CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND
COMMUNICATION TECHNOLOGY

PROJECT - SPECIALIZED REPORT
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

BUILD SPORT EQUIREMENT ECOMMERCE WEBSITE WITH MEVN STACK

Student: Nguyen Thanh Phat
Student ID: B2005853
Cohort: K46

Cantho, 09/2023

CAN THO UNIVERSITY
COLLEGE OF INFORMATION AND
COMMUNICATION TECHNOLOGY

PROJECT - SPECIALIZED REPORT
INFORMATION TECHNOLOGY
(HIGH-QUALITY PROGRAM)

BUILD SPORT EQUIREMENT ECOMMERCE WEBSITE WITH MEVN STACK

Advisor:
Dr. Lam Van Khang

Student:
Nguyen Thanh Phat
Student ID: B2005853
Cohort: K46

Cantho, 09/2023

Contents

List of Figures

List of Tables

Abstract

This project is aimed at creating an online marketplace for sports equipment, offering users a seamless and intuitive shopping experience. The website will be developed using the MEVN stack (MongoDB, Express.js, Vue.js, Node.js), supplemented with Bootstrap for responsive design, and TypeScript for static typing, enhancing code quality and understandability.

The development process will adhere to the principles of the Minimum Viable Product (MVP) approach, resulting in a compact, single-tiered website. This approach ensures the system is scalable, maintainable, and provides a rich, user-friendly interface.

The project's scope is concentrated on fulfilling the essential features of an e-commerce website, including product listing, shopping cart functionality, order processing, user authentication, wishlist management, and product reviews.

The anticipated outcomes of this project include a deeper comprehension of full-stack web development using the MEVN stack, understanding the MVP approach, and gaining insights into the operational aspects of an e-commerce website. This project serves as a practical application of these technologies and concepts, demonstrating their effectiveness in a real-world scenario.

Tóm tắt nội dung

Dự án này nhằm mục đích tạo ra một thị trường trực tuyến cho thiết bị thể thao, mang đến cho người dùng trải nghiệm mua sắm liền mạch và trực quan. Trang web sẽ được phát triển bằng MEVN stack (MongoDB, Express.js, Vue.js, Node.js), được bổ sung Bootstrap cho thiết kế đáp ứng và TypeScript cho kiểu tĩnh, nâng cao chất lượng và khả năng hiểu của mã.

Quy trình phát triển sẽ tuân theo các nguyên tắc của phương pháp Sản phẩm tối thiểu khả thi (MVP), dẫn đến một trang web nhỏ gọn, một tầng. Phương pháp này đảm bảo hệ thống có khả năng mở rộng, dễ bảo trì và cung cấp giao diện người dùng phong phú, thân thiện.

Phạm vi của dự án tập trung vào việc đáp ứng các tính năng thiết yếu của một trang web thương mại điện tử, bao gồm liệt kê sản phẩm, chức năng giỏ hàng, xử lý đơn hàng, xác thực người dùng, quản lý danh sách mong muốn và đánh giá sản phẩm.

Các kết quả dự kiến của dự án bao gồm sự hiểu biết sâu sắc hơn về phát triển web full-stack bằng MEVN stack, hiểu được phương pháp MVP và thu được thông tin chi tiết về các khía cạnh hoạt động của trang web thương mại điện tử. Dự án này đóng vai trò là ứng dụng thực tế của các công nghệ và khái niệm này, thể hiện hiệu quả của chúng trong một tình huống thực tế.

Part I

Introduction

1 Background

The rise of information technology has significantly impacted business operations, with software applications becoming integral to sales operations. However, choosing the right software and platform can be challenging, especially for small and medium-sized enterprises (SMEs).

Traditionally, many businesses have used monolithic architecture, a unified and self-contained model. While this model is convenient in the early stages of a project, it can become complex and hard to manage as the application grows.

To address these challenges, businesses are now turning to the MEVN stack (MongoDB, Express.js, Vue.js, Node.js). This stack allows for a scalable and maintainable system, providing a rich, user-friendly interface.

For instance, a sports equipment shop website can be developed using the MEVN stack. By breaking down the application into smaller parts, it's easier to scale services based on demand and deploy updates more frequently without disrupting the entire system.

In the case of our sports equipment shop website, we will be using the MEVN stack to manage products, orders, and payments. Inventory management wil not be included in the initial version of the system. Instead, it's planned to be incorporated in a future update. This approach allows for the system to be flexible and scalable, accommodating new features as needed while ensuring system reliability.

2 Problem statement

SMEs face difficulties in selecting suitable software and platforms for their evolving digital operations. Traditional monolithic architecture, while initially convenient, becomes cumbersome and challenging to manage as applications grow, hindering scalability and user experience.

For online sports equipment shops, managing products, orders, and payments is vital. However, due to constraints, the initial system version will be include inventory management, but it is planned for a future update. This strategy ensures flexibility, scalability, and system reliability, addressing key challenges

The core problem lies in developing a user-friendly, scalable, and reliable online sports equipment shop using a technology stack that overcomes the limitations of traditional monolithic architecture and effectively manages the fundamental requirements of an e-commerce website.

3 Research Objectives

The primary aim of this project is to design and implement an e-commerce application for a sports equipment shop, along with a front-end application using Vue.js. The specific objectives are as follows:

- **Literature Review:** Conduct a comprehensive review of existing literature on the MEVN stack, its benefits, challenges, and best practices. This will provide a theoretical foundation for the project.
- **Design:** Design a MEVN-based e-commerce application model. The design should consider factors such as scalability, fault tolerance, and ease of adding new features.
- **Development:** Develop a minimum viable product (MVP) of an e-commerce application for a sports equipment shop website using the proposed model. The MVP will include key features such as product management, order processing, and payment processing. Inventory management will be considered for future updates.
- **Documentation:** DDocument the entire process, including the design decisions made, challenges encountered, solutions implemented, and lessons learned. This will serve as a valuable resource for future projects of similar nature.

4 Research Scope

The focus of this research is on the design and implementation of a MEVN-based e-commerce application for a sport equipment shop website. The specific areas covered in this research include:

- **MEVN Stack:** The research will delve into the principles and practices of the MEVN stack. It will explore how to design and implement an e-commerce application using this stack.
- **Front-end Compatibility:** The research will explore the compatibility of the developed back-end API with various front-end frameworks. While the API is designed to be standalone and can work with any front-end framework, the effectiveness of this design will be evaluated
- **Front-end Development with Vue.js:**The research will delve into the integration of the developed back-end API with Vue.js. While the API is designed to be standalone and compatible with any front-end framework, Vue.js has been chosen for its approachability and popularity.
- **Key Features:** The research will focus on implementing key features for an e-commerce application, such as product management, inventory management, order processing, and payment processing.
- **Evaluation:** The research will include an evaluation of the implemented system in terms of functionality, performance, scalability, and reliability.

Please note that while the research aims to cover these areas, it is limited by the project's short time. Therefore, the e-shop application developed as part of this project will be a minimum viable product (MVP) with basic features.

5 Solution approach

The approach to solving the problem involves several steps, each designed to ensure the successful implementation of an e-commerce application for a sports equipment shop, along with a front-end application using Vue.js. The steps are as follows:

- **Literature Review:** The first step involves conducting a comprehensive review of existing literature on the MEVN stack. This will provide a theoretical foundation for the project and inform the design and implementation stages.
- **Design:** The next step is to design the MEVN-based e-commerce application model. The design will take into account factors such as scalability, fault tolerance, and ease of adding new features.
- **Development:** Once the design is complete, the development of the minimum viable product (MVP) begins. This involves coding the back-end services and front-end application, and setting up the necessary databases and interfaces. Inventory management will be considered for future updates.
- **Testing: :** After the MVP is developed, it will be thoroughly tested to ensure it functions as expected. This includes unit testing, integration testing, and system testing.
- **Evaluation:** The MVP will then be evaluated in terms of its functionality, performance, scalability, and reliability. Feedback from this evaluation will be used to identify areas for improvement.

6 Report structure

Chapter 1. Introduction: This chapter provides a general overview of the topic, its potential, and practical applications in the future. It introduces the concept of a MEVN-based e-commerce application for a sport equipment shop website.

Chapter 2. Literature review This section provides a summary of relevant research on the topic and the foundational knowledge necessary to develop an e-commerce application using the MEVN stack.

Chapter 3. System design and implementation This section presents the approach to the problem. It discusses the details of the implementation approach, including the tools used.

Chapter 4. Testing and evaluation. This chapter presents the testing plan and management, testing scenarios for the main functions of the system.

Chapter 5. Conclusion This section presents the results achieved, the remaining limitations, and the system's further development.

Part II

Content

Chapter 1

Requirements and specification

1 Overall Description

This section provides an overview of the requirements and specifications for the e-commerce application for a sports equipment shop. The application will manage products, orders, and payments. Inventory management will not be included in the initial version of the system but is planned for a future update.

1.1 Product Perspective

The e-commerce application is intended to provide a user-friendly, scalable, and reliable online platform for a sports equipment shop. It aims to overcome the limitations of traditional monolithic architecture and effectively manage the fundamental requirements of an e-commerce website.

1.2 Product Functions

The e-commerce application will include the following functions:

- **User Authentication:** Allows users to register and log in to their accounts.
- **Product Management:** Enables administrators to manage the type, brand and products available on the website.
- **Order Management:** Enables administrators to manage orders and update the order status on the website.
- **Order Processing:** Handles the processing of customer orders.
- **Payment Processing:** Manages the payments for customer orders.
- **Product Browsing:** Provide user to view or search list of products available on the store.
  - Search by filter and sort: Allows users to search for products available on the platform.

  - Product views: Provides detailed information about a product or products by type or brand.

- **Wishlist:** Allows users to save their favorite products for later viewing.
- **Basket:** This function allows users to add products to their shopping basket. The Basket page displays a list of products in the user's shopping basket, allowing them to view, update quantities, and remove items. Users can proceed to checkout to complete the purchase.
- **Order:** This function allows users to track the order status of their order and other informations of order.

1.3 User Classes and Characteristics

The application will have two main types of users: customers who are looking to purchase sports equipment, and administrators who manage the products, orders, and payments.

1.4 Operating Environment

The application will be developed using the MEVN stack (MongoDB, Express.js, Vue.js, Node.js) and will be designed to operate on various platforms and devices.

2 Requirements Specification

2.1 External Interface Requirements

The application will have a user-friendly interface for customers to browse products, place orders, and make payments. It will also have an administrative interface for managing products, orders, and payments.

2.2 Functional Requirements

User Use Cases

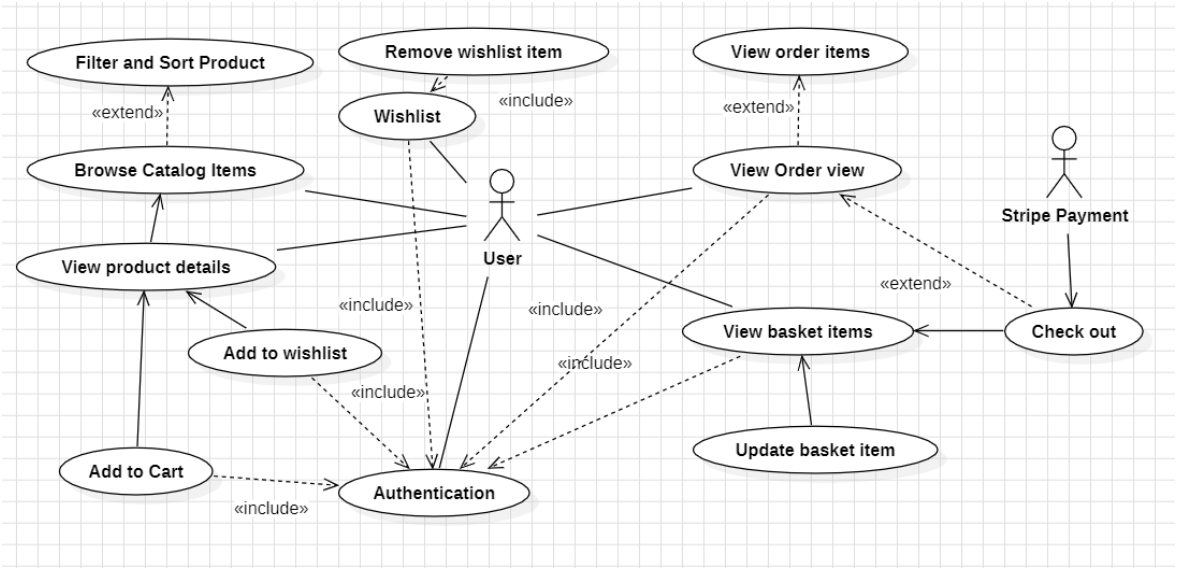


Figure 1.1: User usecase

1. **User Authentication:** The user registers and logs in to their account. This is the first step for users to interact with the system.
2. **Product Browsing:** The user views or searches the list of products available on the store. This includes searching for products by filter and sort, and viewing detailed information about a product or products by type or brand.
3. **Wishlist:** The user saves their favorite products for later viewing. This allows users to keep track of items they are interested in but are not ready to purchase yet.
4. **Basket:** The user adds products to their shopping basket, views, updates quantities, and removes items. The user proceeds to checkout to complete the purchase. This is the main function for users to purchase products.
5. **Order:** The user tracks the order status of their order and other information of the order. This allows users to keep track of their purchases and expected delivery times.

Admin Use Cases

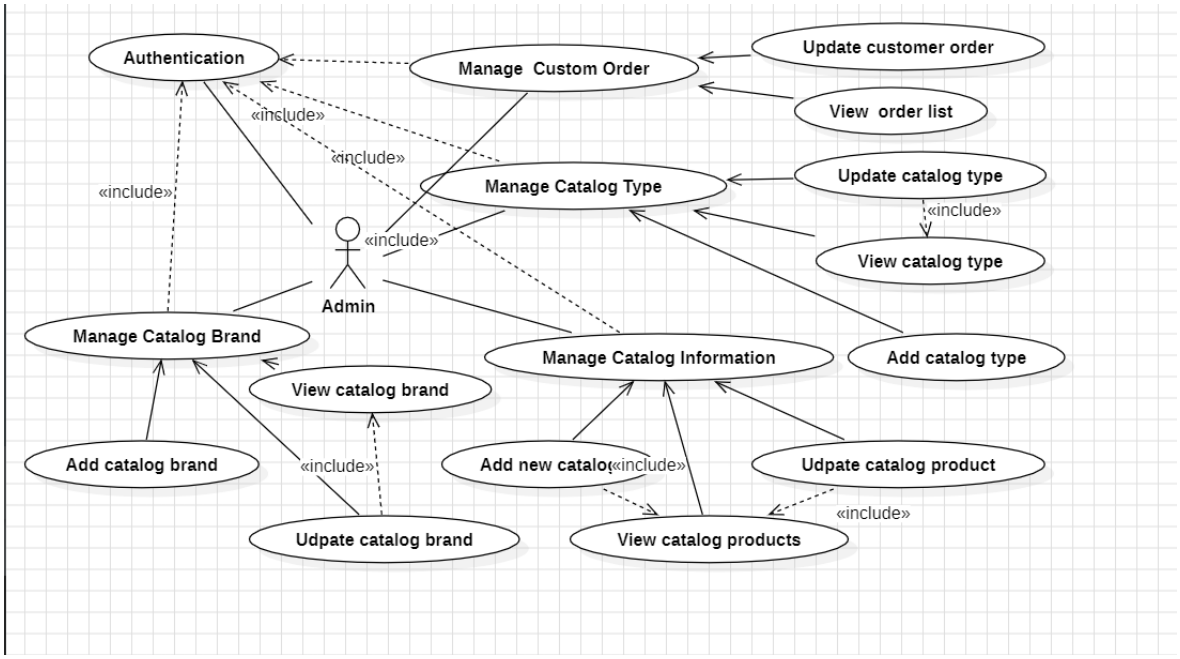


Figure 1.2: Admin usecase

1. **Product Management:** The administrator manages the type, brand, and products available on the website. This is crucial for keeping the product catalog up to date and relevant to the customers.
2. **Order Management:** The administrator manages orders and updates the order status on the website. This is important for ensuring that orders are processed and delivered in a timely manner.

2.3 Nonfunctional Requirements

The application will be designed to be scalable, allowing it to handle a growing number of products and users. It will also be reliable, ensuring that it operates correctly and provides accurate information.

Each function and requirement will be described in detail, including its purpose, how it works, and how it contributes to the overall functionality of the e-commerce application.

Chapter 2

Literature reiview

This literature review provides a comprehensive understanding of the MEVN stack, Docker, their benefits, and best practices. It forms the basis for the design and development of the e-commerce application for a sports equipment shop website.

1 Web technology stacks

A technology stack, often referred to as a solutions stack or a data ecosystem, is a list or combination of programming languages, tools, and technologies that are used to build and run a web application. Each layer of the technology stack builds on the foundation that's been laid by the layer below it.

Choosing the right technology stack is crucial for project success. The technology stack not only affects the speed and timeline of the project but also influences the ability to scale and the overall cost. Some common technology stacks include LAMP (Linux, Apache, MySQL, PHP), MEAN (MongoDB, Express.js, AngularJS, Node.js), and MEVN (MongoDB, Express.js, Vue.js, Node.js).

2 MEVN

2.1 Overview of MEVN stacks

The **MEVN** stack is a popular JavaScript software stack used to build powerful web applications. It consists of four different technologies: MongoDB, Express.js, Vue.js, and Node.js.

- **MongoDB:** A highly scalable and flexible document database with efficient querying and indexing. MongoDB stores data in flexible, JSON-like documents, meaning fields can vary from document to document and data structure can be changed over time.
- **Express.js:** A minimal and flexible Node.js web application framework that provides robust features for web and mobile applications. It provides a myriad of

HTTP utility methods and middleware at your disposal, which makes creating a robust API quick and easy.

- **Vue.js:** A progressive JavaScript framework for building user interfaces. It is approachable, performant, and versatile in building single-page web applications. VueJS focuses on the view layer. It has a very easy learning curve with a simple API which makes it one of the most loved frameworks.
- **Node.js:** An open-source runtime environment and library used to run web applications outside the client's browser. It is mainly used for server-side programming. It is asynchronous, event-driven, and highly scalable to write servers and databases.

2.2 Benefits of the MEVN Stack

The **MEVN** stack provides multiple advantages:

- **Platform Independence:** The MEVN stack is platform-independent, which means it can be used across different operating systems.
- **Single Language Development:** JavaScript is used on all levels of development from client-side to server-side which simplifies the process and makes development faster and efficient.
- **MVC Architecture:** The MVC (Model-View-Controller) architecture in the back-end organizes the server-side, making back-end development faster and more efficient.
- **Easy Learning Curve:** Vue.js is known for its easy learning curve and simple API, which makes it a popular choice among developers.

In the context of this project, the MEVN stack's benefits lie in its ability to create a scalable, robust, and efficient e-commerce application for a sports equipment shop website.

2.3 Challenges of the MEVN Stack

Despite its benefits, the **MEVN** stack does have some challenges:

- **New Framework:** Vue.js is a relatively new framework and doesn't have the support of a large community compared to other frameworks like React or Angular.
- **Lack of Plugins:** Due to its novelty, Vue.js lacks the extensive range of plugins available to older, more established frameworks.
- **Imperfect Core Functionality:** Developers may face difficulties due to its imperfect core functionality since it is not a full-fledged framework. Therefore, developers have to depend on third-party services.

In the context of this project, the MEVN stack's benefits lie in its ability to create a scalable, robust, and efficient e-commerce application for a sports equipment shop website.

2.4 Docker and Containerization

Docker is an open-source platform that automates the deployment, scaling, and management of applications using containerization.

- A **Docker** container is a standalone, executable package that includes everything needed to run an application, including the code, runtime, libraries, environment variables, and config files.
- **Docker** can be used to run MongoDB in an isolated environment, without the need for a dedicated server. This simplifies the setup process, ensures that the database runs in a consistent environment, and makes it easier to migrate to different servers.

Chapter 3

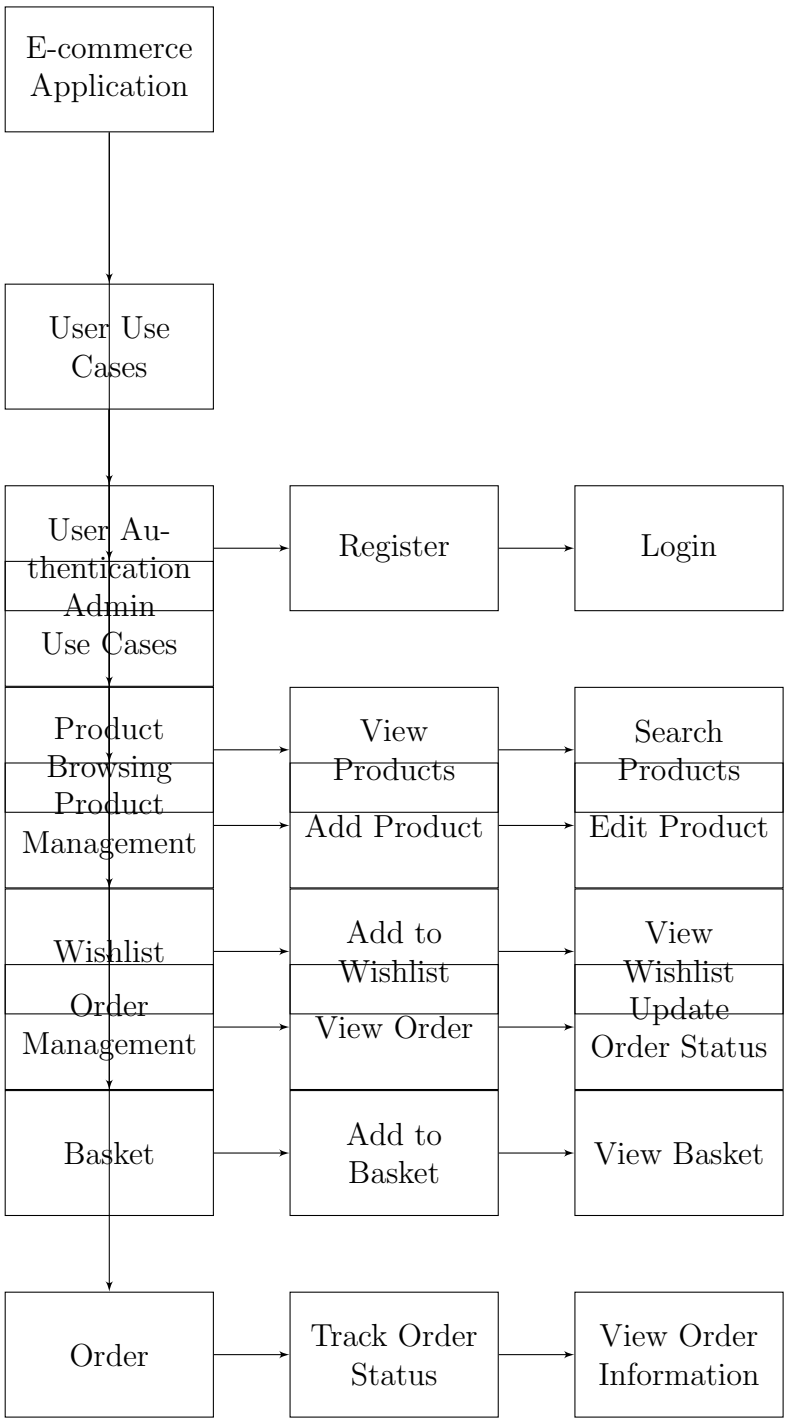
System design and implementation

1 Software Design

Application Architecture

Figure 3.1: Application Architecture

2 Decomposition Description



This diagram breaks down the e-commerce application into its main components: User Authentication, Product Browsing, Wishlist, Basket, Order, Product Management, and Order Management) and further decomposes these components into their respective functions.

3 Data Design

3.1 UserAccount

```
UserAccount
|
|-- username: String
|-- password: String
|-- email: String
|-- role: String
```

Data Description:

- `username`: A unique string that represents the username of the user. It is required and its length must be between 2 and 50 characters.
- `password`: A string that represents the user's password. It is required and will be hashed before being stored in the database.
- `email`: A unique string that represents the user's email address. It is required and must match the specified regular expression pattern.
- `role`: A string that represents the role of the user. It can be either "user" or "admin", with "user" being the default value.

3.2 Session

```
Session
|
|-- email: String
|-- valid: Boolean
|-- username: String
|-- role: String
|-- expireAt: Date
```

Data Description:

- `email`: A string that represents the user's email address.
- `valid`: A boolean that indicates whether the session is valid. The default value is true.
- `username`: A string that represents the username of the user.
- `role`: A string that represents the role of the user. It can be either "user" or "admin", with "user" being the default value.
- `expireAt`: A date that represents the expiration time of the session.

3.3 Brand

```
Brand
```

```
|
|-- name: String
|-- country: String
|-- image: Buffer
```

Data Description:

- `name`: A unique string that represents the name of the brand. It is required and its length must be between 2 and 50 characters.
- `country`: A string that represents the country of the brand. It is required.
- `image`: A buffer that represents the image of the brand.

3.4 CatalogItem

```
CatalogItem
|
|-- name: String
|-- description: String
|-- price: Number
|-- image: Buffer
|-- catalogType: ObjectId
|-- catalogBrand: ObjectId
|-- availableStock: Number
```

Data Description:

- `name`: A unique string that represents the name of the catalog item. It is required and its length must be between 2 and 50 characters.
- `description`: A string that represents the description of the catalog item.
- `price`: A number that represents the price of the catalog item. It is required.
- `image`: A buffer that represents the image of the catalog item.
- `catalogType`: An ObjectId that represents the type of the catalog item. It is required and references the Type model.
- `catalogBrand`: An ObjectId that represents the brand of the catalog item. It is required and references the Brand model.
- `availableStock`: A number that represents the available stock of the catalog item. It is required.

```
Type
|
|-- name: String
|-- description: String
|-- image: Buffer
```

Data Description:

- `name`: A unique string that represents the name of the type. It is required and its length must be between 2 and 50 characters.
- `description`: A string that represents the description of the type.
- `image`: A buffer that represents the image of the type.

3.5 BasketItem

```
BasketItem
|
|-- product: ObjectId (ref to CatalogItem)
|-- productId: String
|-- productName: String
|-- unitPrice: Number
|-- oldUnitPrice: Number
|-- quantity: Number
|-- image: Buffer
```

Data Description:

- `product`: An ObjectId that represents the product in the basket. It references the CatalogItem model.
- `productId`: A string that represents the id of the product.
- `productName`: A string that represents the name of the product.
- `unitPrice`: A number that represents the unit price of the product.
- `oldUnitPrice`: A number that represents the old unit price of the product.
- `quantity`: A number that represents the quantity of the product in the basket.
- `image`: A buffer that represents the image of the product.

3.6 WishlistItem

```
WishlistItem
|
|-- product: ObjectId (ref to CatalogItem)
|-- productId: String
|-- productName: String
|-- oldUnitPrice: Number
|-- image: Buffer
```

Data Description:

- `product`: An ObjectId that represents the product in the wishlist. It references the CatalogItem model.
- `productId`: A string that represents the id of the product.
- `productName`: A string that represents the name of the product.
- `oldUnitPrice`: A number that represents the old unit price of the product.
- `image`: A buffer that represents the image of the product.

3.7 CustomerWishlist

```
CustomerWishlist
|
|-- userId: ObjectId (ref to UserAccount)
|-- items: Array of WishlistItem
```

Data Description:

- `userId`: An ObjectId that represents the user who owns the wishlist. It references the UserAccount model and is required.
- `items`: An array of WishlistItem that represents the items in the wishlist.

3.8 BasketItem

```
BasketItem
|
|-- product: ObjectId (ref to CatalogItem)
|-- productId: String
|-- productName: String
|-- unitPrice: Number
|-- oldUnitPrice: Number
|-- quantity: Number
|-- image: Buffer
```

Data Description:

- `product`: An ObjectId that represents the product in the basket. It references the CatalogItem model.
- `productId`: A string that represents the id of the product. It is required.
- `productName`: A string that represents the name of the product. It is required.
- `unitPrice`: A number that represents the unit price of the product. It is required.
- `oldUnitPrice`: A number that represents the old unit price of the product. It is required.

- `quantity`: A number that represents the quantity of the product in the basket. It is required.
- `image`: A buffer that represents the image of the product.

3.9 CustomerBasket

```
CustomerBasket
|
|-- userId: ObjectId (ref to UserAccount)
|-- items: Array of BasketItem
```

Data Description:

- `userId`: An ObjectId that represents the user who owns the basket. It references the UserAccount model and is required.
- `items`: An array of BasketItem that represents the items in the basket.

3.10 OrderItem

```
OrderItem
|
|-- productId: String
|-- price: Number
|-- productName: String
|-- image: Buffer
|-- quantity: Number
|-- discount: Number
```

Data Description:

- `productId`: A string that represents the id of the product. It is required.
- `price`: A number that represents the price of the product. It is required.
- `productName`: A string that represents the name of the product. It is required.
- `image`: A buffer that represents the image of the product.
- `quantity`: A number that represents the quantity of the product in the order. It is required.
- `discount`: A number that represents the discount on the product. It is required.

3.11 Order

```
Order
|
|-- userId: String
|-- sequenceNumber: Number
|-- orderDate: Date
```

```
|-- paymentStatus: String
|-- orderStatus: String
|-- shippingCity: String
|-- shippingStreet: String
|-- shippingState: String
|-- shippingCountry: String
|-- shippingZipCode: String
|-- total: Number
|-- orderNumber: String
|-- phone: String
|-- orderItems: Array of OrderItem
```

Data Description:

- `userId`: A string that represents the id of the user who placed the order. It is required.
- `sequenceNumber`: A number that represents the sequence number of the order.
- `orderDate`: A date that represents the date of the order. It is required.
- `paymentStatus`: A string that represents the payment status of the order. It is required.
- `orderStatus`: A string that represents the status of the order. It is required.
- `shippingCity`, `shippingStreet`, `shippingState`, `shippingCountry`, `shippingZipCode`: Strings that represent the shipping address of the order. They are required.
- `total`: A number that represents the total amount of the order. It is required.
- `orderNumber`: A string that represents the order number. It is required.
- `phone`: A string that represents the phone number of the user.
- `orderItems`: An array of OrderItem that represents the items in the order.

4 Implementation

4.1 System functionality

4.2 Main views

Home view

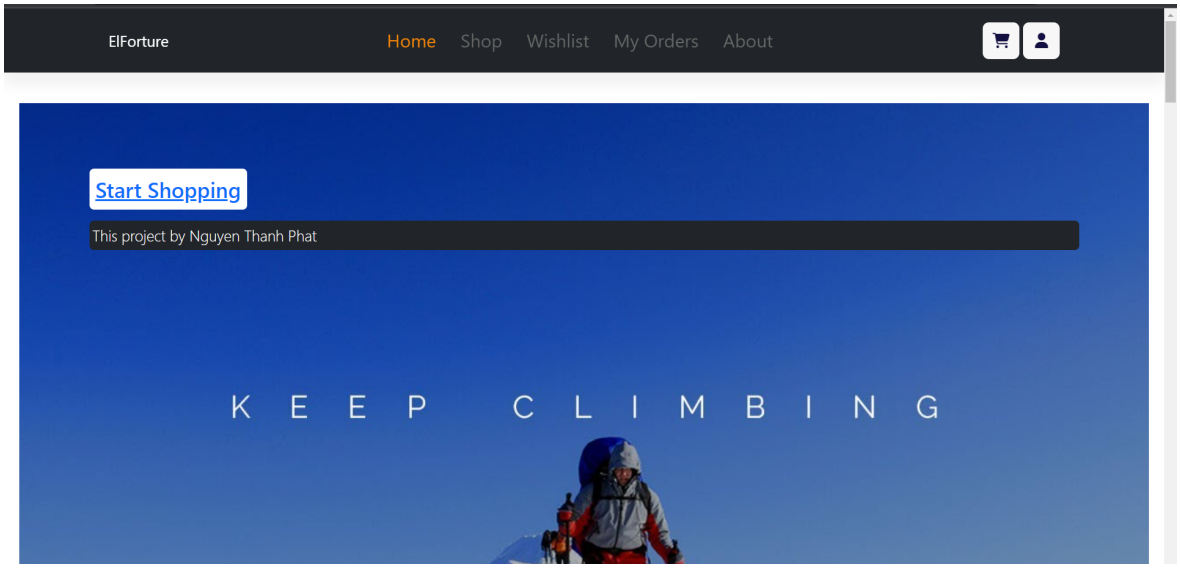


Figure 3.2: Home View

The HomeView allows users to easily navigate between different sections of the website. It is located at the top of the website and includes router links to the Home, Shop, Wishlist, My Orders, and Cart pages.

Additionally, it provides a user account icon for logged-in users, displaying a dropdown menu with options to access login and register, the Admin page or sign out. The menu bar also includes a hamburger icon that toggles the visibility of the main navigation links when the screen size is small.

4.3 Authentication

User Authentication function is a crucial part of the E-Commerce application. It allows users to create a new account or log into an existing one. This function is essential for personalizing the user experience and securing user data.

Register view

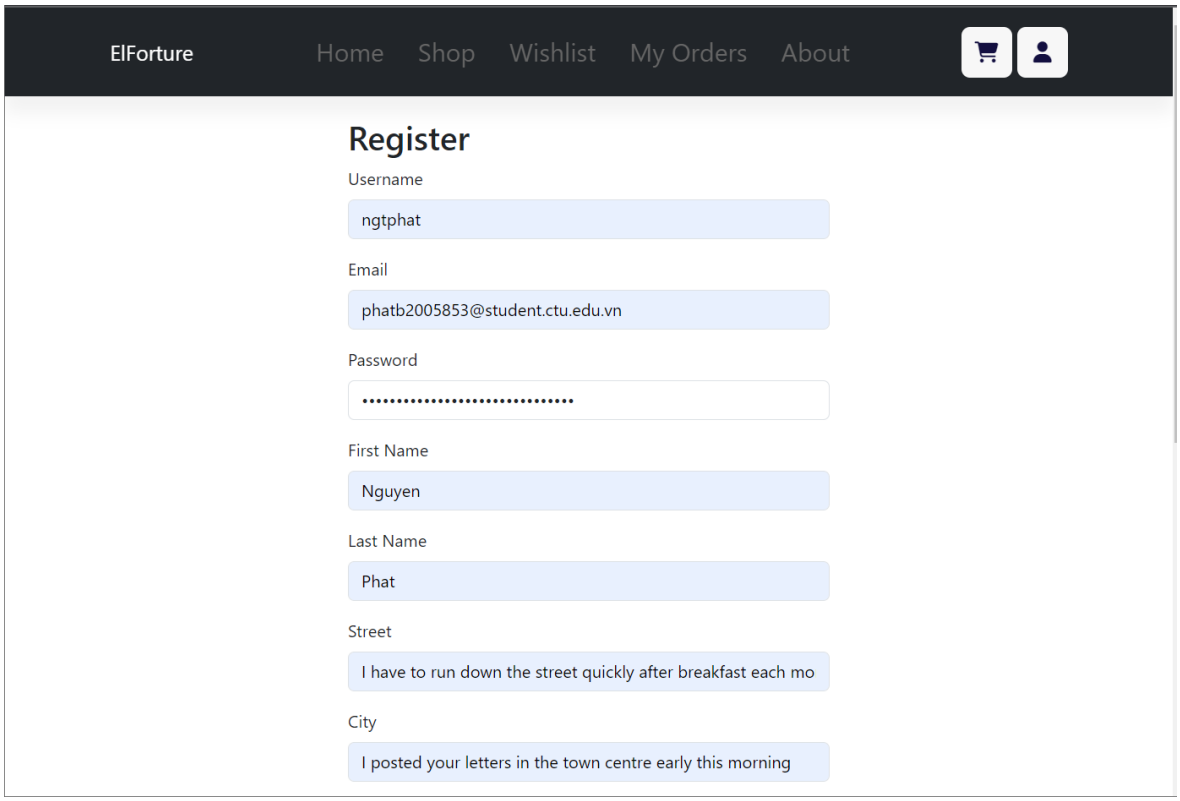


Figure 3.3: Register view

The Register feature enables new users to create an account by providing necessary details such as their name, email, and password. Once the account is created, users can access all features of the E-Commerce application.

Login view

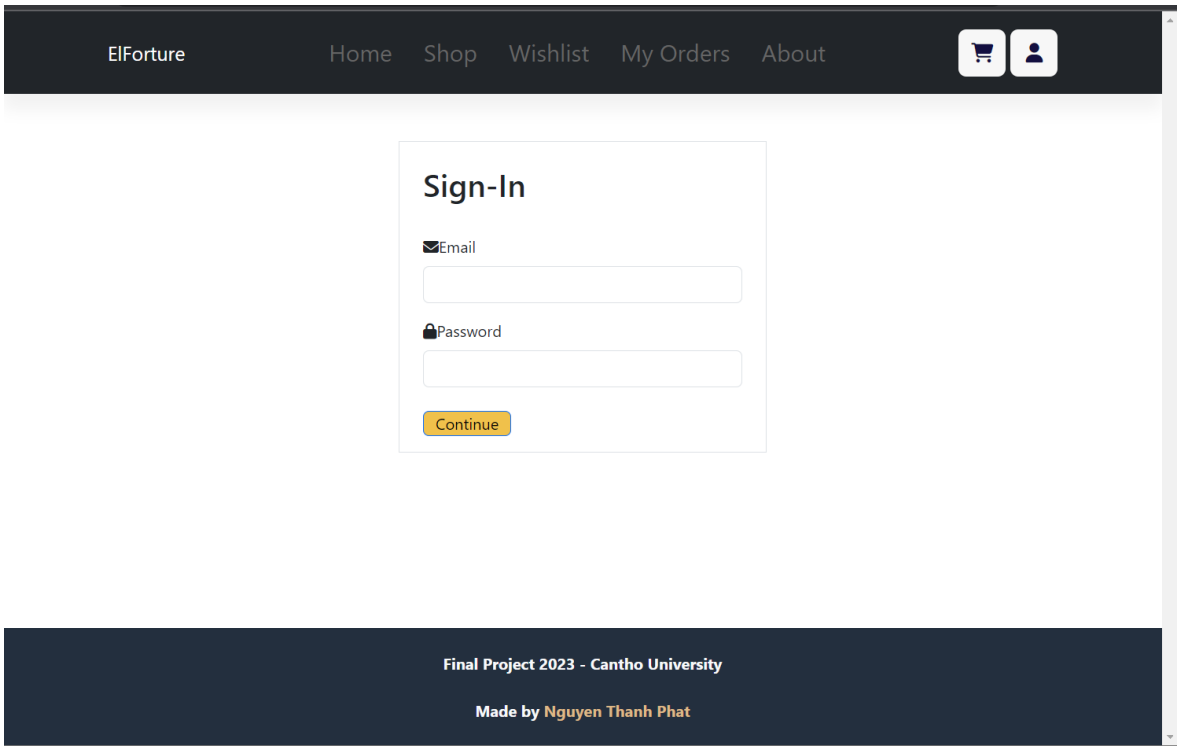


Figure 3.4: Login view

The Login feature allows returning users to access their accounts by entering their registered email and password. Upon successful login, users are redirected to the home page and can navigate through the application as authenticated users.

4.4 Product browse

Search by filter (Shopping browse)

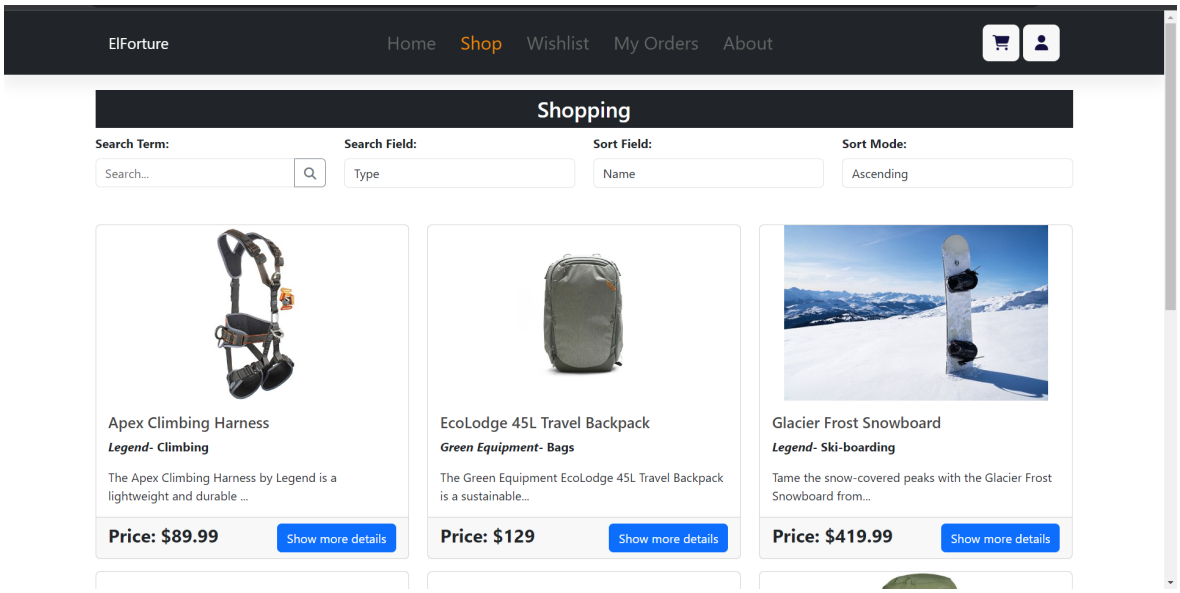


Figure 3.5: Shopping View

The Product Search function is an essential part of the E-Commerce application. It allows users to search for products available on the platform. This function includes several components: View All Products, Search Products By Query, Filter Products By Type, and Sort Products.

Product Details Page

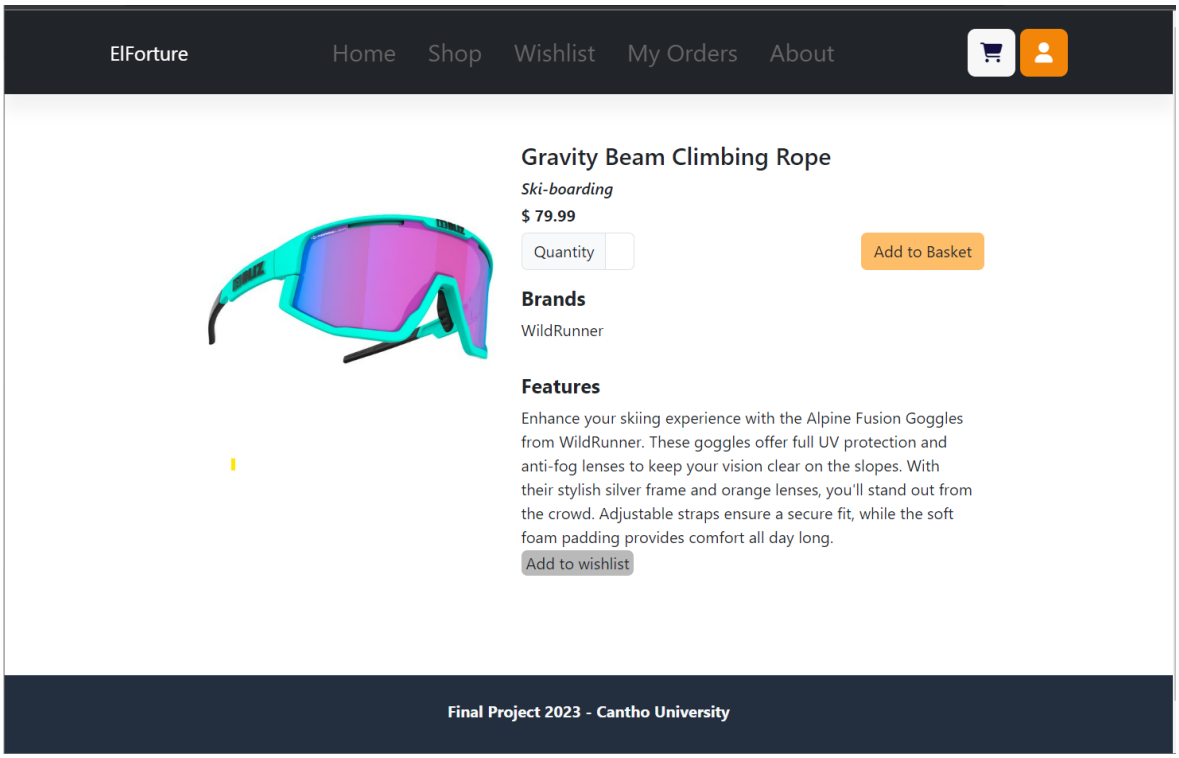


Figure 3.6: Product Detail View

The Product Search function is an essential part of the E-Commerce application. It allows users to search for products available on the platform. This function includes several components: View All Products, Search Products By Query, Filter Products By Type, and Sort Products. Add to Cart and Add to Wishlist notification.

4.5 Wishlist

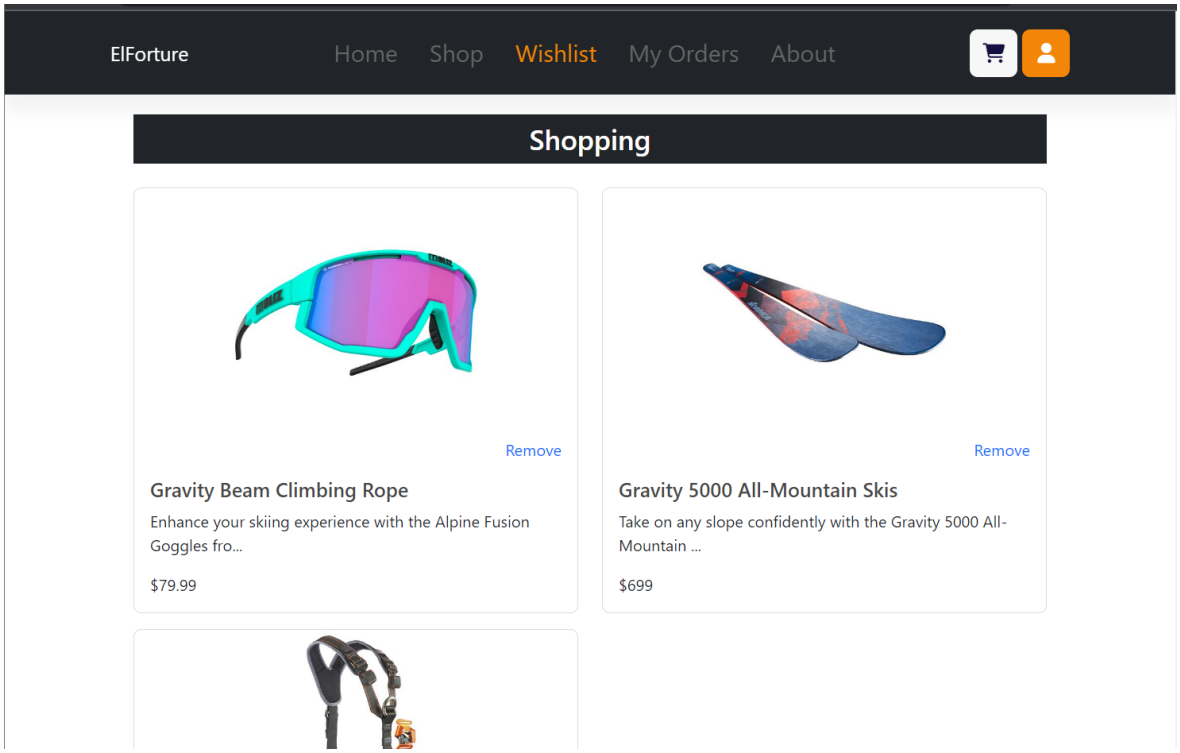


Figure 3.7: Wishlist vieww

The Wishlist page displays a list of products that the user has added to their wishlist. It allows users to view and manage their wishlist items, including adding them to the cart or removing them. Each wishlist item includes a product image, name , overall description, price and option to access more details about the product.

4.6 BasketView

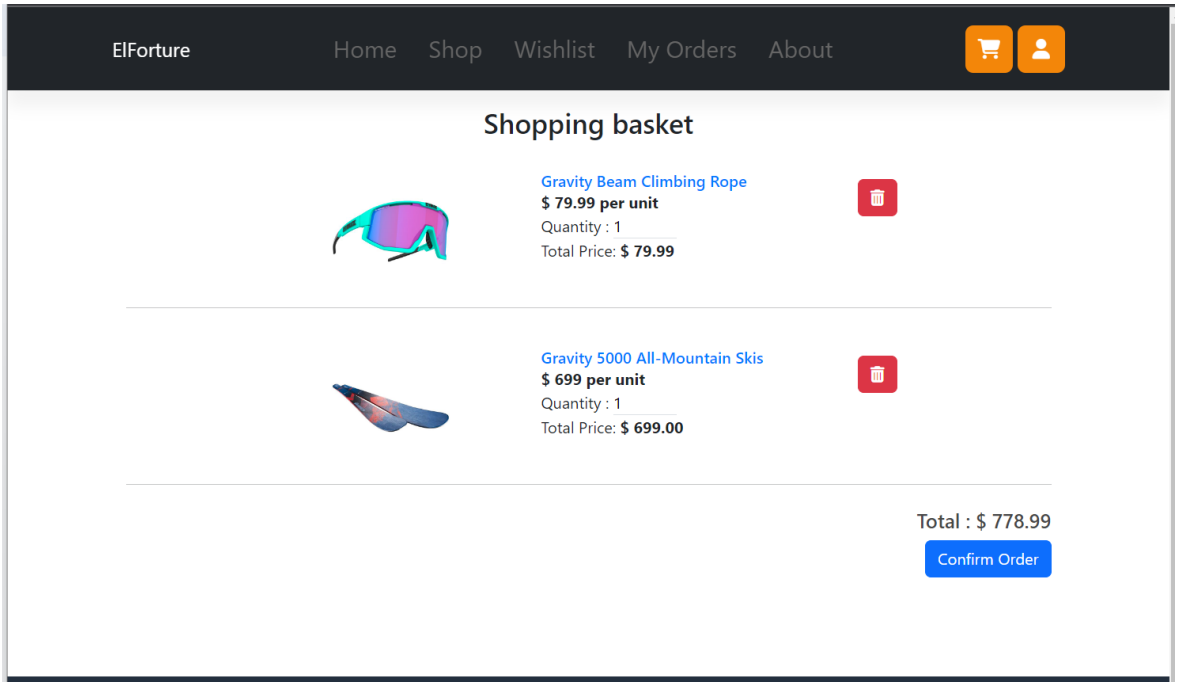


Figure 3.8: Basket View

The BasketView displays a list of products in the user's shopping cart, allowing them to view, update quantities, and remove items. Users can proceed to checkout to complete the purchase.

Process Payment: This page is calling and waiting for a redirect website from Stripe Payment Gateway which processes in Server-side API.

ToCheckOutView

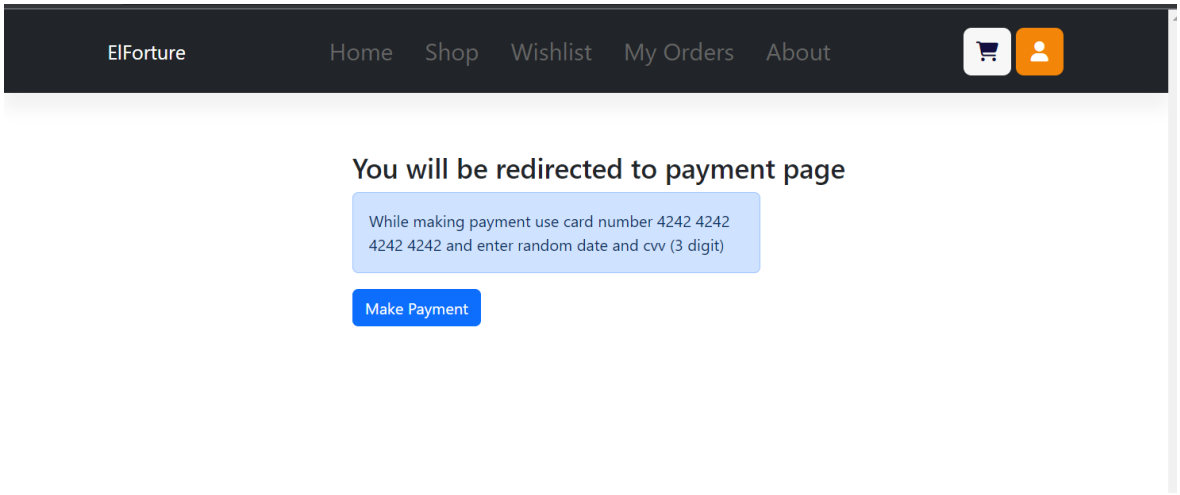


Figure 3.9: Proced To Payment View

Then user will direct to Payment page of Stripe:

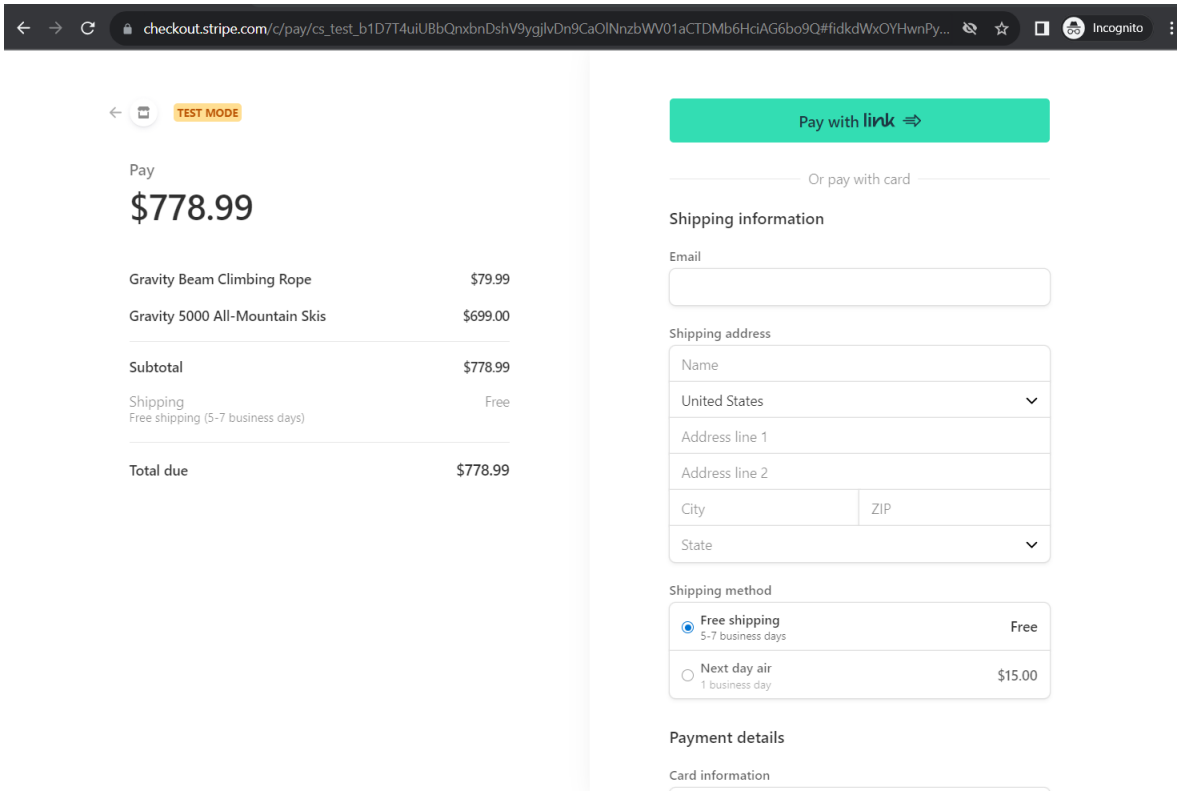


Figure 3.10: Payment page of Stripe

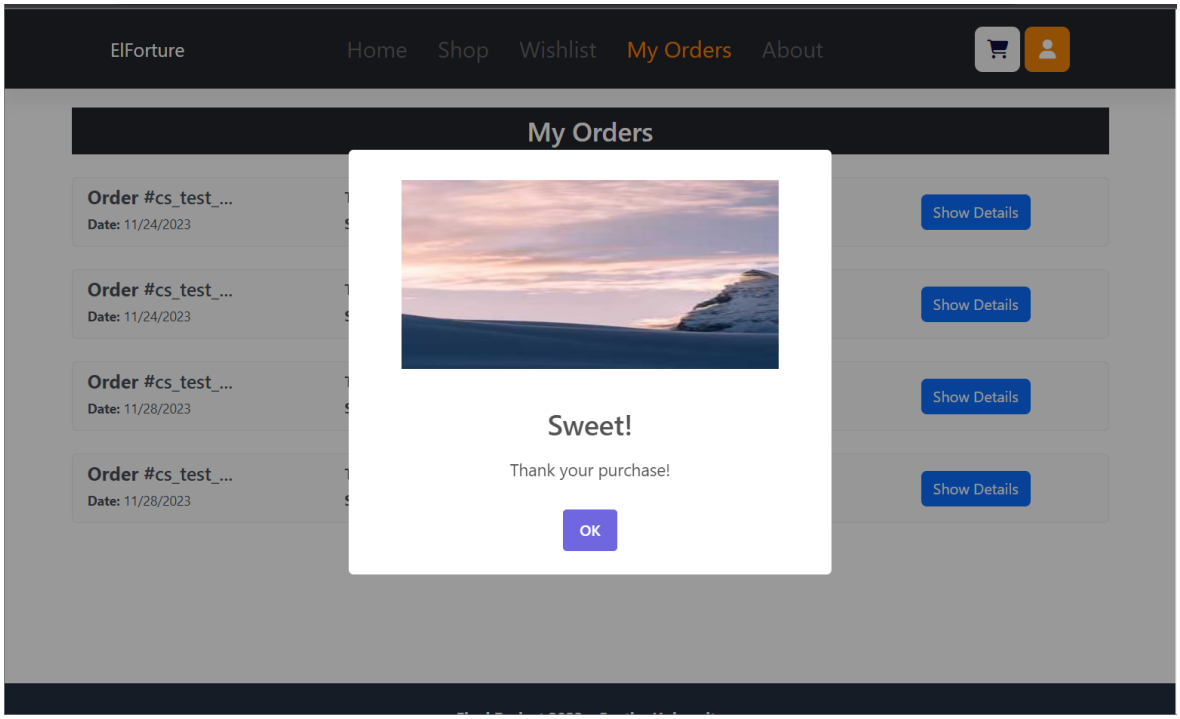


Figure 3.11: Order successfully

4.7 Admin - Catalog Type

The Administrator - Type Management function is a crucial part of the E-Commerce application. It allows the administrator to manage the product types available on the platform. This function includes several components: View All Categories, View Type By ID, Create Type, and Update Type.

View All Categories

The View All Categories component retrieves all the types from the database.

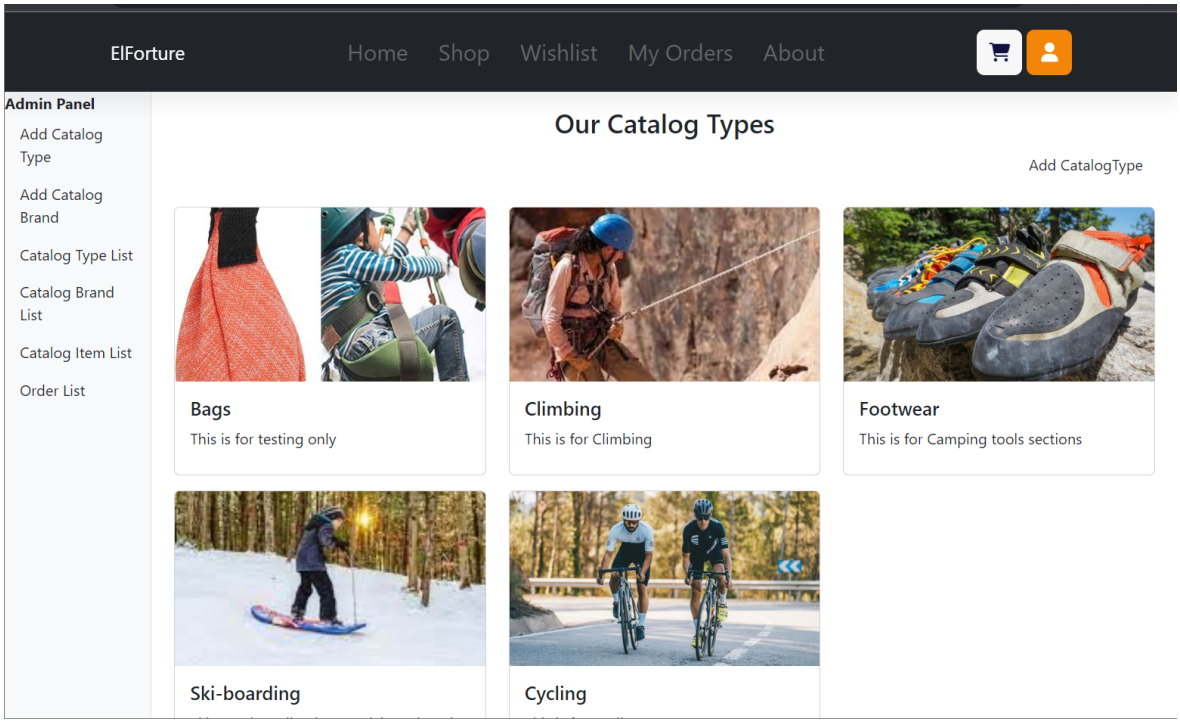


Figure 3.12: View Types

Add new type

TThe Add Category component allows the administrator to add a new type to the database.

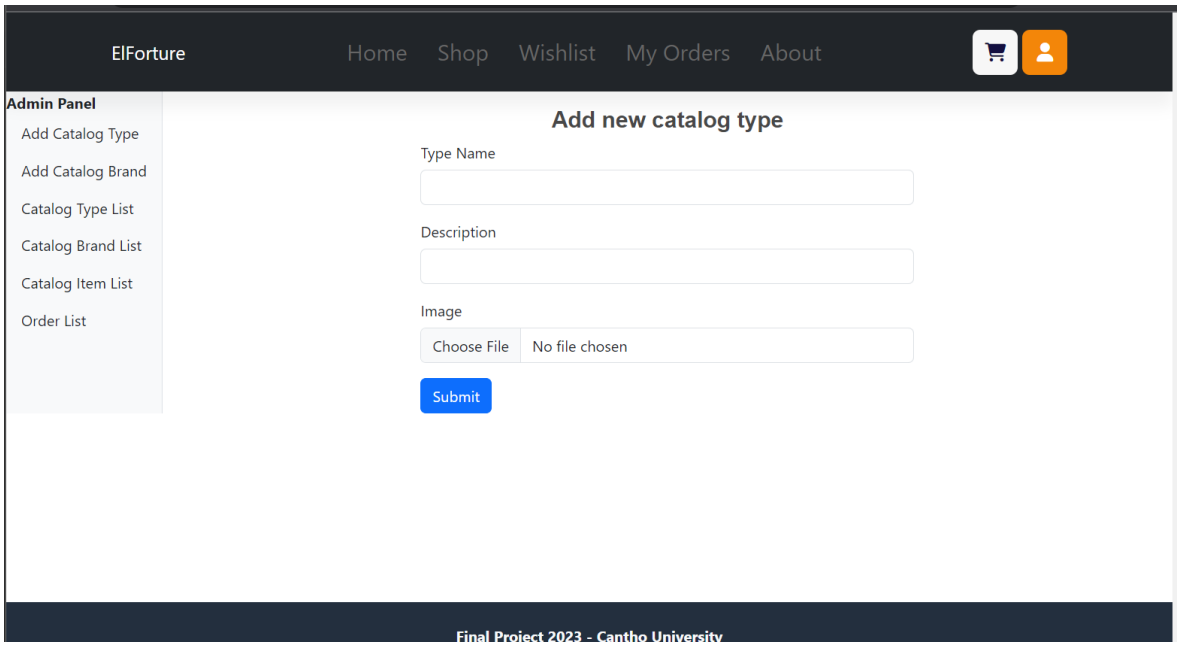


Figure 3.13: View Types

4.8 Admin - Catalog Brand

The Administrator - Brand Management function is a crucial part of the E-Commerce application. It allows the administrator to manage the product types available on the platform. This function includes several components: View All Brands, View Brand By ID, Create Brand, and Update Brand.

View All Brands

The View All Categories component retrieves all the brands from the database.

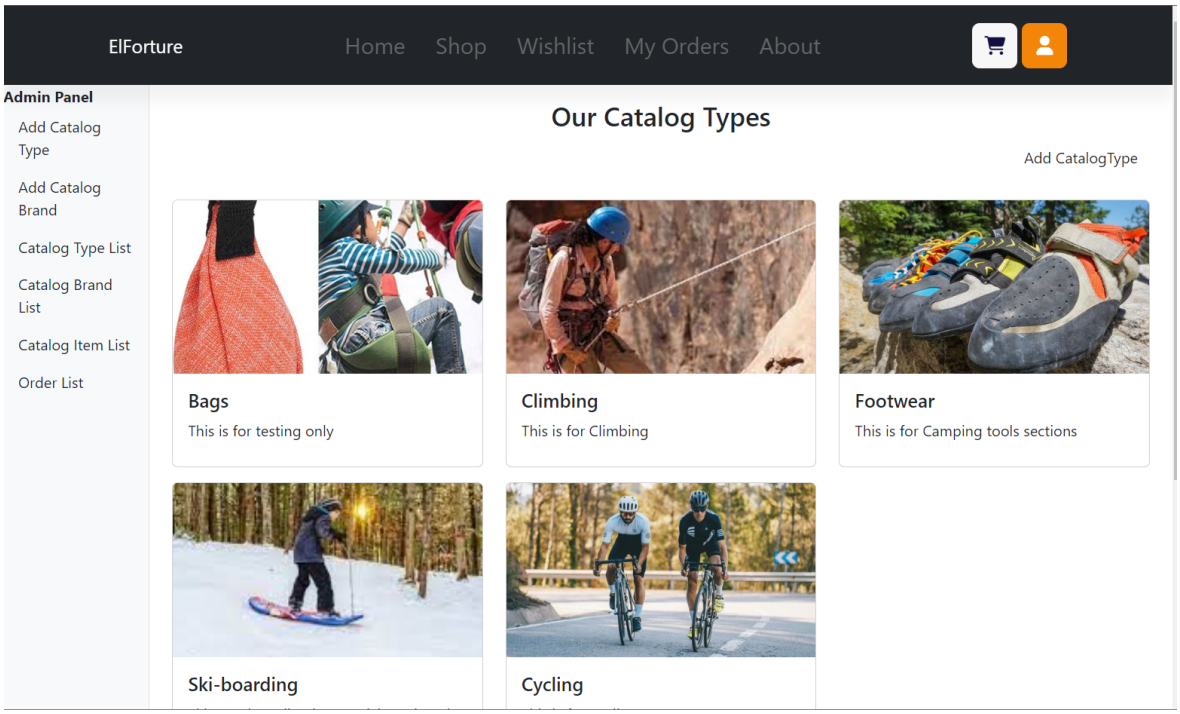


Figure 3.14: Brand View Types

Add new brand

TThe Add Brand component allows the administrator to add a new type to the database.

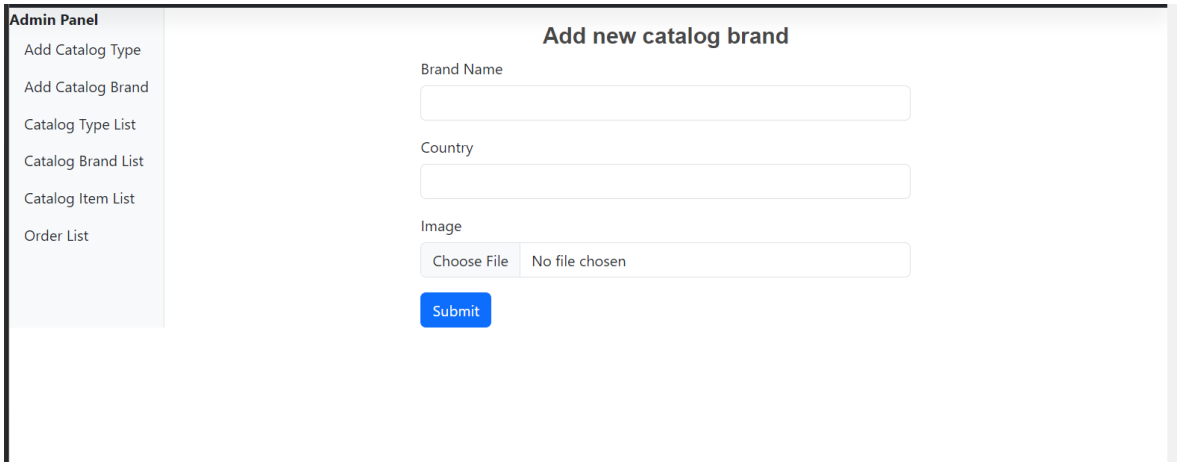


Figure 3.15: Addd New Brand

4.9 Admin - Catalog Product

The Administrator - Product Management function is a key part of the E-Commerce application. It allows the administrator to manage the products available on the platform. This function includes several components: Get All Products, Get Product By ID, Add Product, Update Product Information.

View All Products

The View All Categories component retrieves all the brands from the database.

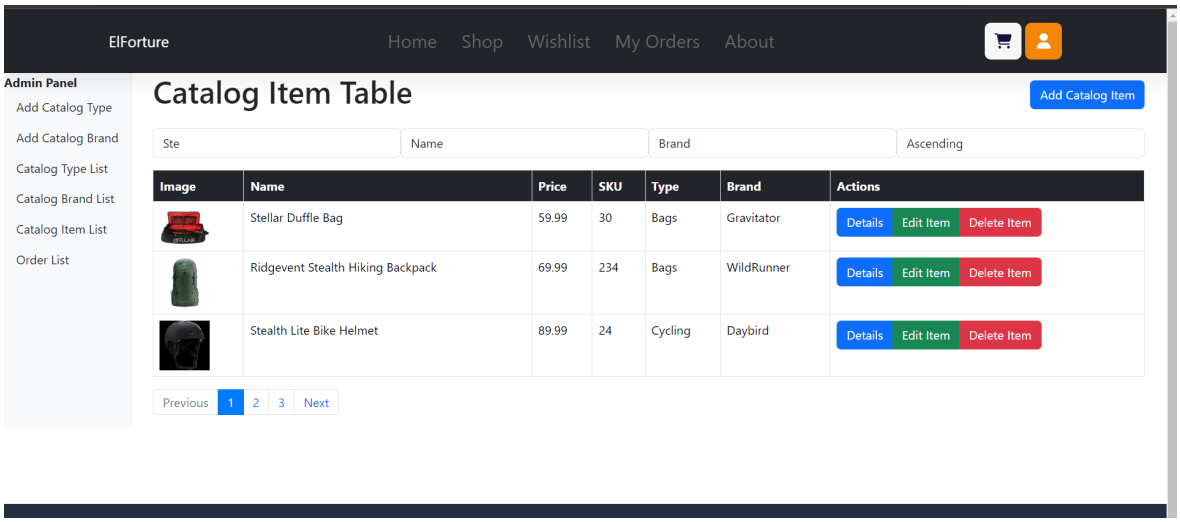


Figure 3.16: View Types

Add new product

TThe Add Category component allows the administrator to add a new product to the database.

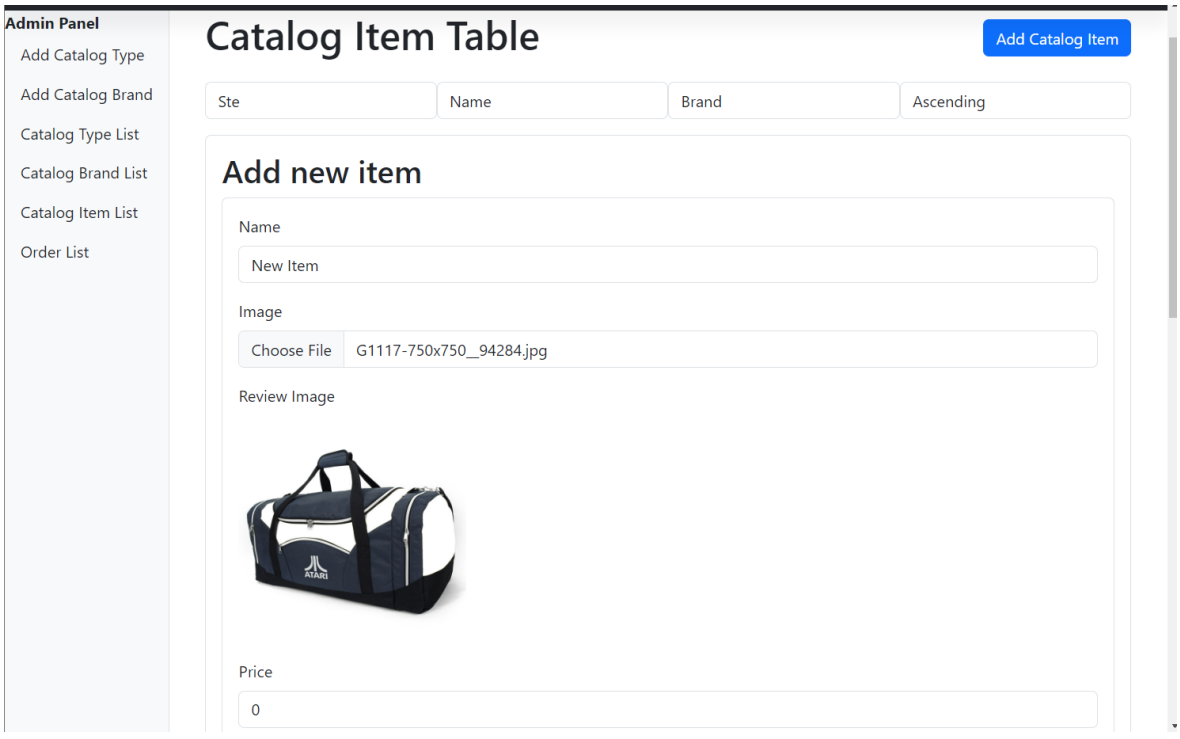


Figure 3.17: add Catlog Item

Edit

TThe Add Category component allows the administrator to add a new product to the database.

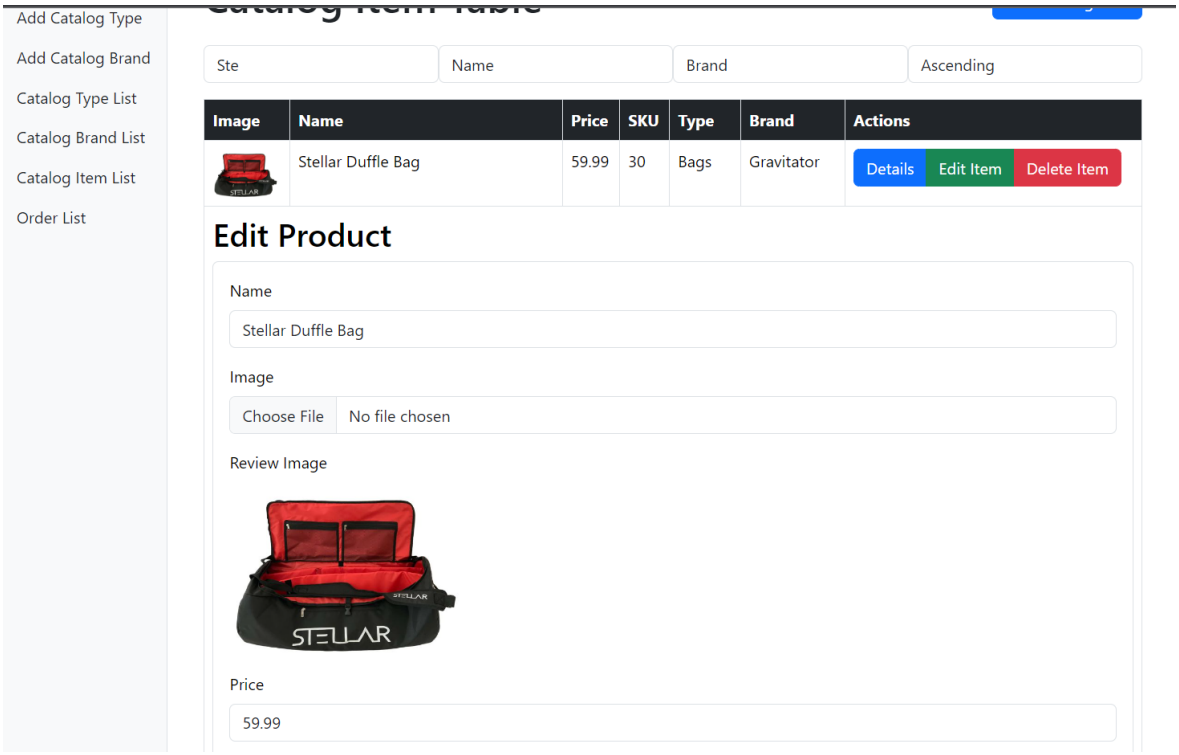


Figure 3.18: edit Catlog Item

4.10 Admin - Customer Orders

The Administrator - Order Management function is a vital part of the E-Commerce application. It allows the administrator to manage the orders placed on the platform. This function includes several components: View All Orders, View Order By ID, and Update Order Status.

View All Orders

The View All Categories component retrieves all the brands from the database.

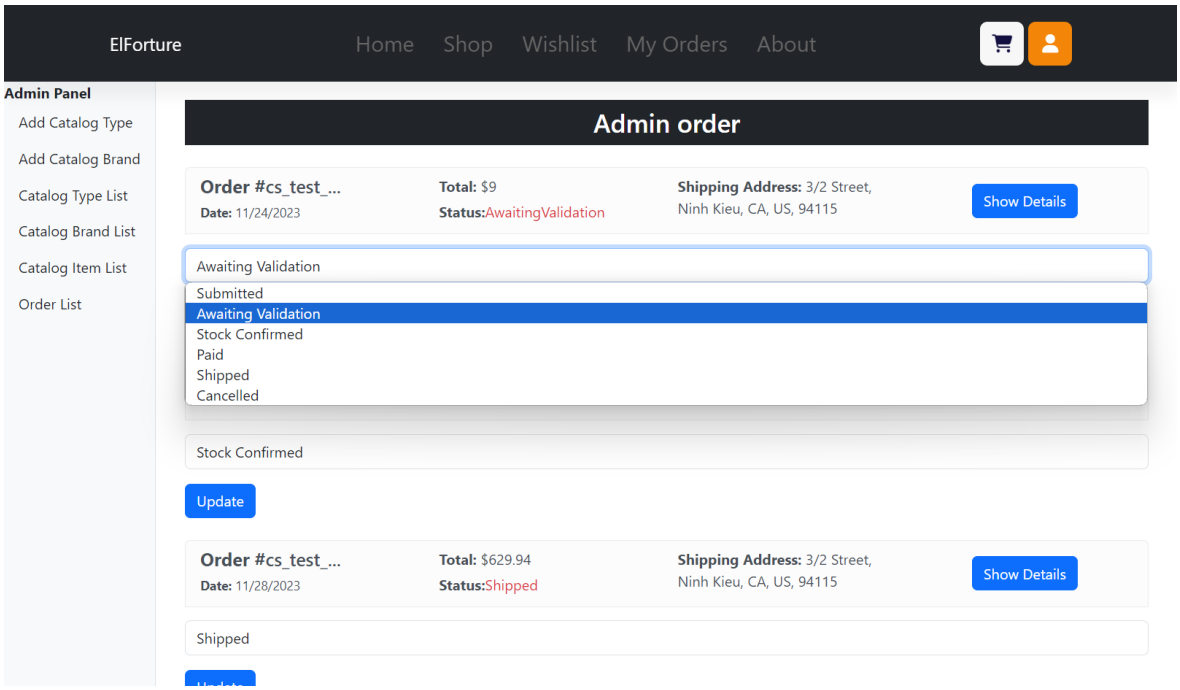


Figure 3.19: View All Orders

Chapter 4

Testing and evaluation.

1 Section 1

Part III

Conclusion

title