tested extensively. All major functionalities worked as expected with a few minor usability bugs identified in the checkout process. **Outcome:** 93% test cases passed; 7% identified as needing minor refinements in UI and flow.

- 2. Security Testing:** Security Testing: Security testing is a critical aspect of my e-commerce web application to ensure that user data is protected and that transactions are secure. The purpose of this testing phase was to find weak points and security attacks against the application which may take advantage compromising information of both the users and the application. In an effort to make my e-commerce web application more secure, I implemented JWT (JSON Web Token) for user authentication and session management. JWTs are a way of securing that the user and to prevent others from hijacking their session.

In Order to have a secured user account, I used an email notification system for security events. In this project, I used email notifications for the verification of the account of new users. When a user registers for a new account, the application sends a verification email to the user's registered email address. This makes sure that the email address of the users is valid and prevents other users who are not authorized from registering with someone else's email.



3. **API Postman Testing:** API testing is an important part of making sure that this backend services of the e-commerce application function properly. I used postman to test Api endpoints by sending http request with different payload and seeing the response and this enabled me to prove that all the interactions between the front-end and back-end of this application works.

Setting Postman for Testing API

Before testing my endpoints, I followed these steps:

- I. Downloaded and Installed the Postman Application
- II. Grouped my API request and made them a compilation for better management
- III. Set up environment variables

Testing Process

The Testing process in postman involved these steps:

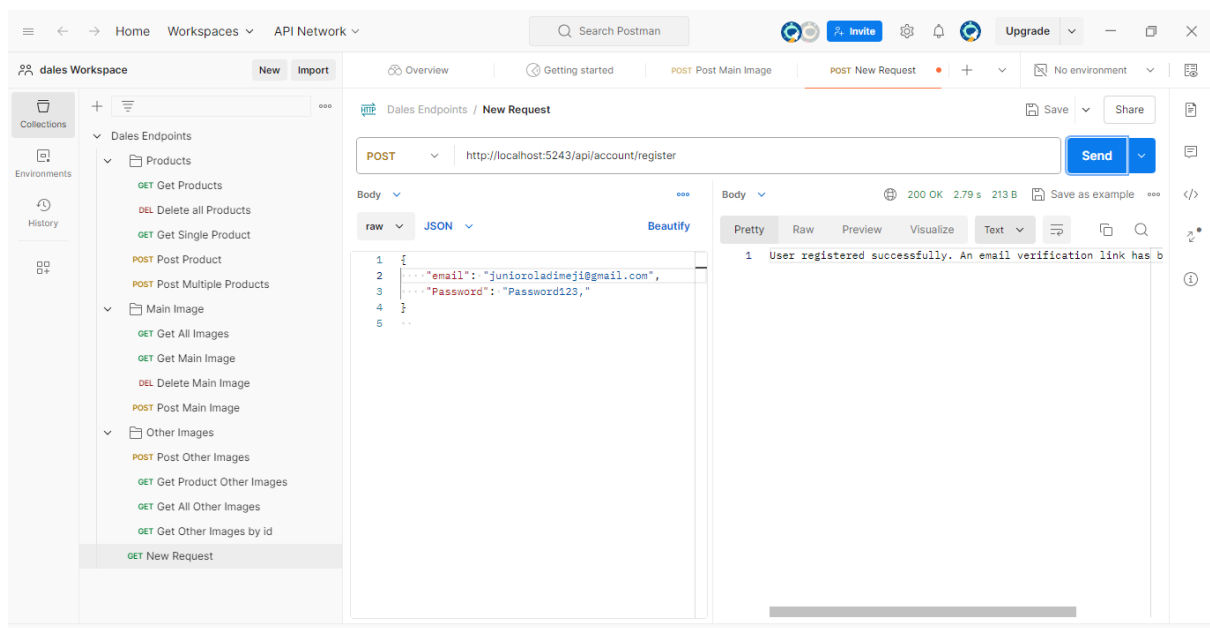
- i. **Create Requests:** I defined HTTP requests for each of the endpoints to be tested, with parameters like request type, headers and body in case required.
- ii. **Send Requests:** I executed the requests and observed the responses.
- iii. **Verify Responses:** I validated the response status code, headers and body as per expectations
- iv. **Automate Testing:** I performed testing in an automated way which helped me to hold multiple requests and a diverse simulation of real-life scenarios using automation features present in Postman.

Example of Test Cases using Postman

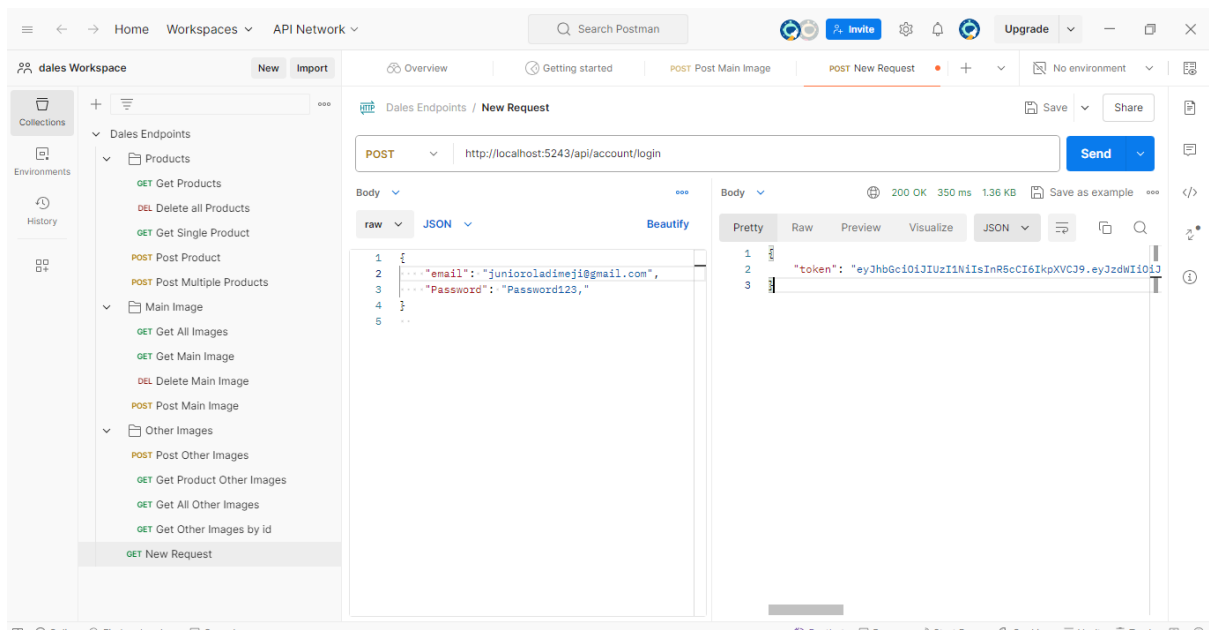
Table 6.2 : API Testcase

Test Case ID	Description	Request Type	Endpoint	Expected Output	Status
TC_1	Test user registration with data	POST	http://localhost:5243/api/account/register	201 'users created'	Pass
TC_2	Test user Login	POST	http://localhost:5243/api/account/login	200 'user Logged in'	Pass
TC_3	Test fetching products	GET	http://localhost:5243/api/products	200 OK	Pass

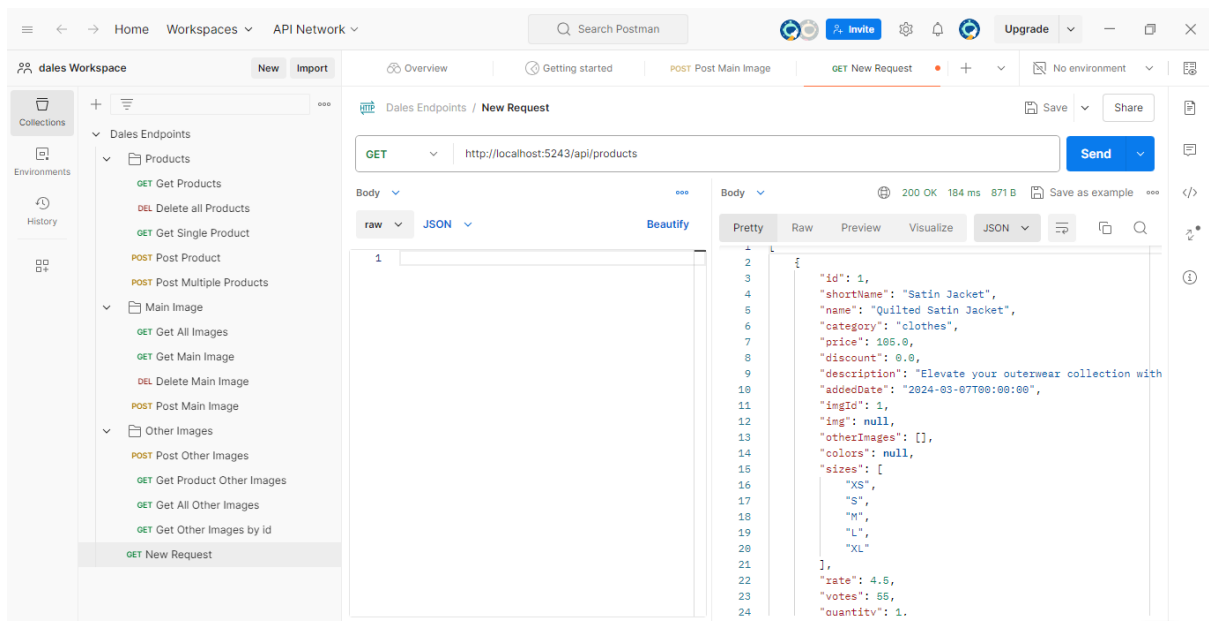
TC_1:



TC_2



TC_3



The tests I conducted for this ecommerce application was crucial during the development process. Through these tests, I was able to make this application more secured and user-friendly and also these tests helped in the early identification of errors and problem in the application.

Chapter 7 Conclusion

7.1 Introduction

This chapter will conclude the overall findings and results from the e-commerce platform development project. The next section of this chapter will recap the aims and objectives of this project identified in the first chapter and also assess how well these aims and objectives were achieved during implementation and finally discussing where the outcomes of this e-commerce application can be further enhanced. Lastly, I shall suggest future work to improve the platform's functionality and usability.

7.2 Review of Project Aims and Objectives

Some of the key objectives of the project included developing an e-commerce website for easy usability, focusing on inventory that spans across everything from clothes to electronics and many other products on a usable interface with modern web technologies while keeping it secure and safe. The project objectives were designed to support these aims through specific, actionable goals.

1. Aim 1: Develop a User-Friendly E-commerce platform
 - a. **Objective 1.1:** Design and implement a user-friendly interface utilizing React to ensure ease of navigation and accessibility
 - **Achievement:** This objective was successful as I managed to design an interface using React. The major components, such as clear navigation, responsiveness, and coherent and consistent visual elements, have been developed, which hereby increases and improves user interaction with the e-commerce application.
 - b. **Objective 1.2:** Integrate payment gateways to enhance a secure and easy transaction
 - **Achievement:** A payment gateway was not integrated into the e-commerce application.
2. Aim 2: Implement Comprehensive Inventory Management
 - a. **Objective 2.1:** Develop backend logic using C# and SQLite to manage product inventories, including adding, updating, and removing products.
 - **Achievement:** The backend logic was implemented and working fine, where product inventories could be created, read, updated & deleted by administrators. This is a key functionality for keeping stock levels and product availability correct.
 - b. **Objective 2.2:** Implement real-time inventory tracking to update stock levels automatically based on user purchases.
 - **Achievement:** A working real-time inventory tracker was also built, and stock levels updated accurately after a purchase. This means fewer hours of manual oversight, which will improve the accuracy and availability of stock.
3. Aim 3: Enhance User Experience with Modern Web Technologies
 - a. **Objective 3.1:** Utilize HTML, CSS, JavaScript, and SASS to create a responsive and visually appealing user interface.
 - **Achievement:** This was achieved by utilizing the power of modern web technologies to build an interactive as well as visually appealing front end that is responsive and works across different devices and screen sizes.

- b. **Objective 3.2:** Implement Redux for state management to improve application performance and user experience.
 - **Achievement:** By means of Redux the state was organized across the client-side of the application, which leads to increased performance and an optimized User experience such as in complex user interactions or handling data flows.
- 4. **Aim 4:** Ensure Secure and Scalable Backend Infrastructure
 - a. **Objective 4.1:** Use ASP.NET Core Web API for the backend to handle server-side operations, ensuring security and scalability.
 - **Achievement:** The backend was built with ASP.NET Core Web API which gave a robust framework to handle server-side operations securely and efficiently. Such setup allows to scale in future when the user demand grows.
 - b. **Objective 4.2:** Implement user authentication and authorization to protect sensitive user information and control access.
 - **Achievement:** User authentication and authorization mechanisms were added to the e-commerce application to protect user data and restrict access levels within this application, which ensures that the application is well secured.

7.3 Review of Requirements

7.3.1 Functional Requirements

Essential Functional Requirements:

- a. Met:
 - **Product Search:** Implemented a product search functionality using keywords and also category filtering.
 - **Product Management:** Comprehensive create, update, read, delete functionality was implemented for product management.
 - **User Account Management:** User authentication and authorization was implemented in the application using JWT token
 - **Shopping Carts and Checkout:** All the functionalities for adding, updating, and removing items in the shopping cart and an easy checkout process were developed.
- b. Not Met:
 - **Payment Gateway:** Payment gateway was not integrated successfully due to time constraint and difficulty to implement
 - **Admin:** Admin page was not implemented due to time constraint

Desirable Functional Requirements:

- a. Met:
 - **Order Management:** A basic order tracking system was implemented in the application.

- **Wishlist Functionality:** Users can add products to a Wishlist in case they would like to purchase the product in the future.
- b. Not Met:
- **Review and Ratings:** This feature was partially developed but not fully implemented due to time constraints.

Luxury Functional Requirements:

- a. Met:
- **Automated Emails:** Implemented automatic email to authenticate the user after they register
- b. Not Met:
- **Product Recommendation:** The product recommendation system was not implemented in the e-commerce application due to limited resources to work with and the complexity involved.

7.3.2 Non-Functional Requirements

Essential Non-Functional Requirements

- a. Met:
- **Security:** Basic security measures were used in the application.
 - **Performance:** The application performs well and ensures fast load of the pages.
 - **Scalability:** The backend architecture of the application supports scalability for future improvement.
- b. Not Met: All essential non-functional requirements were achieved.

Desirable Non-Functional Requirements:

- a. Met:
- **Multiple Browser Compatibility:** The application works well across different types of browsers
 - **Responsiveness:** The e-commerce application is responsive across various screens and devices as it was tested on a mobile phone, a laptop and a large screen monitor
- b. Not Met: All desirable non-functional requirements were met.

Luxury Non-Functional Requirements:

- a. Met
- **Cloud Deployment:** The ecommerce platform was successfully deployed
- b. Not Met
- **Advanced Security:** More advanced security measures were not implemented.

7.3.3 User Interface Requirements

Essential User Interface Requirements:

- a. Met:
 - **Clear Navigation:** The navigations on the application are easy to locate and easy to use.
 - **Responsive Design:** The user interface design is responsive across various devices and screen sizes.
 - **Consistent Design:** Uniform design elements like colors are used across all pages in the application.
 - **Error Handling:** User-friendly error messages are displayed in the application.
- b. Not Met
 - All essential UI requirements were met.

Desirable User Interface Requirements:

- a. Met
 - **Quick Access Menu:** Developed easy navigations and buttons to important section and pages on in the application.
- b. Not Met
 - **Enhanced Design:** Advanced visual elements like animations were not implemented in the application.

Luxury User Interface Requirements:

- a. Met
 - None
- b. Not Met:
 - **Advanced Search System:** Advance search system with images and voices was not implemented in the application
 - **Interactive Product Displays:** Advanced product visuals were not developed.

7.4 Further work and Improvements

Though the current version of this e-commerce web application builds some core functionality and users have a better shopping experience built on it, there still remain quite enhancements that could be added in future developments. There are number of things that can be done to improve this e-commerce web application even better like, more products personalization Features in which machine learning can be used to personalize users experience such as suggestions personalized product offers. Also, the integration of notification system in more areas of the software to provide security alerts for any suspicious activities as well order notifications which would add the required feature of user engagement & Security. Another area of development will be building a mobile application which would cater for the needs of the users that prefers the use of a mobile application to the use of a website. Also, for future enhancements, I would like to have more test to ensure a deeper and more robust coverage of the application. This will involve a wider range of testing, in areas such as performance and to test for vulnerabilities which might have been missed during original development phase. The milestone of finally finishing this e-commerce web application shows that I am capable of creating a working, secure and user-friendly website. This project has allowed me to use many of the web development practices I learned during my education, including proper architectural design processes and best security protocols. This implementation is not only important from a learning perspective of developing a production ready ecommerce application but also critical for future studies and enhancements. This base provides quick progress and iteration of the application to meet changing user needs and market trends keeping its relevance in today ever-changing e-commerce world.

7.5 Conclusion

This project has been an extensive process of design and developing an e-commerce web application that users find useful to work with. The Design was about building an application which should be a proper solution for successful shopping experience of users and the security of their data. But then there are improvements that could be made, especially with luxury features and features like the payment gateway, cloud deployment, and the admin functionality which could be implemented due to time and difficulty of the implementation of these features and a higher level of security. To get even more of an edge on the market, a future version could focus specifically on these areas.

Bibliography

1. Amazon Web Services, Inc., 2022. Fully Managed Relational Database - Amazon RDS. [online] Available at: <https://aws.amazon.com/rds/> [Accessed: July 2024].
2. Microsoft, 2022. ASP.NET | Open-source web framework for .NET. [online] Available at: <https://dotnet.microsoft.com/en-us/apps/aspnet> [Accessed: June 2024].
3. BizWorth, 2022. Valuation in Mergers & Acquisitions: Key Considerations. [online] Available at: <https://www.bizworth.com/blog/valuation-in-mergers-acquisitions-key-considerations> [Accessed: July 2024].
4. FXOpen, 2022. AMZN Share CFD Trading. AMZN Live Chart. [online] Available at: <https://fxopen.com/en-cy/amazon-share-cfd-trading/> [Accessed: July 2024].
5. 3PLGuys, 2022. Navigating Amazon FBA Success: 3PLGuys' California-Centric Logistics Solution. [online] Available at: <https://3plguys.com/blog-post/navigating-amazon-fba-success-3plguys-california-centric-logistics-solution> [Accessed: August 2024].
6. Ritchie, M., 2020. ASP.NET Core 5 and React: Hands-On Full Stack Web Development Using ASP.NET Core 5, React 17, and Entity Framework Core 5. Packt Publishing.
7. Market Business News, 2020. What is an API and how does it work? [online] Available at: <https://marketbusinessnews.com/what-is-an-api-and-how-does-it-work/253627/> [Accessed August 2024].
8. Microsoft, 2003. Logon and Authentication Technologies: Logon and Authentication | Microsoft Learn. [online] Available at: [https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780455\(v=ws.10\)](https://learn.microsoft.com/en-us/previous-versions/windows/it-pro/windows-server-2003/cc780455(v=ws.10)) [Accessed June 2024].
9. EdbMails, n.d. Fix the error 'Unable to mount database ec=1108' in Exchange. [online] Available at: <https://www.edbmails.com/pages/fix-unable-to-mount-database-hr0x80004005-ec1108-error.html> [Accessed July 2024].
10. Havard, K., 2021. Mastering full stack development with React and ASP.NET Core. New York: Apress.

Appendix A - Running and Using Instructions for Ecommerce Project

To run the Ecommerce project locally on your machine, ensure you have the following prerequisites installed:

- .NET 8 SDK: For the backend (ASP.NET Core).
 - Node.js (latest version): For the frontend (React).
1. **Setting Up and Running the Backend (ASP.NET Core):** Navigate to the main folder which is 'EcommerceApp' either manually or with the terminal, within this parent folder is the backend folder where all the server-side files are and the solution file. If not done automatically, restore the necessary NuGet packages with the command 'dotnet restore' and either run the server by clicking the play button in the IDE or run the server with the 'dotnet run' in the terminal. The backend server will start and typically listen on <https://localhost:5274> or a different localhost port.
 2. **Setting Up and Running the Frontend (React):** Navigate to the main folder which is 'EcommerceApp' either manually or with the terminal, within this parent folder is the frontend folder where all the client-side file and components are located. installation of the NodeJS and necessary dependencies are important and can be done with the command 'npm install'. This will install React and all required dependencies specified in the package. Json file. Start the frontend development server by running "npm run start". This will start the React application, typically on <http://localhost:3000>

The backend service has been successfully deployed to Microsoft Azure, and it is fully operational. However, the front-end application was not successfully deployed. Therefore, there is no need to run the backend server locally, as it is already deployed and accessible. The focus now is on running the React front-end locally, which will communicate with the already deployed backend server on Azure.

3. **Project Folder Link**
<https://github.com/Dimeji12/EcommerceApplication>
4. **Video Link**
<https://drive.google.com/file/d/1bGCKFmHeWcRQJSulM-qWDPvzBak4LSwY/view?usp=sharing>

