HUTECH
Đại học Công nghệ Tp.HCM

OUM
OPEN UNIVERSITY MALAYSIA

FINAL YEAR PROJECT

Student's Name: Vo Phu Thinh

Student Id: 2154030043

Class: 21BOIT01

Major: Information Technology

Course Code: CBBR4106

Courser Name: Final Year Project

Supervisor's Name: Le Thi Ngoc Tho

Email: vophuthinhcm@gmail.com

Ho Chi Minh City

28th December, 2024

E-COMMERCE WEBSITE WITH SENTIMENT ANALYSIS FOR REVIEWS

VO PHU THINH

A Final Year Project submitted in fulfillment of the requirements
for the degree of
Bachelor of Information Technology

Open University Malaysia
2024

DECLARATION

Name: Vo Phu Thinh

Matric Number: 2154030043

I hereby declare that this final year project is the result of my own work, except for quotations and summaries which have been duly acknowledged.

Signature: Date: 28/12/2024

E-COMMERCE WEBSITE WITH SENTIMENT ANALYSIS FOR REVIEWS

ABSTRACT

This project focuses on developing an e-commerce platform integrated with sentiment analysis capabilities. The platform aims to enhance customer experience, empower sellers with efficient management tools, and provide admins with robust system oversight. By leveraging natural language processing, the sentiment analysis feature processes customer feedback to extract actionable insights, enabling businesses to improve service quality and adapt to customer preferences. Built using ReactJS for the frontend, Node.js with Express.js for the backend, and MySQL for database management, the system is designed to be scalable and adaptable for future enhancements. The implementation ensures smooth navigation, secure transactions, and the potential for advanced functionalities like multilingual support and AI-driven recommendations in the future.

Keywords: E-commerce, Sentiment Analysis, Customer Experience, Scalable System, NLP

ACKNOWLEDGEMENT

I want to take this opportunity to express my gratitude and appreciation to my supervisor, Le Ngoc Tho, for their guidance, patience, and invaluable advice throughout this project.

I also want to express my heartfelt appreciation to my family and friends for their endless support whenever I faced challenges. Without the support of these parties, it would have been impossible for me to complete this project report successfully.

THANK YOU

VO PHU THINH

28th December, 2024

Table of Contents

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1: INTRODUCTION

1.1 Background of the Study

Along with globalization and the rapid development of information technology, e-commerce plays an increasingly important role, helping businesses expand their markets and enabling customers to shop easily anytime, anywhere. In Vietnam, e-commerce has continuously developed thanks to improved technology infrastructure, convenient electronic payments, and efficient logistics services.

However, the e-commerce market is becoming more competitive. Businesses must focus on product and service quality and understand customers' feelings and needs. As consumers frequently share opinions online, analyzing sentiments from comments has become the key to understanding preferences, improving the shopping experience, and building customer trust.

Based on this need, the project will build an e-commerce website integrated with sentiment analysis features for customer comments. This system will help businesses manage customers better, optimize marketing strategies, enhance the shopping experience, strengthen their brand, and increase competitiveness in the market.

1.2 Objectives of the Study

The objective of the study is to develop a user-friendly e-commerce website. This website will help customers easily search, compare, and purchase products safely and conveniently. In addition, the system will integrate sentiment analysis technology to better understand customers' psychology, preferences, and shopping trends through their comments. From this, businesses can improve the shopping experience, build trust, enhance service quality, and optimize business decisions.

1.3 Scope and Limitation

- Scope

The study focuses on building a user-friendly e-commerce website that allows customers to easily search, compare, and purchase products. The system will include a sentiment analysis feature based on user comments and product reviews to understand customer psychology and

shopping trends. The sentiment analysis results will improve customer experience and support businesses in optimizing marketing strategies, managing products, and making better business decisions. The platform will be designed with flexibility and scalability to ensure continuous development and competitiveness in the rapidly changing e-commerce market.

- Limitation

Sentiment analysis will primarily be based on text comments in English and will not include multi-language, image, or audio analysis. The accuracy of sentiment analysis depends on the quality of training data and natural language processing techniques, which may not detect sarcasm or implied language effectively. Initially, the system will focus only on core functionalities and not include advanced features (e.g., product recommendations based on user behavior) due to time, resources, and data limitations.

1.4 Problem Statement

In today's highly competitive e-commerce landscape, businesses face significant challenges in understanding their customers' sentiments and preferences. Online comments and reviews provide valuable feedback, but analyzing them manually takes too much time, is inconsistent, and can make mistakes. Without a good sentiment analysis system, businesses might miss important insights to improve the shopping experience, product quality, and market position.

Not understanding customer sentiments makes it harder to make the best decisions, improve marketing strategies, and build trust and loyalty. This is why there is a strong need for an automated system that can analyze user comments, understand sentiments accurately, and turn these insights into useful improvements for e-commerce platforms.

CHAPTER 2: SYSTEM ANALYSIS AND DESIGN

2.1 Identifying Requirements

An e-commerce platform must be developed to support online business operations, attract new customers, and optimize overall efficiency. The system will provide distinct functionalities for each user category:

User (Visitor):

+ Accesses the website without engaging in purchasing activities.
+ Can view product information, promotions, and general store details.
+ May decide to register later to become a Customer.

Customer (Registered Shopper):

+ Holds a registered account and participates in shopping activities.
+ Add products to a shopping cart or wish list, make purchases (via PayPal or cash on delivery), and request returns.
+ Can track orders and rate products once delivery is completed.
+ Can view and edit personal information (including phone number and address).

Seller (Store):

+ Registers as a seller to add, delete, and edit product listings.
+ Manages orders, including updating order statuses and communicating with customers.
+ Creates and manages promotional events, discount vouchers, and store information.
+ Can withdraw earnings from the system.
+ Views sentiment analysis results from customer reviews to adapt business strategies or improve services.

Admin (System Administrator):

+ Manages user accounts (Users, Customers, and Sellers), overseeing all store activities.
+ Monitors orders across the system and handles withdrawal requests from Sellers.
+ Manages product information (view-only) and promotional events (view-only).
+ Ensures overall security, data integrity, and smooth operation of the platform.

2.2 Use Case Diagram

2.2.1 Identifying Actors

+ User: Visitors to the website who do not engage in shopping activities.
+ Customer: Users with registered accounts who participate in shopping activities.
+ Seller (Store): Individuals or organizations with seller accounts responsible for managing products and orders.
+ Admin: The system administrator who manages and supervises all system activities.

2.2.2 Identifying Use Cases

Common Functions for All Actors:

- Register an account
- Log in
- View product information and promotional events
- View store information
- Add products to the shopping cart or wish list
- Purchase products or request returns (payment via PayPal or cash on delivery)
- Rate products (only after successful delivery)
- Send/receive messages
- View and edit personal information (including phone number and address)
- View and track orders

Table 1: Overview of User Roles and Actions

Actor	**Use Case**
User	Register an account Log in View product information View promotional events Add products to the shopping cart Add products to the wish list View store information
Customer	Register an account Log in View product information View promotional events Add products to the shopping cart Add products to the wish list View and edit personal information Make payments (via PayPal or cash on delivery) Request product returns Rate products (only after successful delivery) Send/receive messages View and track orders
Seller	Register as a seller Log in Add or delete products

	Manage orders Add or delete promotional events Withdraw funds Send/receive messages with customers Add or delete vouchers (discount codes) Edit store information View sentiment analysis results (from customer comments, visible only to Sellers)
Admin	View order information across the system Manage all sellers (stores) Manage all user accounts in the system Manage all products (view only, cannot delete) Manage all events (view only, cannot delete) Process withdrawal requests from sellers (stores)

2.2.3 General Use Case Diagram

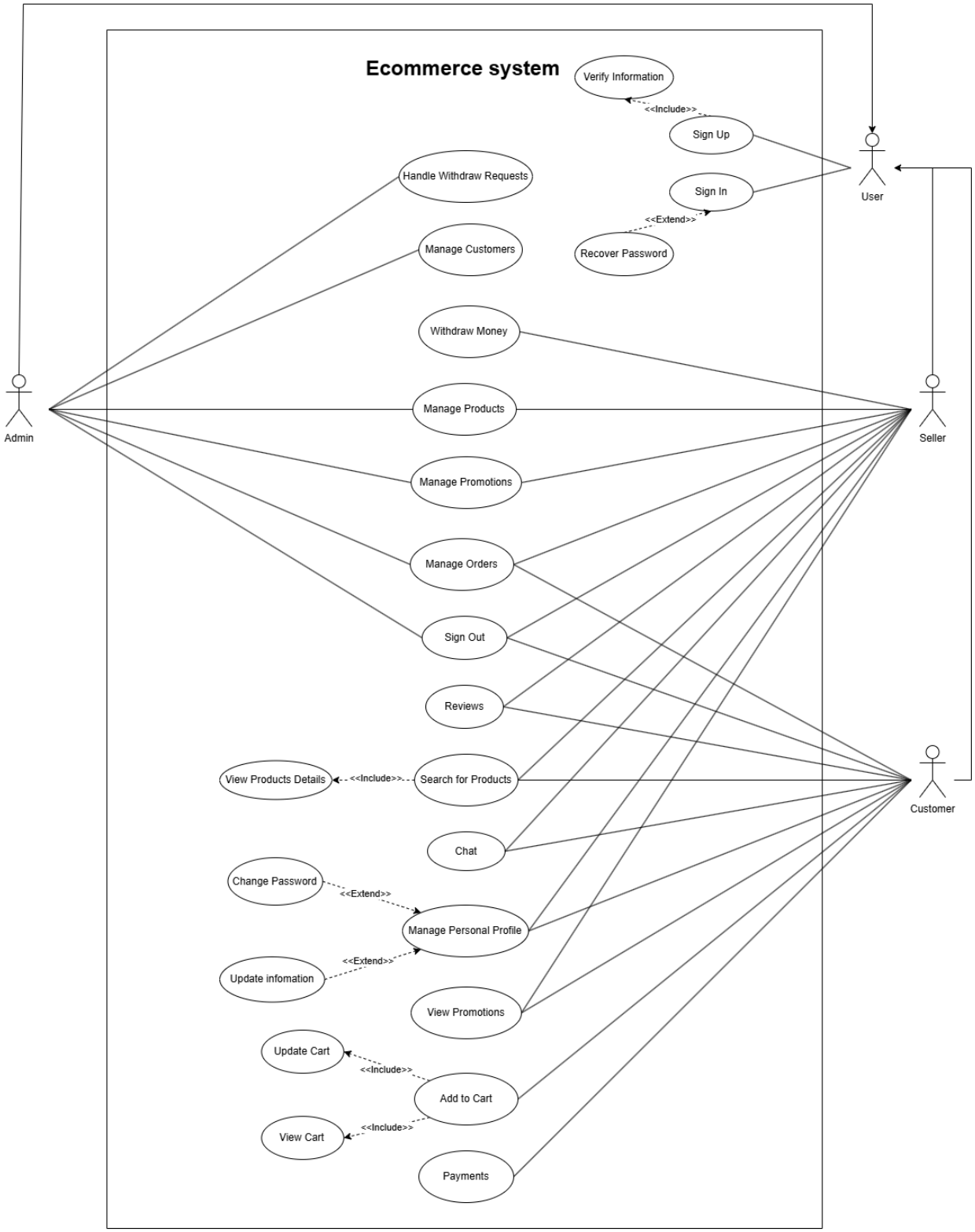

Figure 1: General Use Case Diagram

2.3 Use Case Specification

2.3.1 Use Case Specification: "Login"

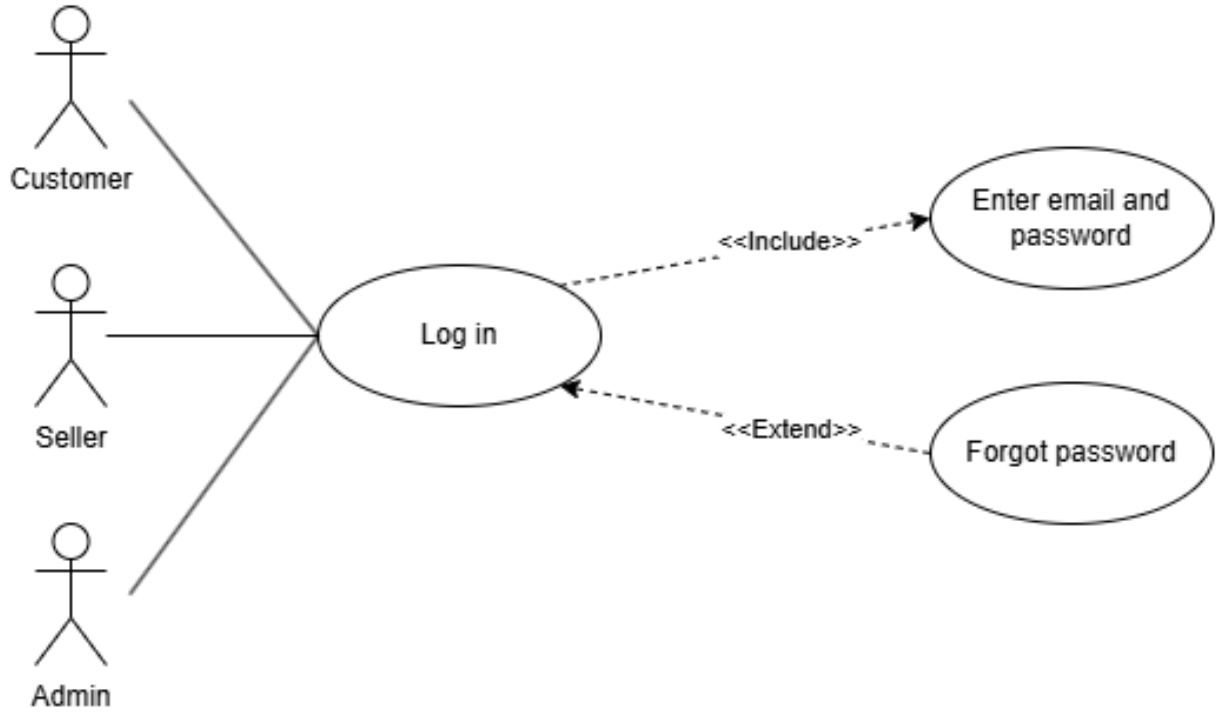

Figure 2: Use Case Specification: "Login"

Table 2: Use Case Specification: "Login"

Use Case Name: Log In	**Priority**: High
Primary Actors: Customer, Seller, Admin	**Type of Use Case**: Detailed, Essential
Stakeholders and Interests - Customer: Wants to log into the system to browse products, add to cart, and make purchases. - Seller: Wants to log into the system to manage products, view orders, and handle store operations. - Admin: Wants to log into the system to manage users, products, and overall system configurations.	

Brief Description

- This use case describes the process by which a Customer, Seller, or Admin logs into the system. It includes entering valid credentials and extends to a password recovery option if the user has forgotten their password.

Constraints:

- Customers, Sellers, and Admins must select the "Log In" option to access their respective features.

Type: External

Relationships

- Association: Customer, Seller, Admin

Main Flow of Events

1. The actor (Customer, Seller, or Admin) selects the "Log In" option on the website.
2. The system includes the sub-process Enter email and password (<<include>>).
3. Enter email and password: The system prompts the actor to input valid credentials (email/username and password).
4. The system verifies the credentials:

- If valid, the system logs the actor in and redirects them to their respective dashboard or homepage.
- If invalid, the system displays an error message.

5. The actor is now authenticated and can access functionalities based on their role (Customer, Seller, or Admin).

Sub-flows

1. Processing Log In

- The system verifies the account credentials (email/username and password).
- Displays a message indicating whether the login was successful or unsuccessful.

2: Processing Log Out

- The system logs out the actor and displays a message confirming successful logout.

2.3.2 Use Case Specification: " Manage Products"

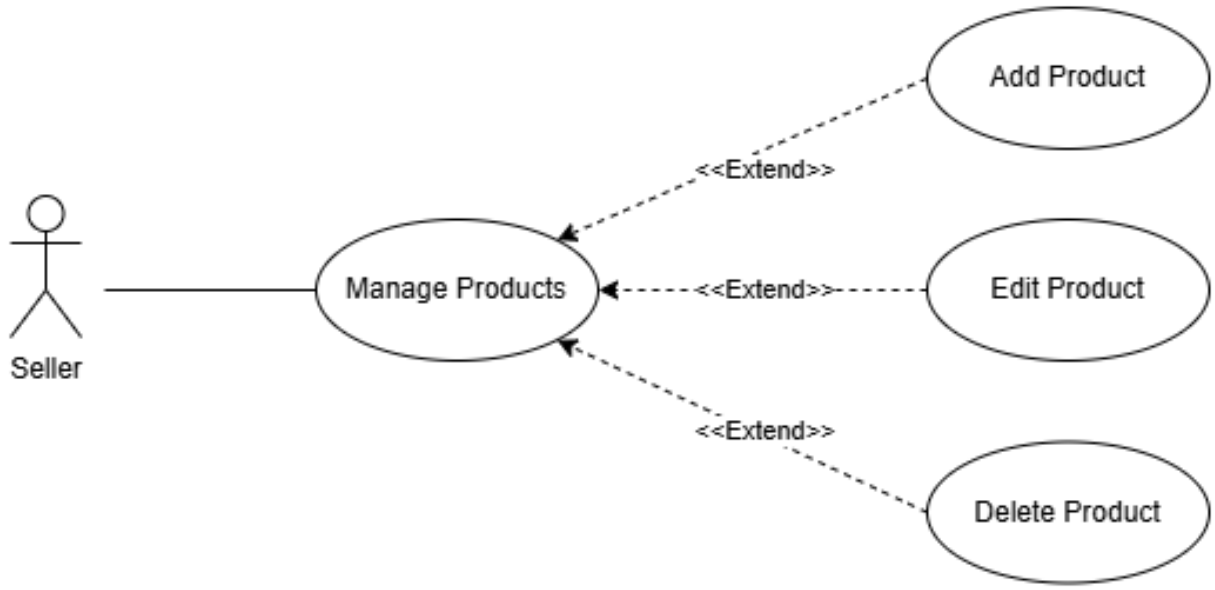

Figure 3: Use Case Specification: " Manage Products"

Table 3: Use Case Specification: " Manage Products"

Use Case Name: Manage Products	**Priority**: High
Primary Actors: Customer, Seller, Admin	**Type of Use Case**: Detailed, Essential
Stakeholders and Interests - Seller: Wants to manage products by adding, deleting, or editing product information.	
Brief Description - This use case describes the process by which a Seller manages products, including adding, editing, and deleting product details.	
Constraints - The Seller must log in to the system using their account and select the “Manage Products” functionality. **Type:** Internal	
Relationships - Association: Seller	

Main Flow of Events

1. The Seller logs in to the system using their account.
2. The system displays the seller dashboard.
3. The Seller selects the "Manage Products" functionality.
4. The system displays the product management interface with options to add, edit, or delete products.
5. The Seller performs one of the following actions:

- Add a new product: Enters product details and the system validates and saves them.
- Edit an existing product: Select a product, update the details, and confirm changes.
- Delete a product: Select a product, confirm the deletion, and remove it from the system.

2.3.3 Use Case Specification: " Manage Orders"

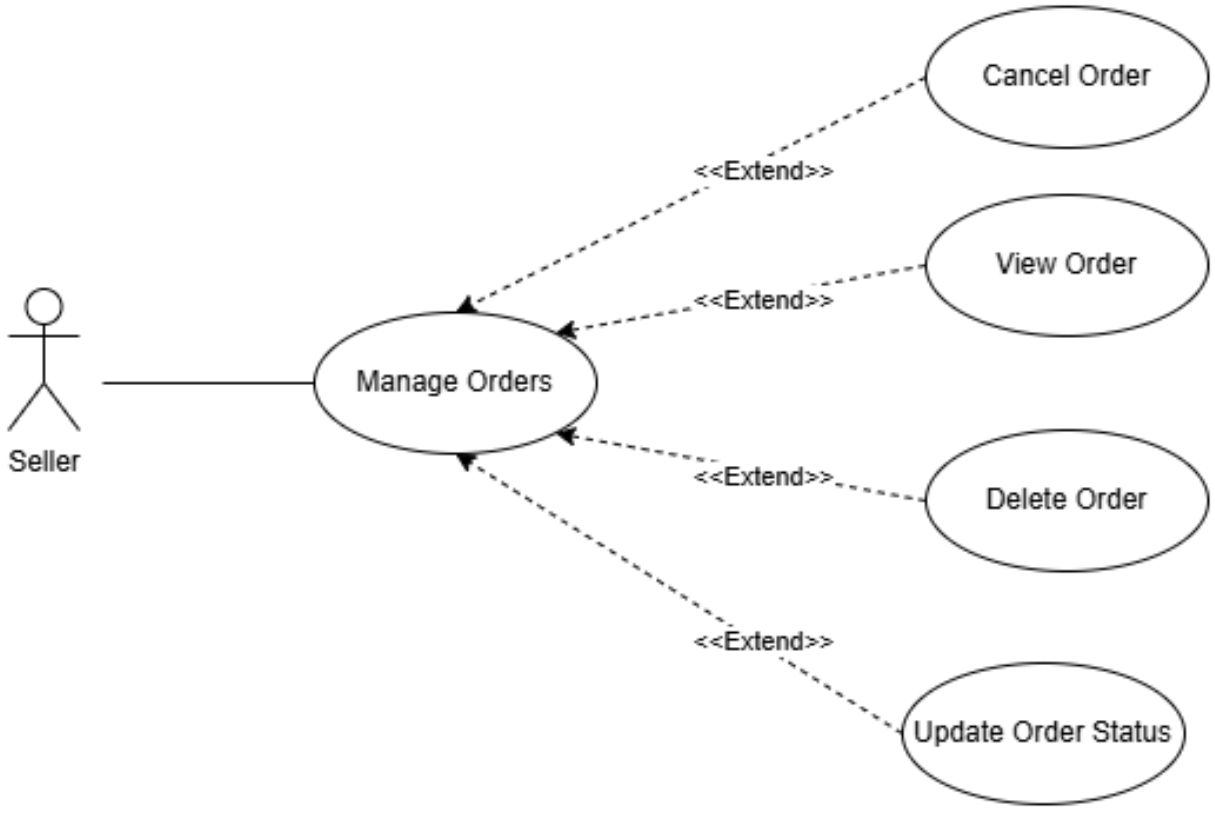

Figure 4: Use Case Specification: " Manage Orders"

Table 4: Use Case Specification: " Manage Orders"

Use Case Name: Manage Orders	**Priority**: High
Primary Actors: Seller	**Type of Use Case**: Detailed, Essential
Stakeholders and Interests - Seller: Wants to manage orders by viewing, canceling, deleting, or updating the status of orders efficiently.	
Brief Description - This use case describes how the Seller manages orders, including viewing order details, canceling orders, deleting orders, and updating order statuses.	
Constraints - The Seller must log in to the system using their account and select the “Manage Orders” functionality. **Type:** Internal	
Relationships - Association: Seller	
Main Flow of Events 1. The Seller logs into the system using their account. 2. The system displays the Seller dashboard. 3. The Seller selects the "Manage Orders" functionality. 4. The system displays the interface for order management, allowing the Seller to perform the following actions: - View Order: View detailed information about an order. - Cancel Order: Cancel an active order. - Delete Order: Remove an order from the system. - Update Order Status: Modify the status of an order (e.g., "Processing," "Shipped," "Delivered").	

2.3.4 Use Case Specification: " Handle Withdraw Requests"

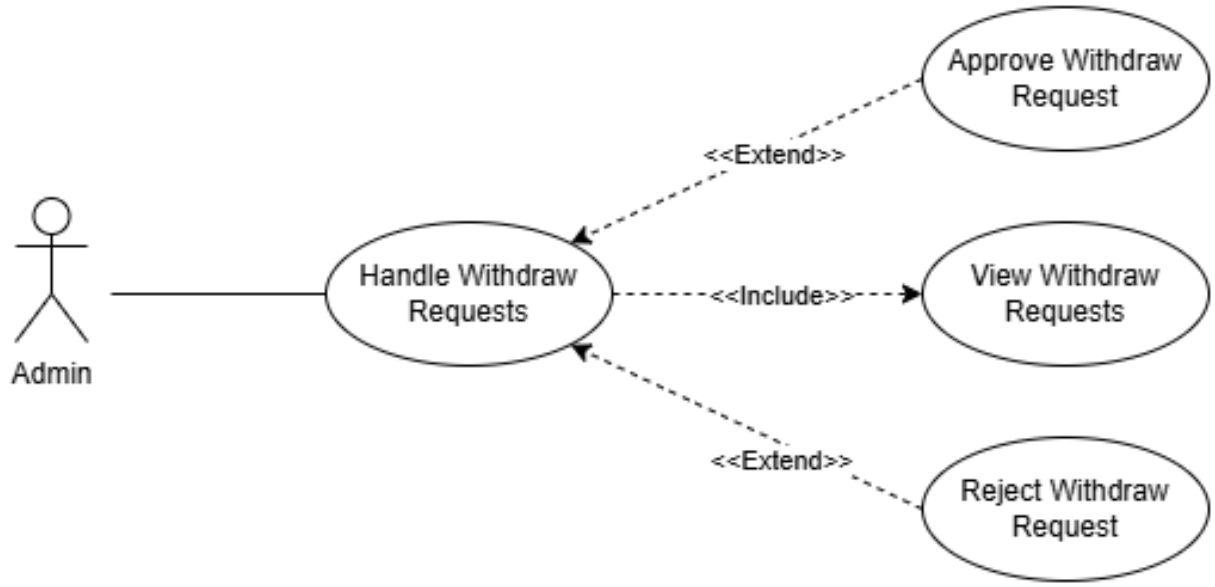

Figure 5: Use Case Specification: " Handle Withdraw Requests"

Table 5: Use Case Specification: " Handle Withdraw Requests"

Use Case Name: Handle Withdraw Requests	**Priority**: High
Primary Actors: Admin	**Type of Use Case**: Detailed, Essential
Stakeholders and Interests - Admin: Wants to efficiently manage withdraw requests from Sellers, including viewing, approving, and rejecting requests.	
Brief Description - This use case describes how the Admin handles withdraw requests submitted by Sellers. The Admin can view the list of requests, approve valid requests, and reject invalid or incomplete requests.	
Constraints - The Admin must log in to the system using their credentials and access the “Handle Withdraw Requests” functionality. **Type:** Internal	

Relationships - Association: Admin - Include: View Withdraw Requests - Extend: Approve Withdraw Request, Reject Withdraw Request
Main Flow of Events 1. The Admin logs into the system using their account. 2. The system displays the Admin dashboard. 3. The Admin selects the "Handle Withdraw Requests" functionality. 4. The system includes the sub-process View Withdraw Requests, displaying all pending withdrawal requests submitted by Sellers. 5. The Admin performs one of the following actions: - Approve Withdraw Request: Confirms and processes a valid request, transferring funds to the Seller. - Reject Withdraw Request: Marks the request as rejected and provides a reason for rejection.

2.3.5 Use Case Specification: " Withdraw Money"

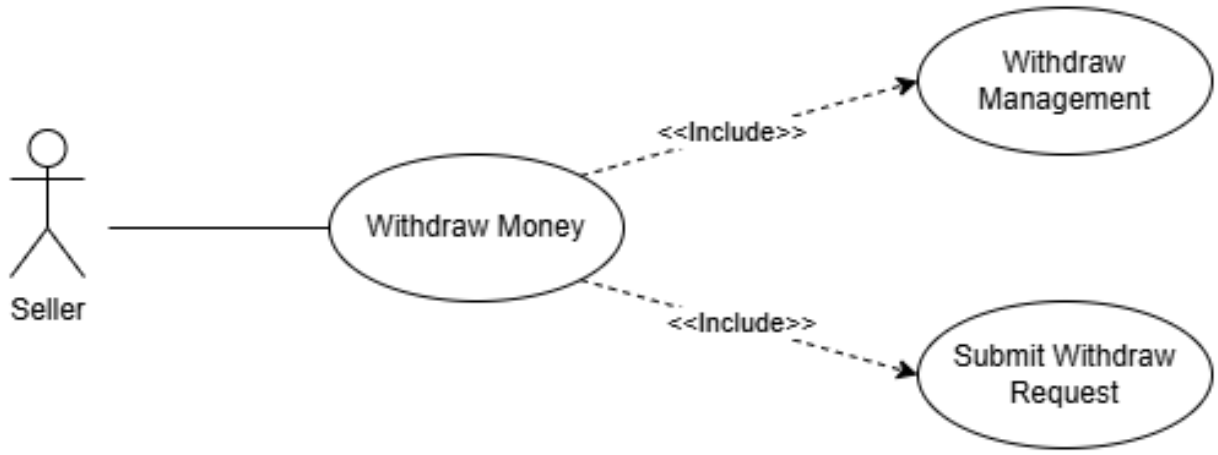

Figure 6: Use Case Specification: " Handle Withdraw Requests"

Table 6: Use Case Specification: " Withdraw Money"

Use Case Name: Withdraw Money	**Priority**: High
Primary Actors: Seller	**Type of Use Case**: Detailed, Essential
Stakeholders and Interests: - Seller: Wants to withdraw earnings by submitting a request and managing withdrawal details.	
Brief Description - This use case describes how the Seller initiates and manages the process of withdrawing money. It includes submitting a withdraw request and accessing withdrawal management features.	
Constraints: - The Seller must log in to the system using their account to access the withdrawal features. **Type:** Internal	
Relationships - Include: Submit Withdraw Request, Withdraw Management	
Main Flow of Events 1. The Seller logs into the system using their account. 2. The system displays the Seller dashboard. 3. The Seller selects the "Withdraw Money" functionality. 4. The system includes the sub-process Submit Withdraw Request, allowing the Seller to: - Enter the withdrawal amount. - Confirm the withdrawal request. - The system processes the request and adds it to the list of requests for Admin approval. 5. The system includes the sub-process Withdraw Management, enabling the Seller to: - View the status of previously submitted withdrawal requests. - Cancel or modify pending requests (if applicable).	

2.4 Sequence Diagram

2.4.1 Sequence Diagram for Seller and Customer

The system includes functionalities such as "Login," "Register ," and "Change Password," which follow similar workflows and interfaces for all user roles, including Seller and Customer.

- **Sequence Diagram: Login**

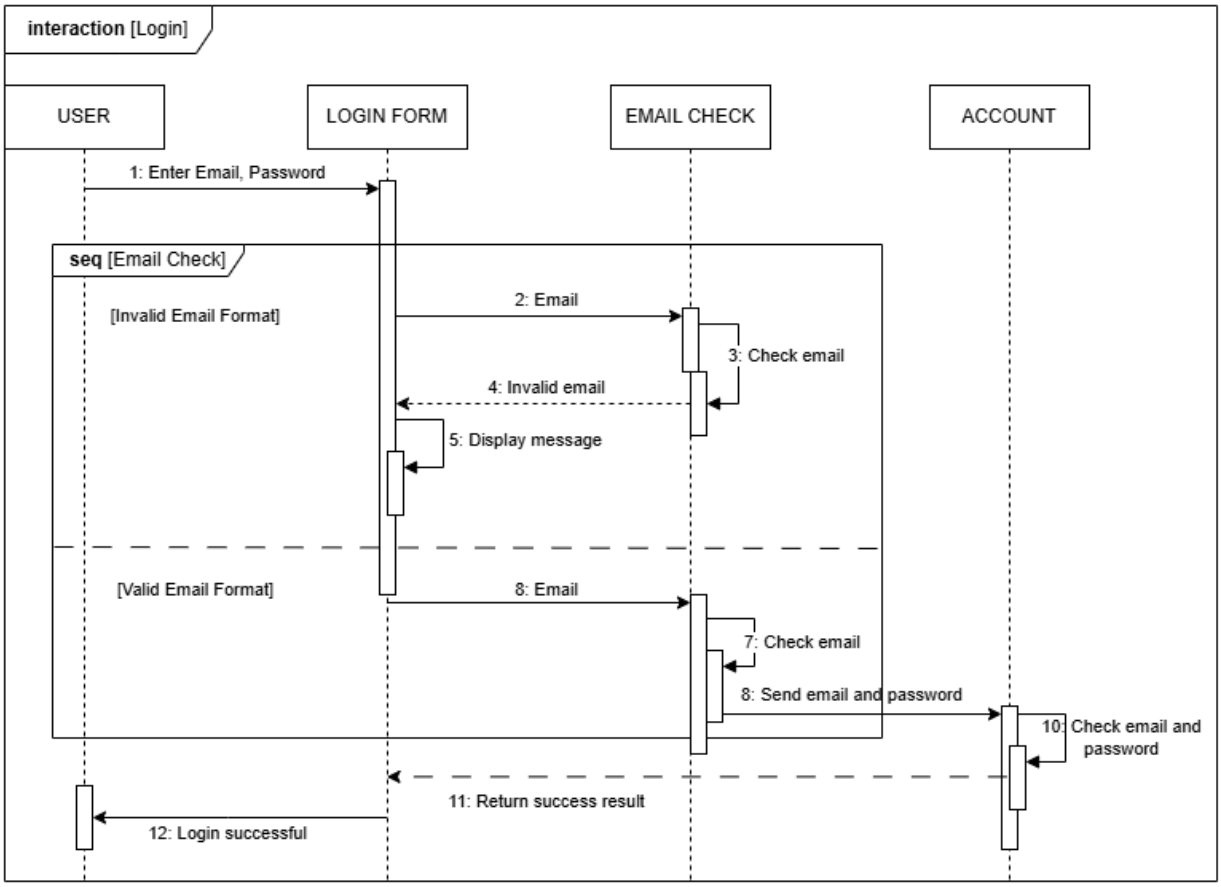

Figure 7: Sequence Diagram Login

Scenario Description: Login

Preconditions: None.

Main Scenario:

1. The user or seller enters their Email and Password.
2. The system displays the login interface.
3. The user inputs their information and sends a login request.
4. The system checks the format of the Email.

5. If the Email is valid:

- The system sends the Email and Password for verification.
- The system validates the information in the database.
- The system returns a successful login result.

6. The user or seller successfully logs in.

Alternative Scenarios:

Email Validation:

- The user enters an incorrectly formatted Email.
- The system checks and detects the invalid Email.
- The system displays a notification requesting the user to re-enter a valid Email.

- **Sequence Diagram: Registration**

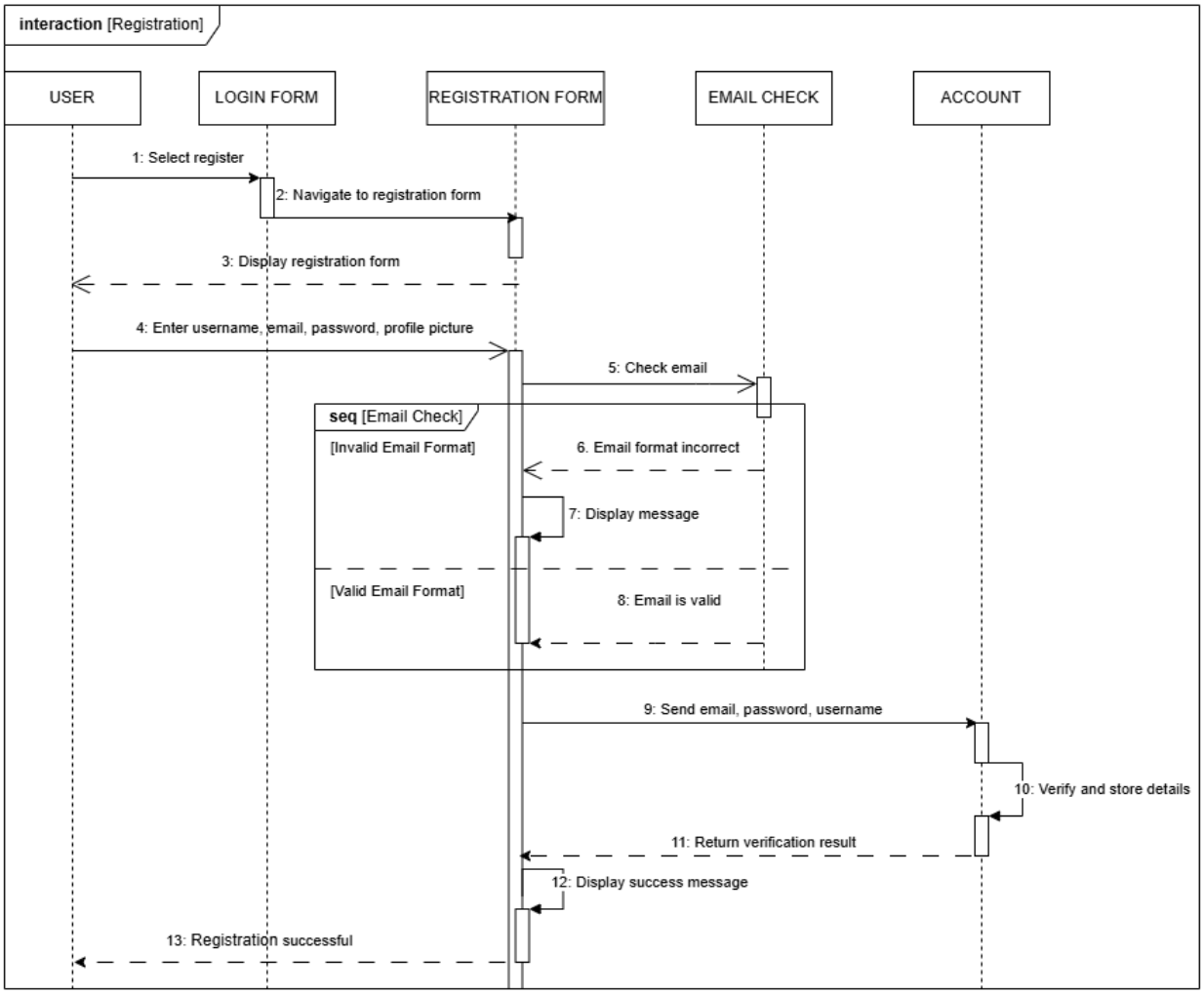

Figure 8: Sequence Diagram Registration

Scenario Description: Registration

Preconditions: None.

Main Scenario:

1. The user selects the **Register** function from the login interface.
2. The system displays the registration interface.
3. The user enters their registration information.
4. The system validates the information and creates an account for the user.
5. The system returns a notification to the user.

Alternative Scenarios:

- Email Format Validation:

1. The user enters an improperly formatted email.
2. The system displays a message and requests the user to re-enter the email.

- Email Existence Check:

1. The customer provides an email already registered by another user.
2. The system notifies users and requests them to use a different email.

- **Sequence Diagram: Change password**

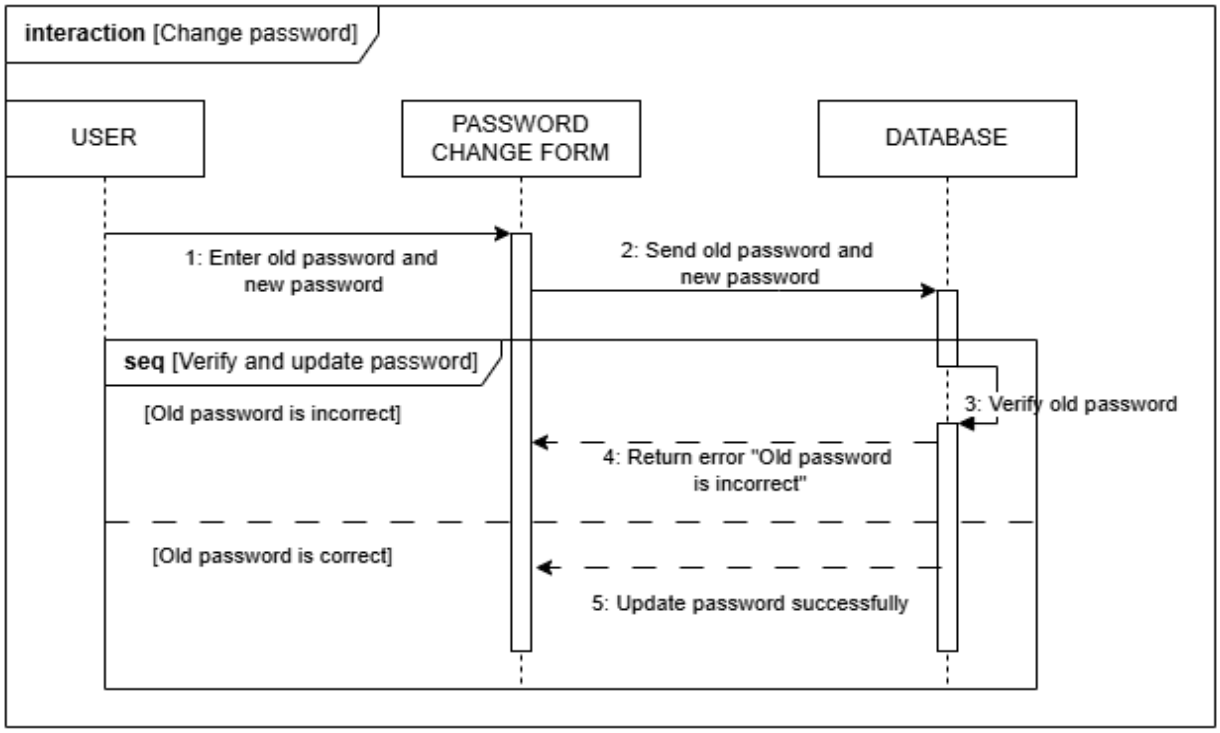

Figure 9: Sequence Diagram Change password

Scenario Description: Change Password

Preconditions: The user must be logged into the system.

Main Scenario:

1. The user selects the "Change Password" function from the user information update interface.
2. The user enters the old password and the new password.
3. The system verifies the input and returns the result to the user.

Alternative Scenarios:

- Incorrect Old Password:

1. The system detects that the old password provided by the user does not match the stored password in the system's database.
2. The system notifies the user and prompts them to re-enter the correct old password.

2.4.2 Sequence Diagrams for Individual Entities

2.4.2.1 Sequence Diagram for Customers

- **Sequence Diagram: Product Search**

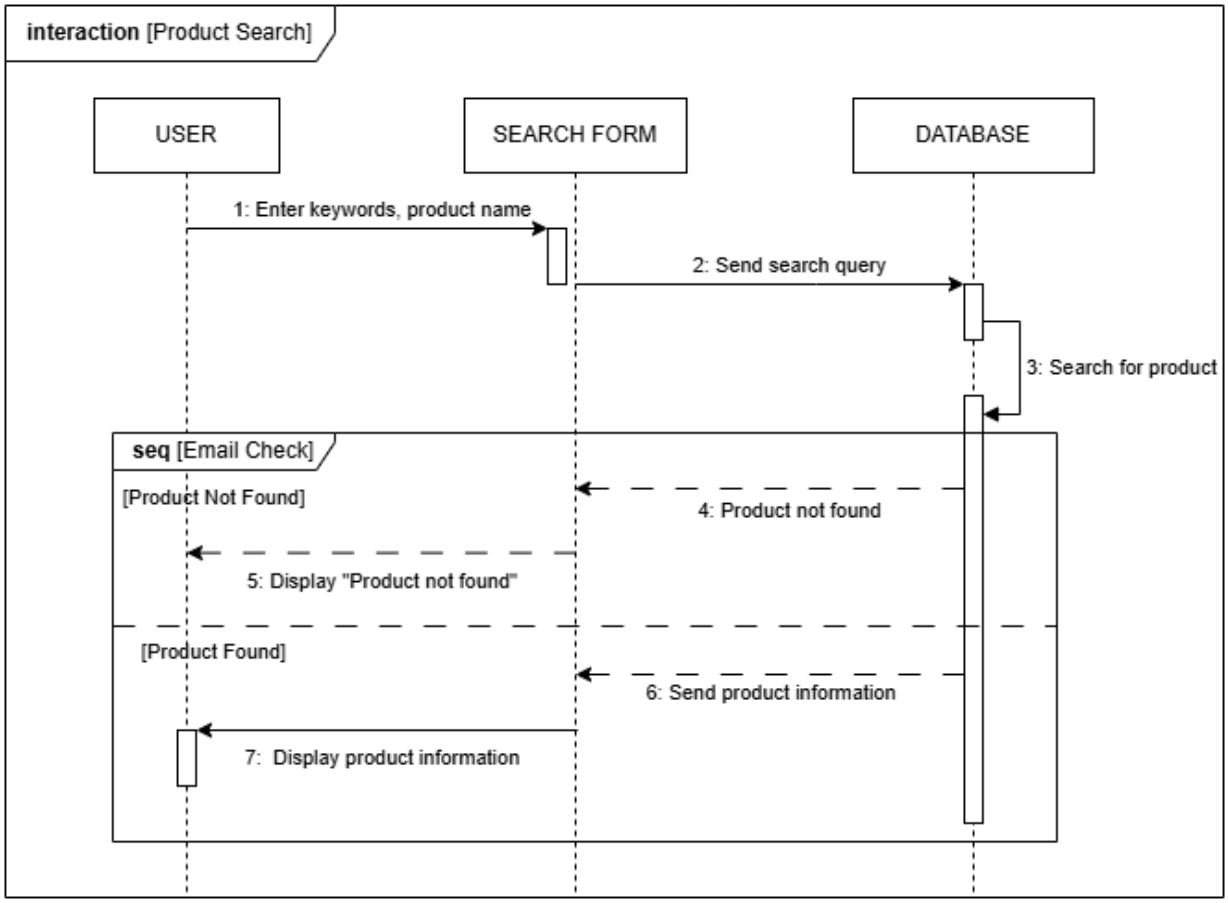

Figure 10: Sequence Diagram Product Search

Scenario Description: Product Search

Preconditions: None.

Main Scenario:

1. The user enters keywords or the product name in the search form.
2. The system sends the search query to the database.
3. The database searches for the product.
4. If the product is found:

- The database sends the product information back to the system.
- The system displays the product information to the user.

Alternative Scenarios:

If the product is not found:

- The database informs the system that the product does not exist.
- The system displays a message: "Product not found."

- **Sequence Diagram: Add Product to Cart**

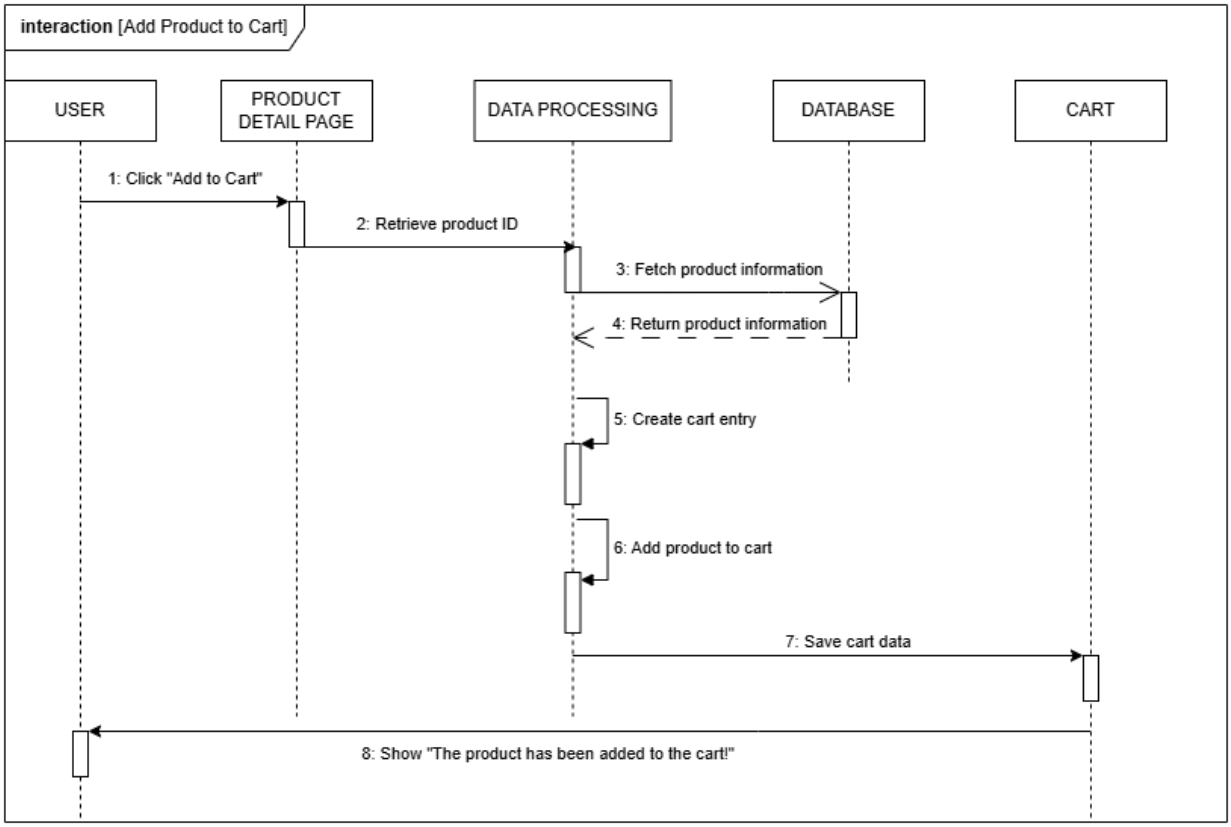

Figure 11: Sequence Diagram Add Product to Cart

Scenario Description: Add Product to Cart

Preconditions: None.

Main Scenario:

1. The customer views the product details.
2. The customer adds the product to their shopping cart if they wish to purchase it.
3. The system verifies and retrieves product information and then adds the product to the user's shopping cart.

Alternative Scenarios:

Out-of-Stock Products:

1. The user attempts to add a product to their shopping cart.
2. The system checks the product details and stock quantity. The system notifies the user if the stock is less than the requested quantity.

- **Sequence Diagram: Add Product to Favorites**

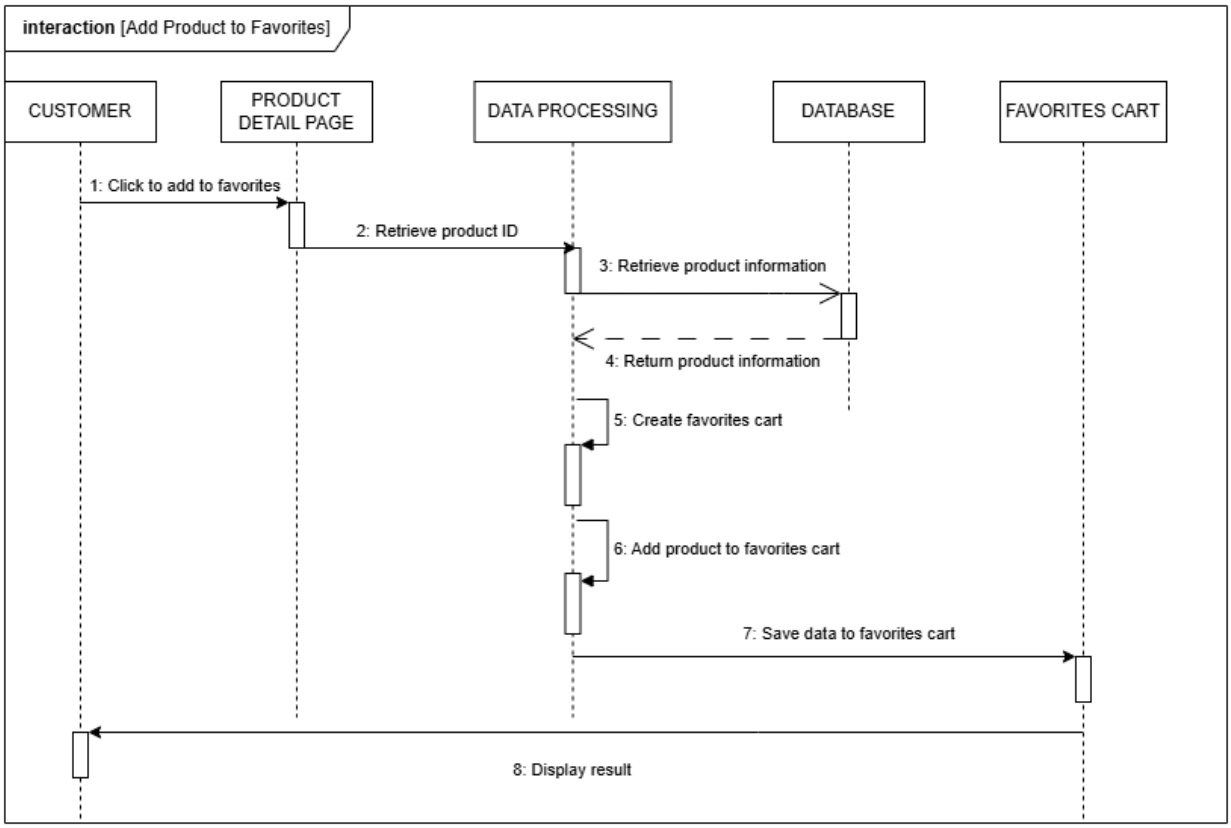

Figure 12: Sequence Diagram Add Product to Favorites

Scenario Description: Add Product to Favorites

Preconditions: None.

Main Scenario:

1. The customer wishes to track a product they like but does not want to purchase immediately. They choose to add the product to their favorite cart.
2. The system verifies and retrieves the product information and then adds the product to the favorites cart.
3. The system notifies the customer of the successful action.

- **Sequence Diagram: View Products by Category**

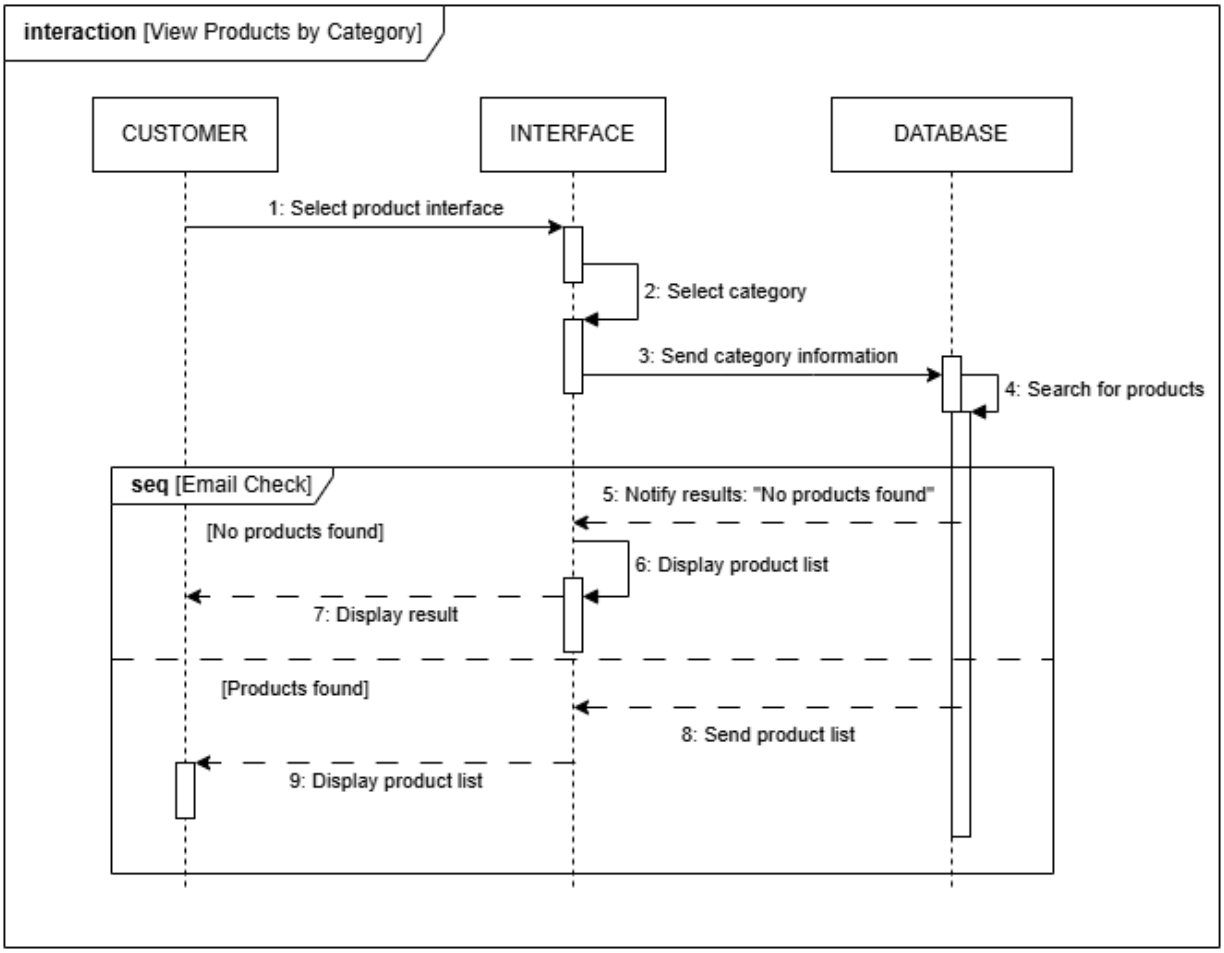

Figure 13: Sequence Diagram View Products by Categoryory

Scenario Description: View Products by Category

Precondition: None.

Main Scenario:

1. The user selects the product interface.
2. The system displays the login interface.
3. The user selects the category of the product they want to search for.
4. The system processes the request and returns the filtered results.

Alternative Scenarios:

No products found:

- The system queries the information entered by the customer but finds no matching results in the database.
- The system returns a message saying "No products found" and displays an empty result to the customer.

- **Sequence Diagram: Payment**

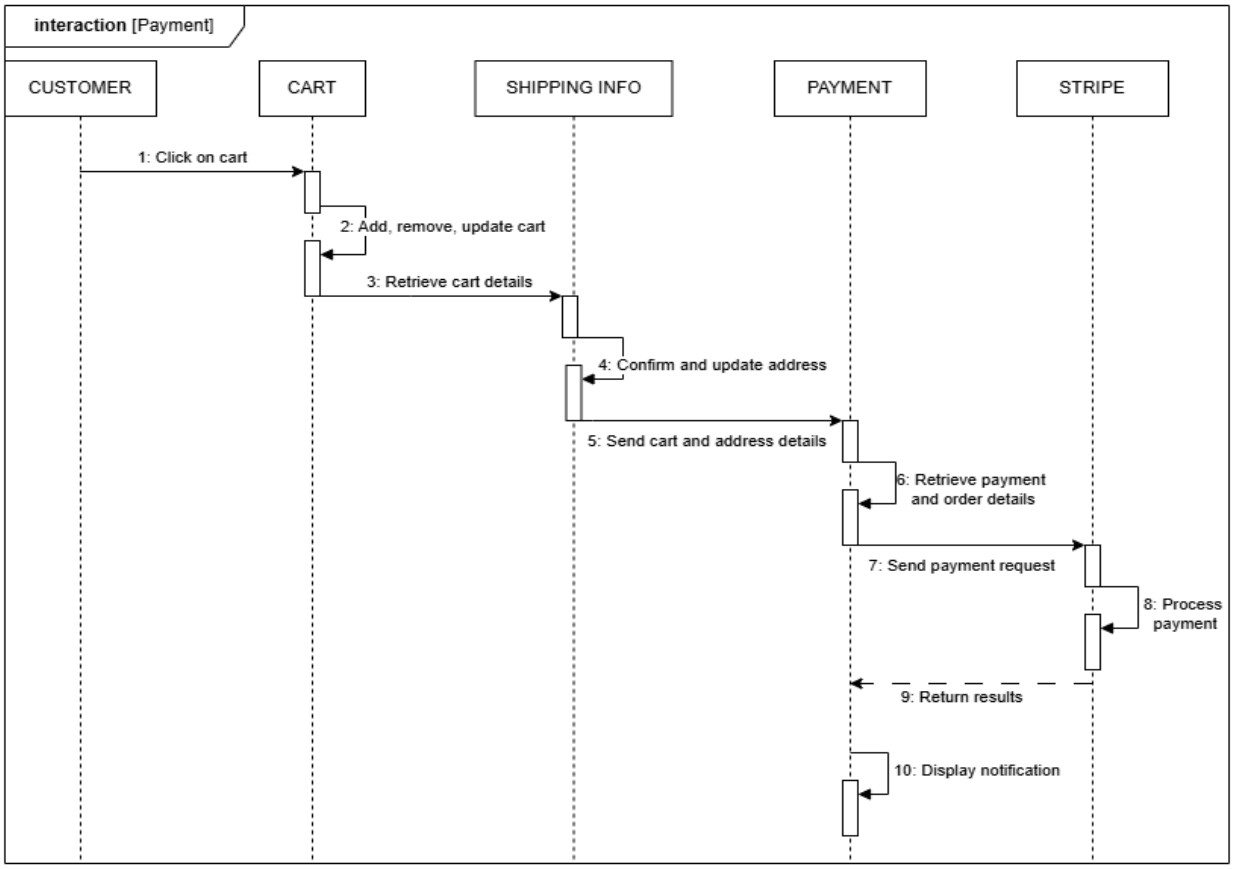

Figure 14: Sequence Diagram Payment

Scenario Description: Payment Process

Precondition: The customer is logged into the system.

Main Scenario:

1. The Customer clicks on the cart to proceed with their purchase.
2. The system allows customers to add, remove, or update items in their cart.
3. The system retrieves the updated cart details.

4. The customer confirms or updates their shipping address.
5. The system sends the cart details and the shipping address to the payment module.
6. The payment module retrieves the payment and order details.
7. A payment request is sent to a third-party payment gateway (Stripe).
8. Stripe processes the payment and sends the result back to the payment module.
9. The payment module returns the result to the system.
10. The system displays the payment notification (success or failure) to the customer.

Alternative Scenarios:

Invalid Payment Information:

- If the payment gateway detects invalid payment information, the system notifies the customer to re-enter valid details.

Payment Failure:

- If the payment fails (e.g., insufficient funds), the system notifies the customer and allows them to retry.

2.4.2.2 Sequence Diagram for Seller

- **Sequence Diagram: Add New Product**

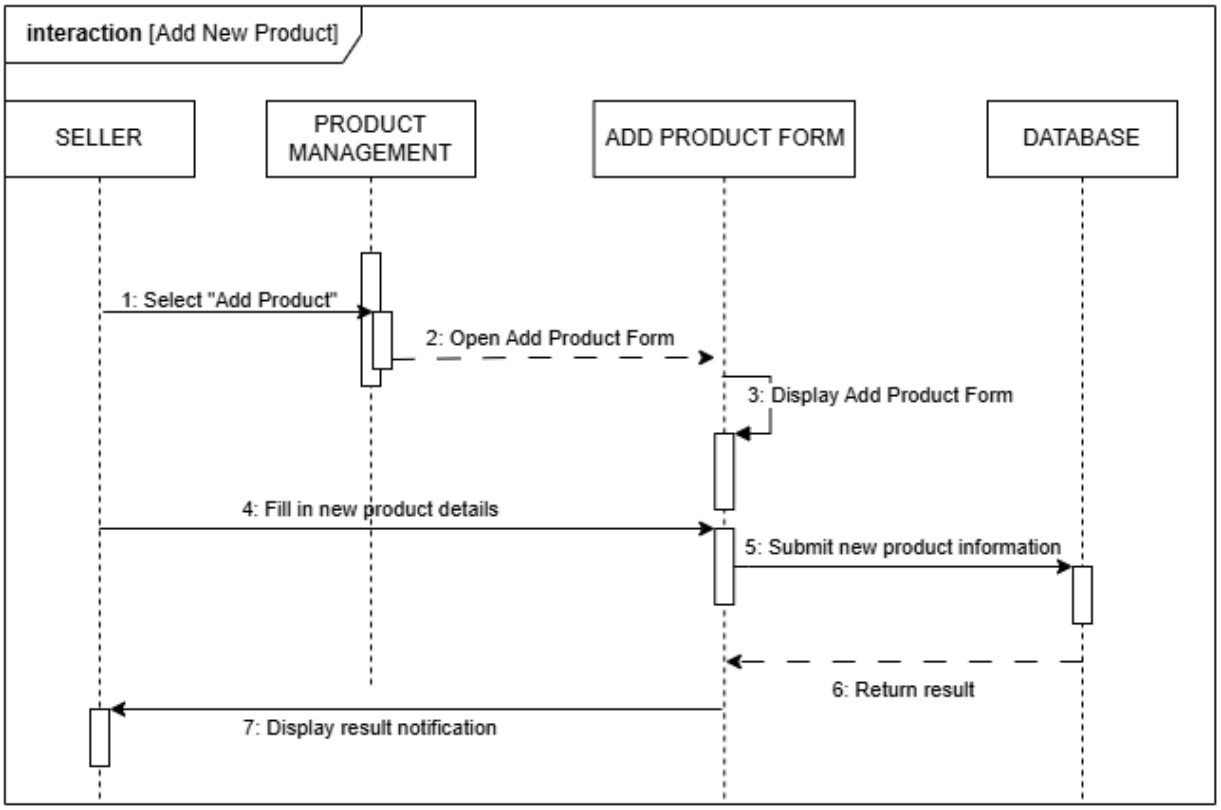

Figure 15: Sequence Diagram Add New Product

Scenario Description: Add a New Product

Precondition: The user must be logged in as a seller to access the product management interface.

Main Scenario:

1. The seller selects the "Add Product" function from the product management interface.
2. The system opens the Add Product Form interface.
3. The system displays the add new product form.
4. The seller enters the product details into the form.
5. The system sends the new product information to the database.
6. The database processes the information and returns the result.
7. The system displays the result notification (success or failure) to the seller.

Alternative Scenarios:

Incomplete or Invalid Product Information:

- The system checks and detects missing or invalid information (e.g., negative price, missing description...).
- The system prompts the seller to revise and resubmit the information.

- **Sequence Diagram: Edit Product Information**

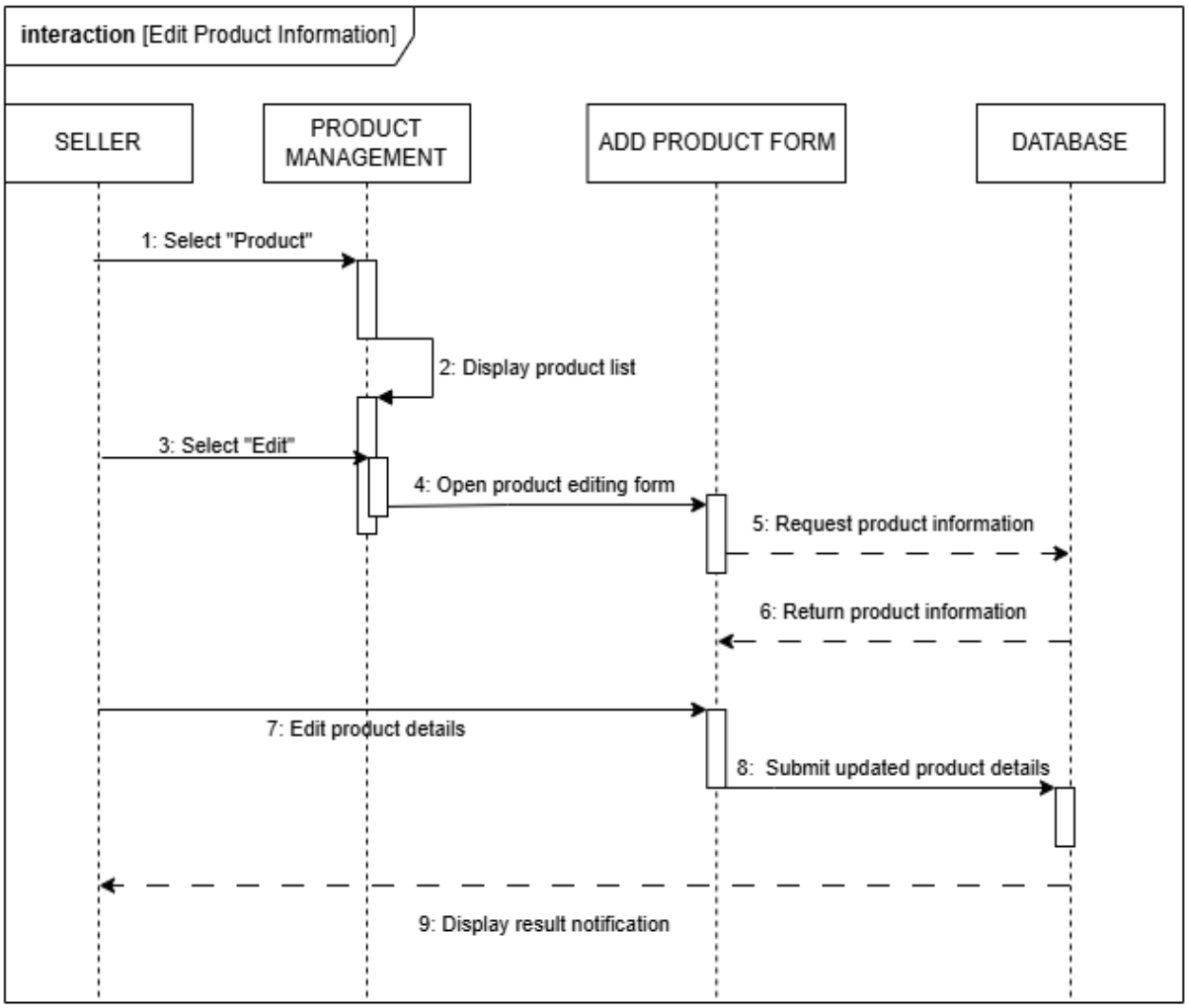

Figure 16: Sequence Diagram Edit Product Information

Scenario Description: Edit Product Information

Precondition: The seller must be logged in to access the Product Management interface.

Main Scenario:

1. The seller selects the "Product" section from the product management interface.
2. The system displays the list of products managed by the seller.
3. The seller chooses a product and selects the "Edit" option.
4. The system opens the Product Editing Form.
5. The system requests the database to retrieve the product's current information.
6. The database returns the product's existing details to the system.
7. The seller edits the product details in the form (e.g., name, description, price, stock).
8. The updated product details are submitted to the database.
9. The system displays a result notification to inform the seller whether the update was successful or not.

Alternative Scenarios:

Invalid Product Details:

- The seller enters invalid details (e.g., negative price, empty name).
- The system detects errors and prompts the seller to correct the information

- **Sequence Diagram: Delete Product**

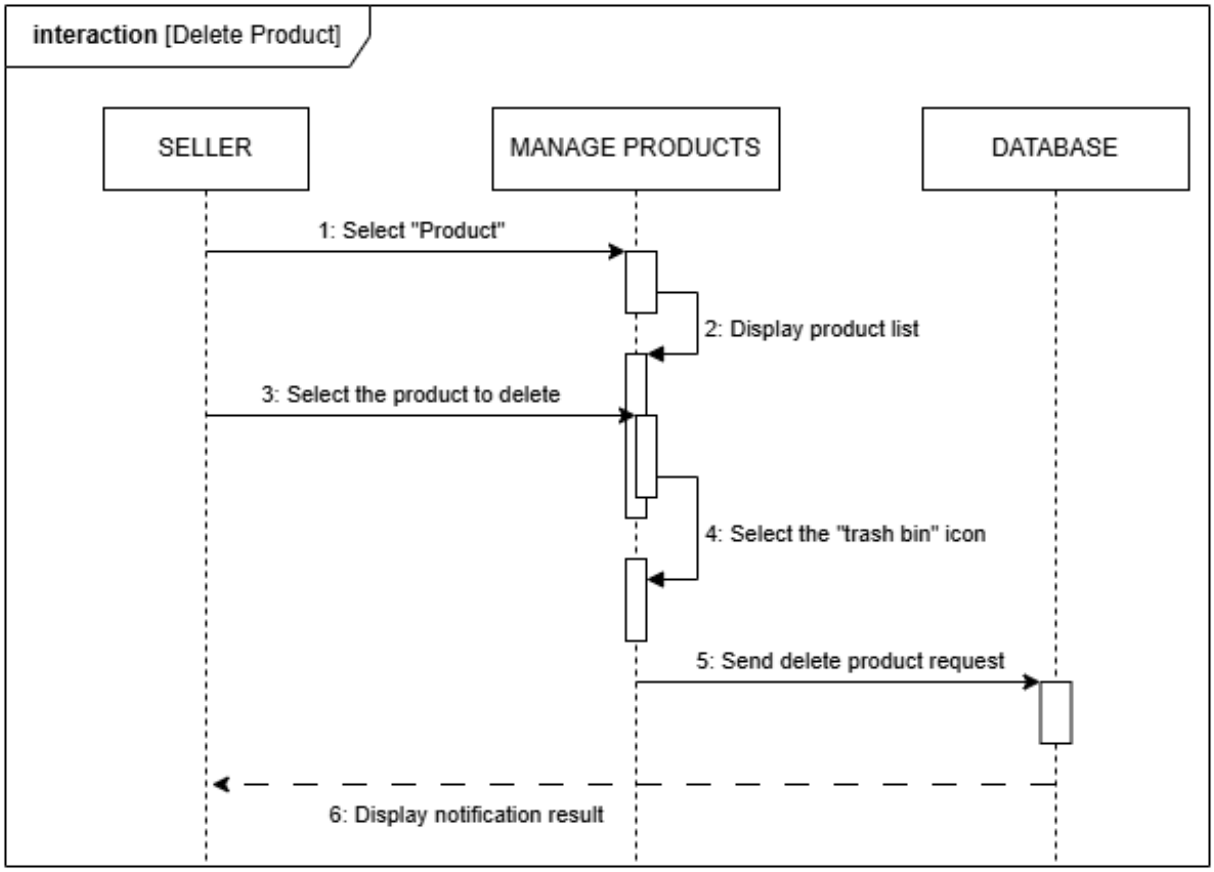

Figure 17: Sequence Diagram Delete Product

Scenario Description: Delete Product

Precondition: The seller must be logged in to access the Manage Products interface.

Main Scenario:

1. The seller navigates to the Product section in the product management interface.
2. The system displays the list of products managed by the seller.
3. The seller selects the product they want to delete.
4. The seller clicks the "Trash Bin" icon to indicate the deletion request.
5. The system sends a delete product request to the database.
6. The system displays a notification result to inform the seller whether the deletion was successful.

- **Sequence Diagram: Sentiment Analysis of Product Reviews**

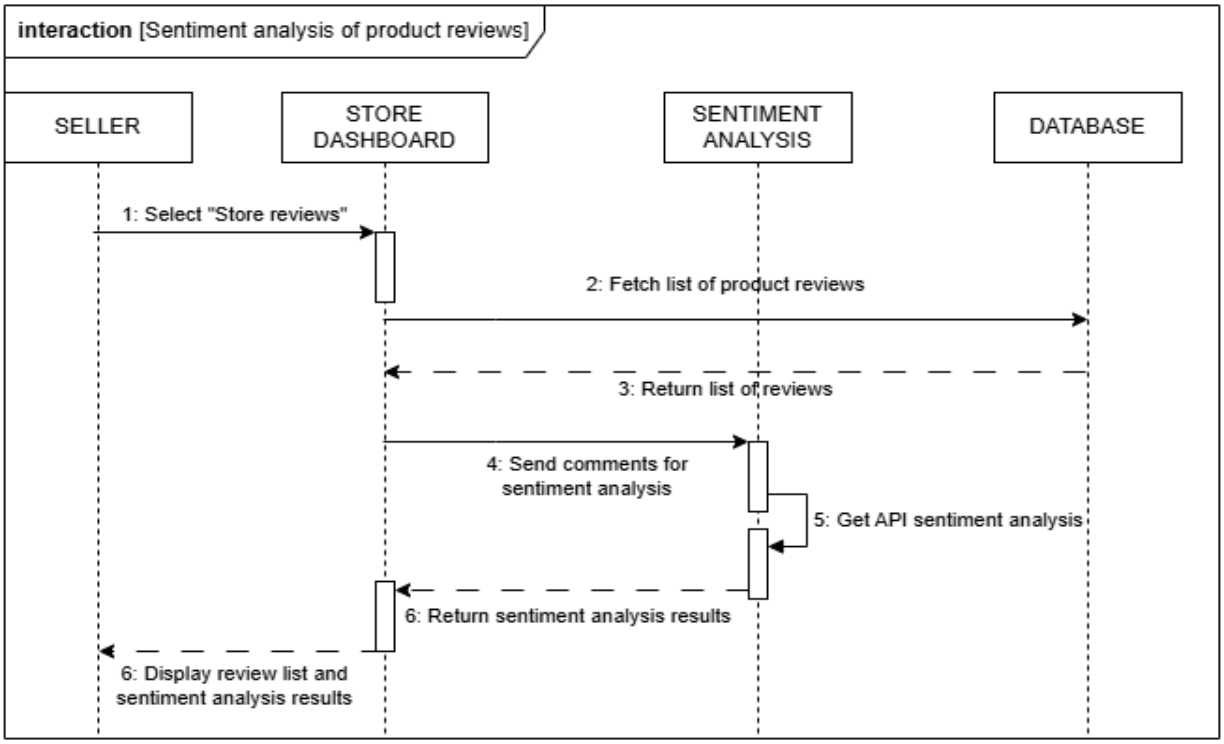

Figure 18: Sequence Diagram Sentiment Analysis of Product Reviews

Scenario Description: Sentiment Analysis of Product Reviews

Precondition: The seller must be logged into the store dashboard.

Main Scenario:

1. The seller selects "Store reviews" from the store dashboard.
2. The system fetches the list of product reviews from the database.
3. The database returns the list of reviews to the store dashboard.
4. The system sends the fetched reviews to the sentiment analysis service for processing.
5. The sentiment analysis service performs an API call to analyze the reviews' sentiment.
6. The sentiment analysis service returns the results to the system.
7. The system displays the review list and the sentiment analysis results to the seller.

Alternative Scenario:

Reviews in Vietnamese:

- If a review is written in Vietnamese, the system detects the language.
- The system skips sending the Vietnamese reviews to the sentiment analysis service.

2.4.2.3 Sequence Diagram for Admin

- **Sequence Diagram: Edit Account Permission**

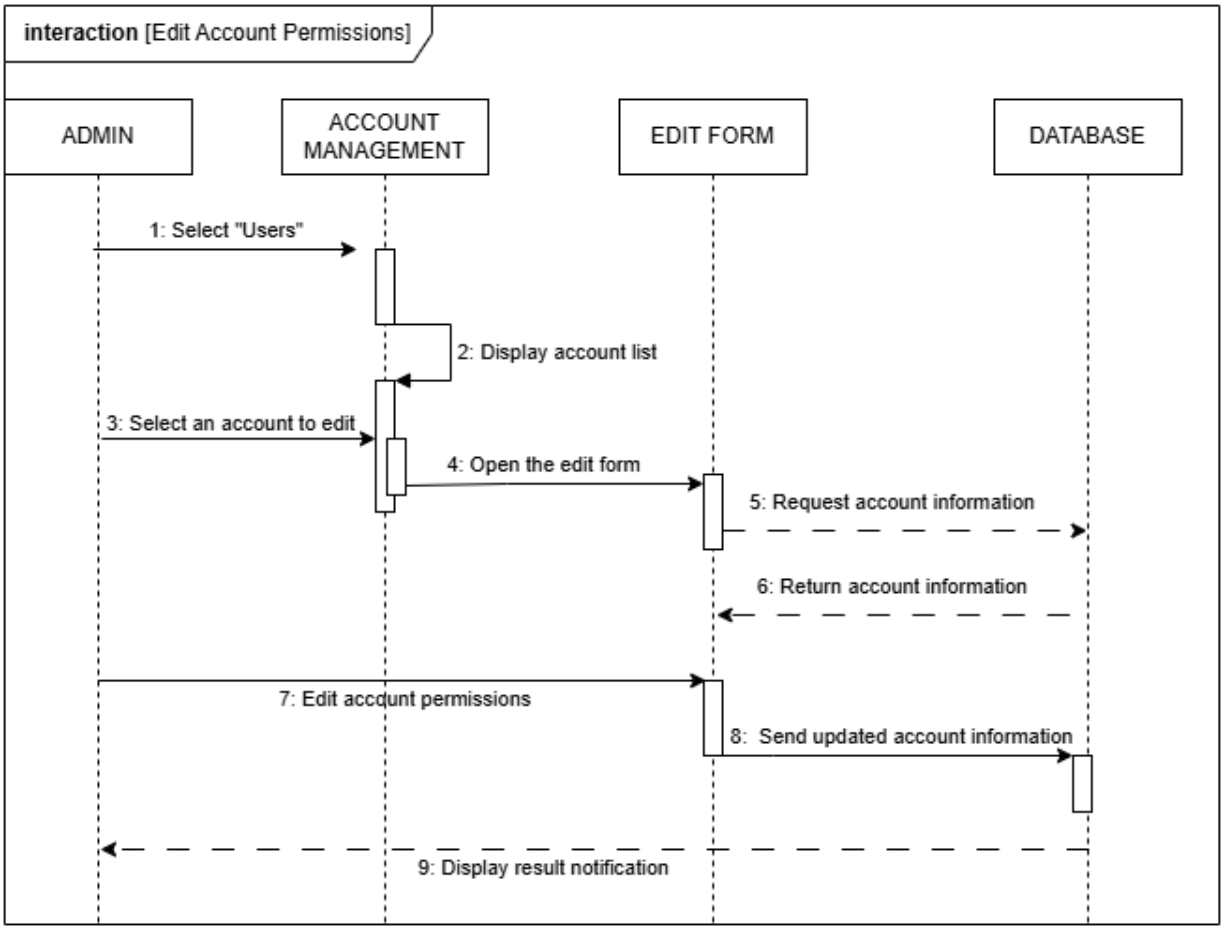

Figure 19: Sequence Diagram Edit Account Permission

Scenario Description: Edit Account Permissions

Precondition: Users must log in to the system using an admin account.

Main Scenario:

1. The admin selects the "Users" option in the account management interface.
2. The system displays a list of user accounts.
3. The admin selects an account to edit.
4. The system opens the account editing interface.
5. The system sends a request to the database to retrieve detailed account information.
6. The database responds with the detailed account information.
7. The admin edits the permissions for the selected account.

8. The system sends the updated account information to the database.
9. The database updates the account successfully.
10. The system alerts the admin about the successful update.

- **Sequence Diagram: Delete account**

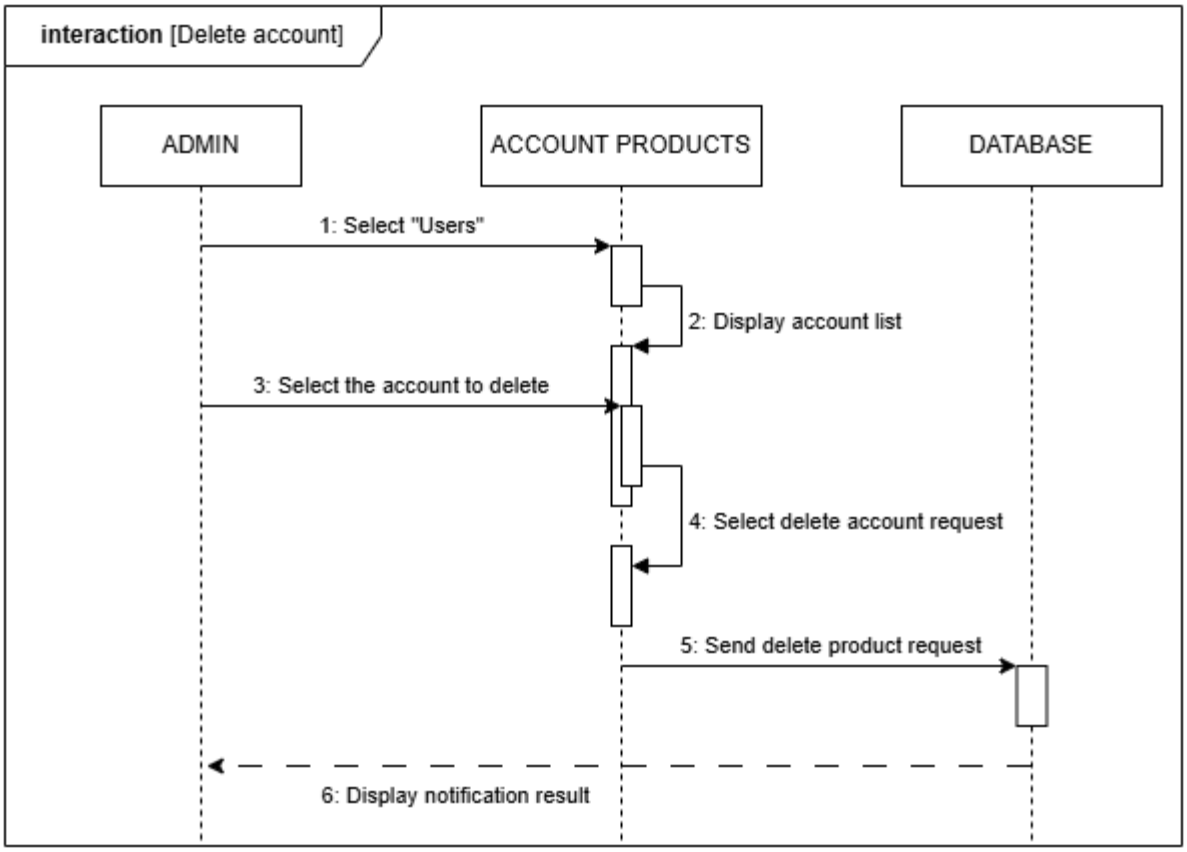

Figure 20: Sequence Diagram Delete account

Scenario Description: Delete Account

Precondition: The user must log in to the system using an Admin account.

Main Scenario:

1. The admin accesses the management function.
2. Select the "Account Management" function.
3. The system displays a list of accounts.
4. The admin selects the account they want to delete and clicks the "Delete Account" button.
5. The system processes the request and returns the result.

2.5 Activity Diagram

2.5.1 Login Activity Diagram

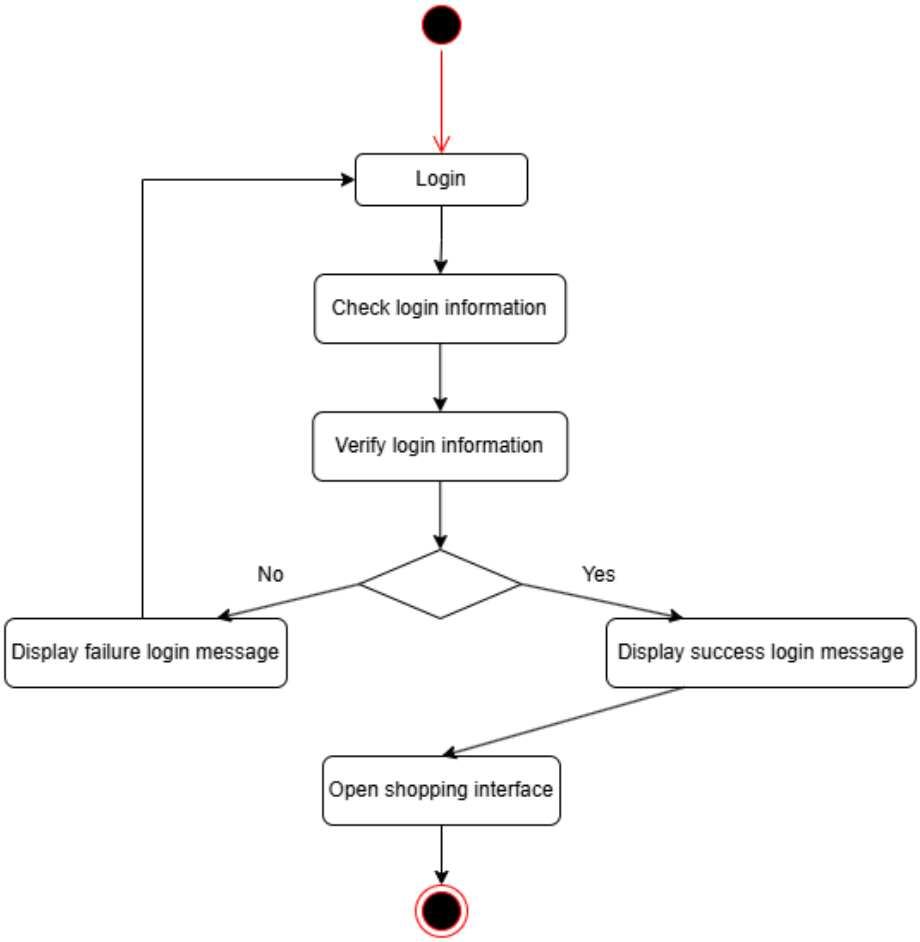

Figure 21: Login Activity Diagram

Summary Description:

- The login process begins.
- The system verifies the login information.
  - Case 1: If the information is correct, the system displays a success message, opens the shopping interface, and ends the process.
  - Case 2: If the information is incorrect, the system displays a failure message and returns to the login process.

2.5.2 Registration Activity Diagram

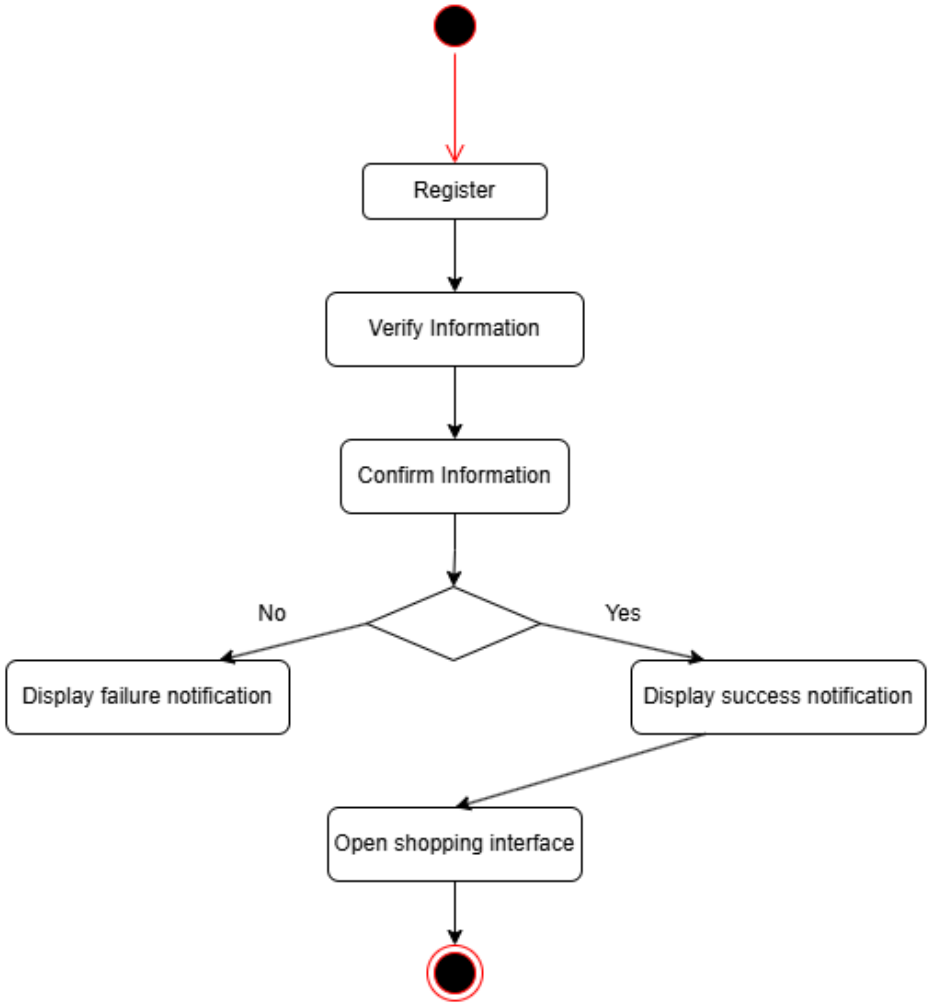

Figure 22: Registration Activity Diagram

Summary Description:

- The registration process begins.
- The system verifies the registration information and confirms it.
- Case 1: If the information is correct, the system displays a success message, opens the shopping interface, and ends the process.
- Case 2: If the information is incorrect, the system displays a failure message and returns to the registration process.

2.5.3 Search Activity Diagram

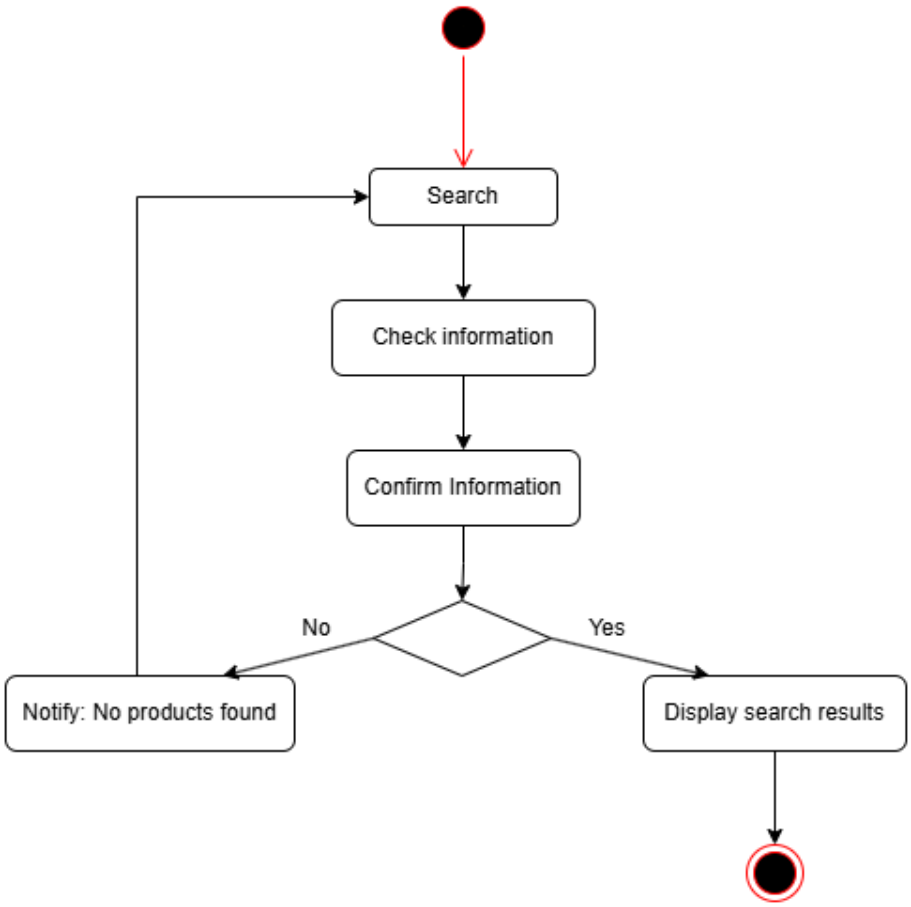

Figure 23: Search Activity Diagram

Summary Description:

- The search process begins.
- The buyer enters the desired keyword.
- The system verifies the information.
- Case 1: If the information is correct, the system displays the search results and ends the process.
- Case 2: If the information is incorrect, the system returns to the search process and prompts the user to re-enter the keyword.

2.5.4 Shopping Cart Activity Diagram

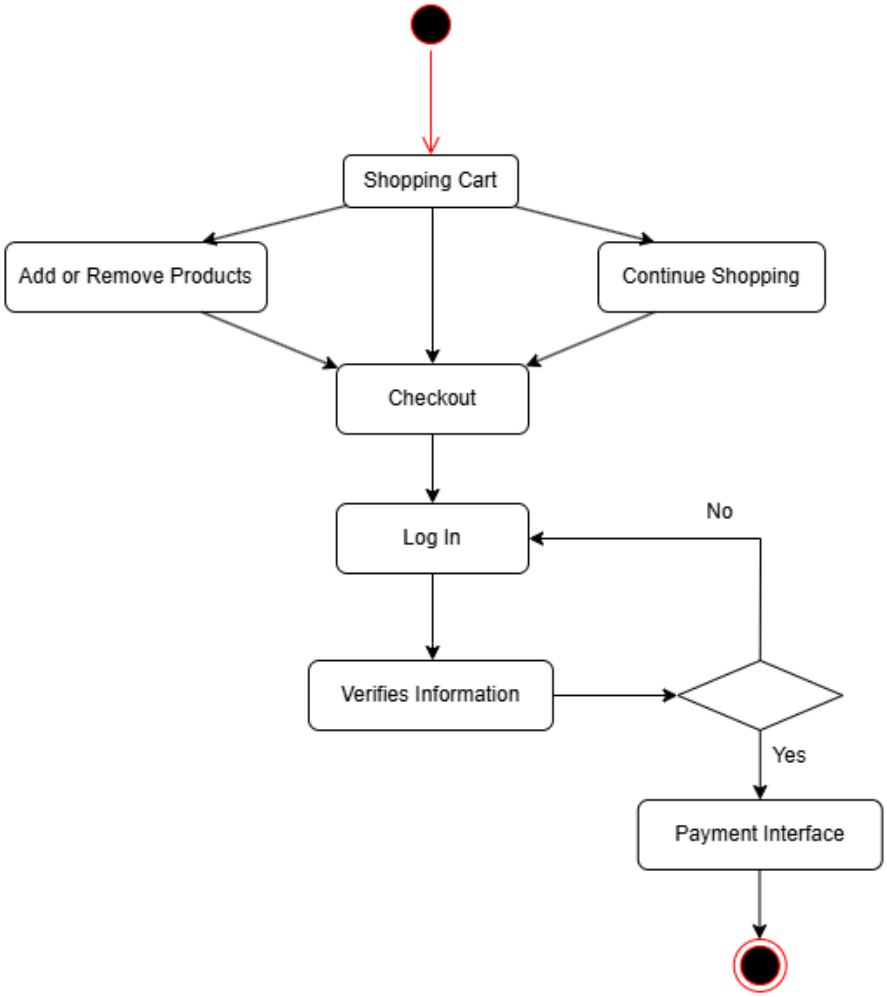

Figure 24: Shopping Cart Activity Diagram

Summary Description:

- The system displays the shopping cart menu, which includes options to add, remove products, continue shopping, and update the cart.
- Next, the system proceeds to the checkout process.
- Case 1: If the user is not logged in, the system will open the login interface.
- Case 2: If the user is already logged in, the system will open the checkout interface.

2.5.5 Payment Activity Diagram

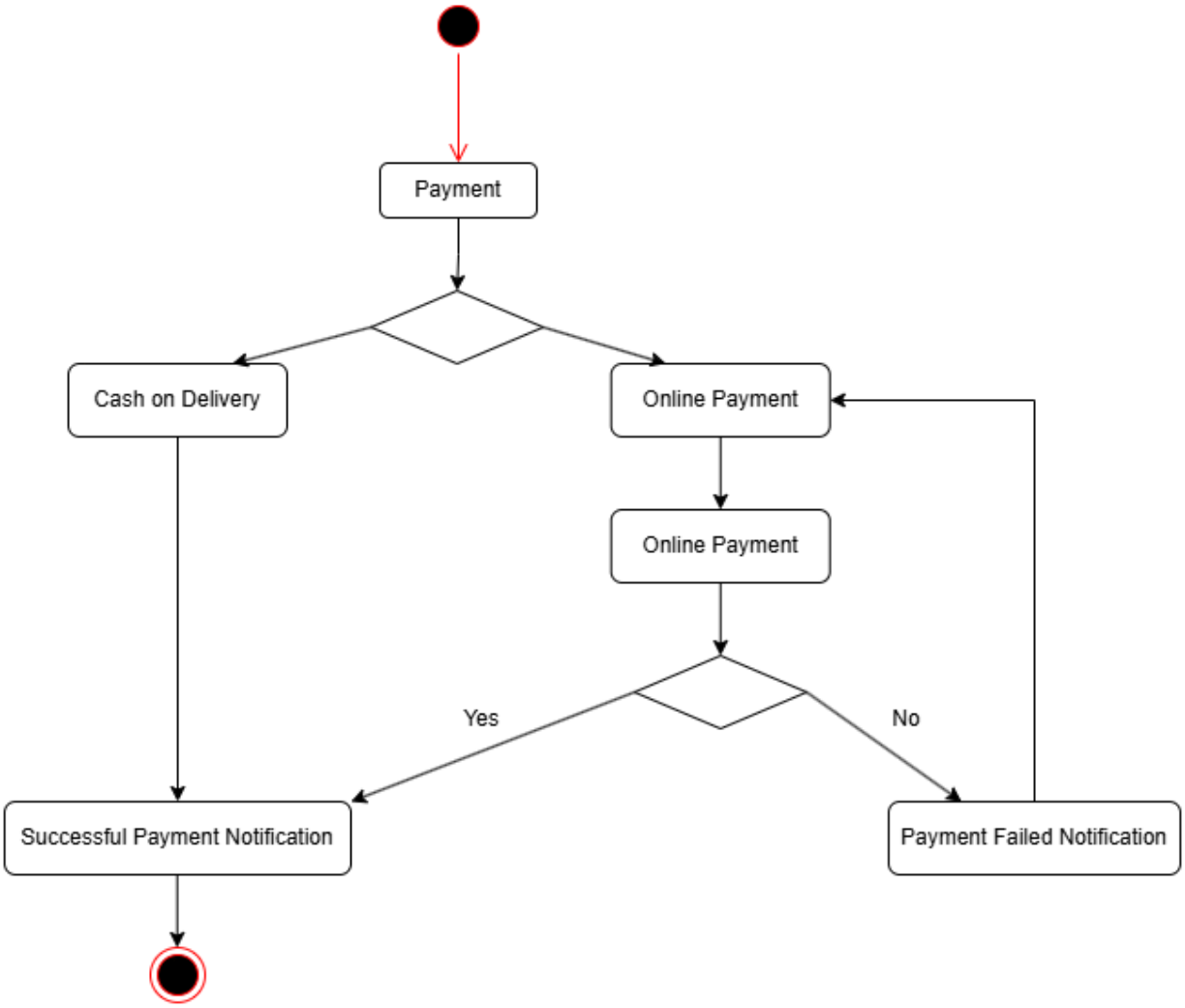

Figure 25: Payment Activity Diagram

Summary Description:

- The payment process begins.
- The system displays two payment options: Cash on Delivery and Online Payment.

- Case 1: If the user selects Cash on Delivery, the system will notify the user that the order has been successfully placed.

- Case 2: If the user selects Online Payment, the system will open an interface for the user to enter credit card information.

  The system will verify the payment card information:

  + If the information is incorrect, the system will reopen the payment interface and notify the user of the error.
  + If the information is correct, the system will notify the user that the order has been successfully placed.

2.6 Designing the Database

A relational database is a type of database in which data is related to other information within the database. It stores and organizes data with references between two or more sources. It also ensures consistency, preventing a single attribute from being stored in a different format in another file. Users only need to modify the data in one table, which will automatically update everywhere.

System databases are arranged according to a specific structure, forming interrelated data fields or records. Users can edit, add, access, and retrieve these data for different purposes.

A database system can overcome all the disadvantages of information systems when stored in file form: It minimizes duplication and increases information sharing. Thanks to its security permission features, it allows control over how many people can access the information.

Below are the database tables:

Table 7: Database tables "conversations"

Column	Data Type	Description	Notes
id	Integer	Conversation ID	
groupTitle	String	Title of the group conversation	
members	Json	List of members participating in the conversation	Content of the most recent message
lastMessage	Text	Content of the most recent message	
lastMessageId	String	ID of the most recent message	
created_at	Date	Timestamp when the conversation was created	
updated_at	Date	Timestamp of the most recent conversation update	

Table 8: Database tables "couponcode"

Column	Data Type	Description	Notes
id	Integer	Couponcode ID	
name	String	Name of the discount code	
value	Decimal(10, 2)	Discount value	
minAmount	Decimal(10, 2)	Minimum order amount required for discount eligibility	
maxAmount	Decimal(10,2)	Maximum discount amount	
shopId	String	ID of the shop that created the discount code	
selectedProducts	Text	List of products eligible for the discount code	Stored as JSON
created_at	Date	Timestamp when the code was created	

Table 9: Database tables "event"

Column	Data Type	Description	Notes
id	Integer	Event ID	
name	String	Event name	
description	Text	Detailed description of the event	
category	String	Event category	

start_Date	Dateonly	Event start date	
Finish_Date	Dateonly	Event end date	
status	String	Event status	Stored as JSON
tags	String	Tags (keywords) for the event	
originalPrice	Decimal(10, 2)	Original price for products within the event (if any)	
discountPrice	Decimal(10, 2)	Discounted price	Required
stock	Integer	Quantity of products allocated to the event	
images	Json	Array of event images	
shopId	String	ID of the shop that created the event	
shop	Json	Shop information in JSON	
sold_out	Integer	Number of items sold	
created_at	Date	Timestamp when the event was created	

Table 10: Database tables "message"

Column	Data Type	Description	Notes
id	Integer	Message ID	
conversationId	String	ID of the conversation in which the message appears	
text	Text	Message content	
sender	String	Sender	user ID or username
images	Text	Attached images in the message	Could be stored as URLs
created_at	Date	Timestamp when the message was created	
updated_at	Date	Timestamp when the message was last updated	

Table 11: Database tables "order"

Column	Data Type	Description	Notes
id	Integer	Order ID	
cart	JSON	Details of the cart	
shipping_address	JSON	Shipping address information	
user	JSON	Information about the person placing the order	
total_price	Decimal(10, 2)	Total amount of the order	
status	String	Order status (Processing, Shipped, Delivered, etc.)	
payment_info	JSON	Payment information	
paid_at	Date	Timestamp when payment was successfully completed	
delivered_at	Date	Timestamp when delivery was successfully completed	
created_at	Date	Timestamp when the message was created	

Table 12: Database tables "product"

Column	Data Type	Description	Notes
id	Integer	Product ID	
name	String	Product name	
description	Text	Detailed description of the product	
category	String	Product category	
tags	String	Quick descriptor tags	
originalPrice	Decimal(10, 2)	Original price of the product	
discountPrice	Decimal(10, 2)	Discounted price	
stock	Integer	Quantity of products in stock	
images	JSON	List of product images	
reviews	JSON	List of product reviews	Could contain an array of objects.
ratings	Decimal (3, 2)	Average rating of the product	
shopId	String	ID of the shop that owns the product	
shop	JSON	Shop information (name, avatar, etc.)	Stored as JSON
sold_out	Integer	Number of products sold	
created_at	Date	Timestamp when the message was created	

Table 13: Database tables "shop"

Column	Data Type	Description	Notes
id	Integer	Shop ID	
name	String	Name of the shop	
email	String	Login email for the shop	
password	String	Password	Hashed password
description	Text	Brief description of the shop	
address	String	Detailed address of the shop	
phoneNumber	Bigint	Shop’s phone number	
role	String	Role (Seller)	
avatar	String	Shop’s avatar image	
zipCode	Integer	Zip code (if any)	
withdrawMethod	JSON	Withdrawal method (PayPal, bank account, etc.)	
availableBalance	Decimal(10, 2)	The shop’s available balance	
created_at	Date	Timestamp when the message was created	

Table 14: Database tables "user"

Column	Data Type	Description	Notes
id	Integer	User ID	
name	String	User’s name	
email	String	Email (used for login)	
password	String	Password	Hashed password
phoneNumber	Bigint	User’s phone number	
addresses	JSON	List of addresses (if there are multiple)	Stores an array of objects (e.g. { city, street, ... }).
role	String	Role (user, admin, etc.)	Defaults to "user"
avatar	String	User’s avatar	Could be a URL
created_at	Date	Timestamp when the user was created	

Table 15: Database tables "withdraw"

Column	Data Type	Description	Notes
id	Integer	Withdrawal request ID	
seller	JSON	Shop (seller) information requesting the withdrawal	May store the shop's ID, bank details, etc., as JSON.
amount	Decimal(10, 2)	Amount to withdraw	
status	String	Processing status	
created_at	Date	Timestamp when the withdrawal request was created	

2.7 ERD Model of the System

The ERD model, short for Entity Relationship Diagram, describes entities and their relationships (entity linkage). ERD diagrams are commonly used to design or troubleshoot relational databases in software engineering, business information systems, education, and research.

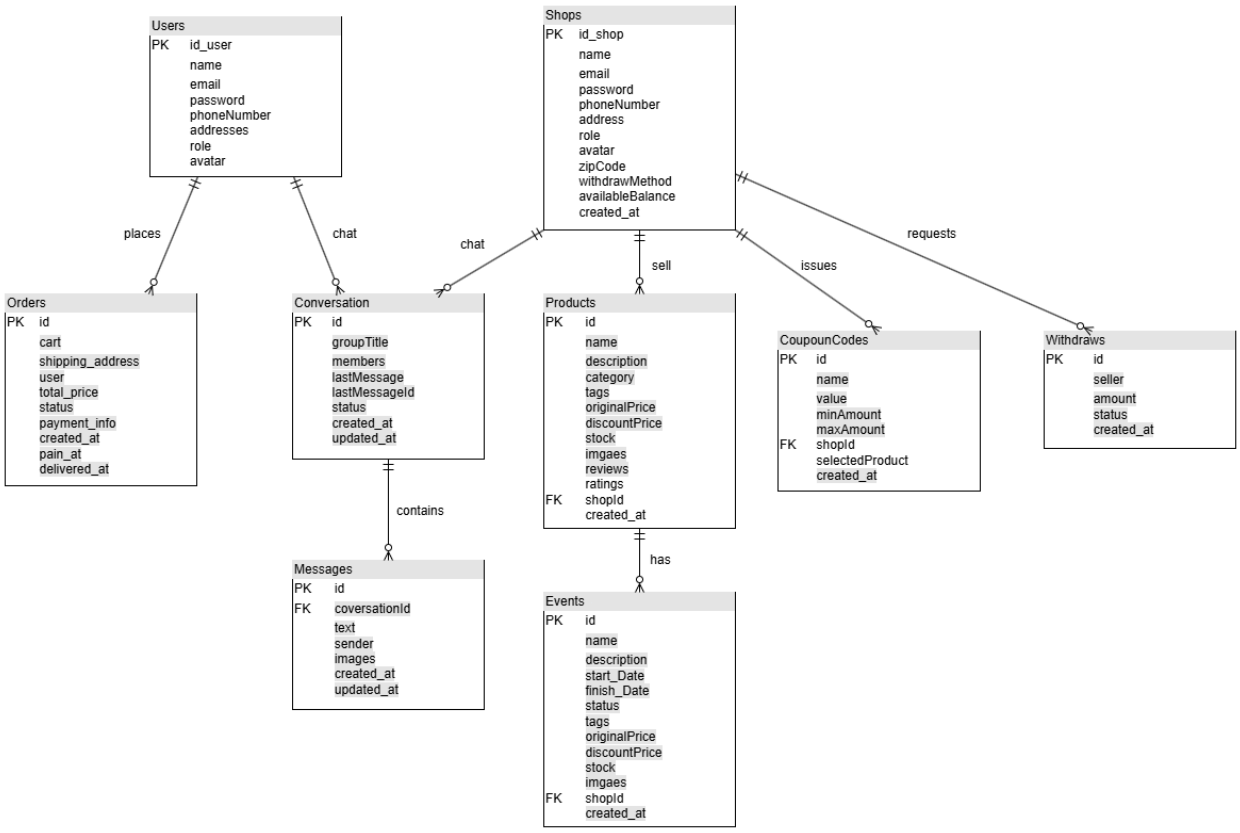

Figure 26: Entity Relationship Diagram (ERD)

CHAPTER 3: TOOLS AND SOFTWARE DEVELOPMENT ENVIRONMENT

3.1 ReactJS

React is a JavaScript library developed by Facebook to build highly interactive, stateful, and reusable UI components. React works both on the client-side and server-side (server-side rendering) and facilitates seamless integration. One of React's core strengths is its ability to compare changes between the current and previous renders to update the DOM (Document Object Model) with minimal modifications, enhancing performance (Goray, 2024).

Key Features in ReactJS

- **Virtual DOM**

  Virtual DOM is a technology designed to improve application performance. It is a JavaScript object that simulates the real DOM structure, containing all the necessary information to build the actual DOM.

  When data changes, React calculates the differences between the Virtual DOM and the real DOM, optimizing updates to minimize re-rendering costs and improve speed.

- **One-way Data Binding**

  React employs a one-way data flow mechanism (one-way data binding). Data flows from the parent component to the child component through props.

  This simplified data flow ensures tighter control and makes debugging easier. This feature is particularly useful in building applications with dynamic user interfaces, such as Facebook's Newsfeed, where statuses, likes, shares, and comments are constantly updated.

- **JSX (JavaScript XML)**

  JSX is an extended syntax that allows you to write HTML directly within JavaScript.

  Faster: Optimized for performance during compilation into JavaScript code.

  Safer: As a statically typed language, JSX detects errors during compilation.

  Easier: It inherits JavaScript features, making it familiar and easy for JavaScript developers to use.

- **Component**

  React is built around components, unlike other frameworks that rely on templates. An application in React is composed of small, reusable components.

  Each component has its state, and React automatically updates the UI when the state changes. Components make it easier to organize code, especially in large-scale projects.

- **Props and State**

  Props: Attributes passed from a parent component to a child component. They are immutable (cannot be changed).

  State: Represents the internal state of a component. When the state changes, React automatically re-renders the UI.

3.2 JavaScript

JavaScript is a widely used programming language for creating interactive websites. It can be embedded into HTML to make websites dynamic and engaging. Additionally, JavaScript can be used for server-side development through platforms like Node.js.

3.3 Node.js

Node.js is a runtime environment for executing JavaScript on the server side, built on Google's V8 JavaScript Engine. Node.js enables:

- Server-side task execution using JavaScript.
- Building scalable and high-performance applications due to its event-driven, non-blocking I/O architecture.
- Access to numerous utility packages via npm (Node Package Manager) for various development needs, from web development to microservices.

3.4 Visual Studio Code

Visual Studio Code is among the most popular source code editors many developers choose. With outstanding features such as speed, lightweight design, cross-platform support, and extensive extension capabilities, Visual Studio Code has increasingly become an essential tool. As an open-

source software, Visual Studio Code is considered a top choice for developers working on various programming projects.

3.5 Postman

Postman is a tool designed for working with APIs, particularly popular for REST APIs. It is one of the most widely used tools for API testing. With Postman, developers can perform REST API requests without writing any code, check whether data has been added to the database, and determine the status of requests (e.g., success with status code 200). Postman also displays data in JSON or HTML format for easier review.

This tool supports all common HTTP methods, such as GET, POST, PUT, PATCH, DELETE, and more. Additionally, Postman allows the storage of the history of executed requests, making it convenient to reuse them in subsequent tests.

3.6 diagrams.net (draw.io)

diagrams.net (draw.io) is an online tool that supports diagramming and modeling. It enables the creation of block diagrams, flowcharts, UML diagrams (Unified Modeling Language), and many other types. diagrams.net plays a significant role in planning, analyzing, and designing systems, an essential factor in any project. diagrams.net maximizes project productivity and quality with its modeling capability, independent of programming languages, platforms, or developers. The tool provides an easy-to-use interface, allowing users to create clear and understandable diagrams. Moreover, diagrams.net supports direct storage on platforms like Google Drive and Dropbox or export in various formats (PNG, SVG, XML, etc.), offering maximum convenience and flexibility.

3.7 GitHub

GitHub is a cloud-based service that allows hosting version control using Git. Git helps with version control, team collaboration, and project management, key aspects of modern software development. GitHub allows developers to easily host, share, and collaborate on projects.

The platform offers an intuitive and user-friendly interface, making it accessible for beginners and experienced developers. With this tool's integration with CI/CD pipelines such as GitHub Actions, you can also automate testing, building, and deployment within your software delivery.

GitHub works well with tools like Visual Studio Code, where users can track, commit, and push code changes directly in the editor. This integration simplifies code management and helps developers work more efficiently. With these features, GitHub makes managing projects easier and more accessible.

3.8 XAMPP

XAMPP is an easy-to-install Apache distribution that contains MariaDB, PHP, and Perl. With its preconfigured Apache, MySQL, PHP, and phpMyAdmin support, users can develop, test, and ebug almost any website or application offline. XAMPP provides a local server hosting environment, enabling users to run website demos directly on their computers without purchasing hosting or a VPS.

For developers working with Node.js, XAMPP can complement projects by providing a local MySQL database for backend storage. Node.js applications can connect to the MySQL server in XAMPP using libraries like MySQL or Sequelize, enabling efficient database operations. This integration allows developers to seamlessly build, test, and deploy database-driven Node.js applications in a local development environment (Thanh, n.d.).

3.9 MySQL

MySQL is a widely used and highly regarded Relational Database Management System (RDBMS) in the software development community. Initially developed by MySQL AB and later acquired by Oracle Corporation, MySQL is a popular choice for various applications, ranging from small websites to large-scale systems requiring stability and performance (RDBMS, n.d.).

Key Features of MySQL

- **Structured Relational Framework**

Data is organized in tables with rows and columns, and relationships between tables are explicitly defined, ensuring data integrity and consistency.

- **ACID Compliance**

MySQL supports ACID (Atomicity, Consistency, Isolation, Durability) standards, ensuring data reliability and consistency, particularly for critical transactions.

- **High Performance and Flexibility**

With features like query optimization, effective indexing, and multiple storage engines (e.g., InnoDB, MyISAM, Memory), MySQL performs well even in high-load environments.

- **Replication and Clustering**

Supports replication to copy data from a controller server to multiple agent servers, enhancing fault tolerance and ensuring data availability. Additionally, clustering (e.g., MySQL Cluster) meets scalability demands for larger systems.

- **Robust Security and Role Management**

Provides user authentication mechanisms, role-based access control, and permission management to safeguard data.

- **Integration and Extensibility**

MySQL works seamlessly across platforms (Windows, Linux, macOS) and supports multiple programming languages (PHP, Python, Java, C++, etc.), making it suitable for diverse applications.

3.10 Sequelize

Sequelize is a powerful ORM (Object Relational Mapping) that simplifies working with relational databases like MySQL in Node.js applications. Using Sequelize, developers can interact with databases through friendly JavaScript methods instead of writing raw SQL queries, speeding up development and reducing errors when managing databases (habtesoft, 2024).

Key Features of Sequelize

- **Support for Multiple Databases**

Sequelize is compatible with multiple database systems, including MySQL, PostgreSQL, SQLite, and MSSQL, offering flexibility in project selection.

- **Database Schema Management**

Sequelize provides tools like migrations to modify database schemas easily and in a controlled manner, ensuring consistency across development, testing, and production environments.

- **Data Relationship Modeling**

Sequelize supports defining relationships in databases, such as One-to-One, One-to-Many, and Many-to-Many, allowing for effective data management and querying.

- **Powerful Querying Capabilities**

Sequelize offers intuitive APIs for performing operations like SELECT, INSERT, UPDATE, and DELETE, with flexible and developer-friendly query building.

- **Seamless Integration with Node.js**

Sequelize integrates smoothly with popular frameworks like Express.js and ReactJS, enabling rapid and efficient backend application development.

CHAPTER 4: SYSTEM IMPLEMENTATION AND TESTING

4.1 System Guides / Manual

- **Frontend (React)**

- All user interface (UI) and interaction elements are built with React. Inside the **frontend/** directory, the source code is divided into components and pages serving core features such as product display, cart management, and login forms.
- Users interact via a browser, sending requests to the server or navigating internally within the React application (if React Router is used).
- A state management system (Redux, Context API, or Hooks) helps synchronize data across components, reducing errors and increasing reusability.
- When data is required, the frontend sends requests to the backend API (using fetch or axios), receives JSON responses, and updates the UI accordingly.

- **Backend (Node.js + Express)**

- Handles logic, API routing, and database connections. The source code is located in a **backend/** directory.
- Express defines routes such as /api/products, /api/users, etc., which the frontend can access via HTTP methods (GET, POST, PUT, DELETE, ...).
- A controller manages the logic at each endpoint: validating data, interacting with the Sequelize model to read/write data, and returning JSON responses.
- The system can also integrate additional middleware for authentication, error handling, or file operations (if needed).

- **Database (MySQL + Sequelize)**

- Stores relational data in tables (Users, Products, Orders, etc.). With Sequelize as the ORM, developers interact with the database using JavaScript models and methods instead of writing raw SQL queries.
- Connection settings (host, user, password, database) are placed in a .env file or a dedicated configuration file.

- The **models/** directory contains each model (User, Product, Order,...) with fields corresponding to columns in the database tables. Relationships (one-to-many, many-to-many) between tables are also defined here.

4.2 Installation Manual

4.2.1 Initial Requirements

4.2.1.1 Node.js

- Download the latest version from https://nodejs.org.
- Verify the installation by running:

```
node -v
```

4.2.1.2 MySQL

- Install MySQL separately or use XAMPP (includes MySQL/MariaDB).
- If using XAMPP, enable the Apache and MySQL modules, then access phpMyAdmin at http://localhost/phpmyadmin.

4.2.1.3 Git

- Clone or download the project source code from GitHub.

4.2.2 Project Setup

4.2.2.1 Clone or Download the Project

```
git clone https://github.com/vophuthinh/Ecommerce-app.git
cd Ecommerce-app
```

4.2.2.2 Install Dependencies

- Back-end:

```
cd backend
npm install
```

+ This command automatically installs Node.js packages such as **Express**, **Sequelize**, **Tailwind CSS**..

- Front-end:

```
cd frontend
npm install
```

+ Installs React and other front-end libraries.

- Socket.io

```
cd socket
npm install
```

+ Install socket packages

4.2.2.3 Create and Configure the Database

+ Using phpMyAdmin, run:

```
CREATE DATABASE new-nodejs;
```

+ Update the connection details in the project's environment file (config/.env):

```
DB_HOST = localhost
DB_USER = root
DB_PASS =
DB_NAME = new-nodejs
```

4.2.3 Running the Application

- Back-end

```
cd backend
npm start
```

- Front-end:

```
cd frontend
npm start
```

- Socket.io

```
cd socket
npm start
```

4.3 Testing Plan and Test Output

4.3.1 Test Case 1: Creating a User Account

Function

Allows a new user to register (create an account) in order to use the system.

Objective

Ensure that users can successfully create an account when providing all required information.

Ensure that the system returns an error if the registration information is invalid or duplicated (email, phone number, etc.).

Test Information

Entity: User (buyer) — a new user who does not yet have an account.

Prerequisites:

- The user does not already have an account in the system (their email or phone number does not exist).

Test cases are as follows:

- Valid Input

Table 16: Test Case Successful Registration

Test Case	Input	Expected Result	Pass/Fail
Successful Registration	**name**: "Võ Phú Thịnh" **email**: "vophuthinhnc@gmail.com" **password**: "Vphuthinh332003@"	System sends an email to "vophuthinhnc@gmail.com" with an activation link.	Pass

Actual Result

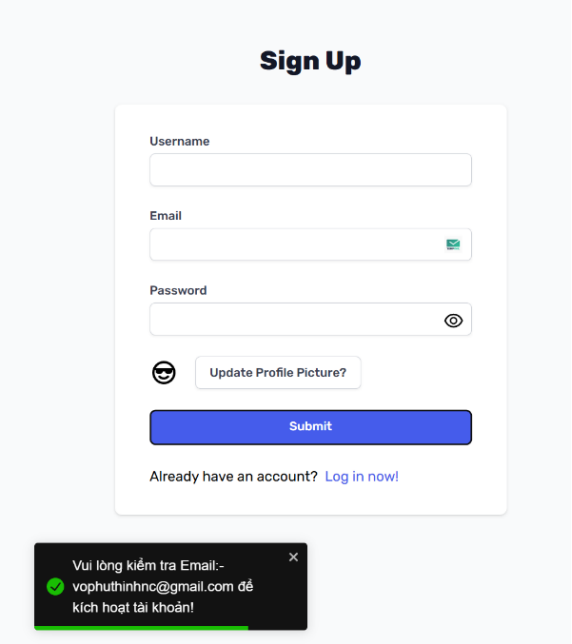

Figure 27: Screenshot of Successful Registration

Figure 28: Screenshot of Account Verification via Gmail

- Invalid Input

Table 17: Test Case Invalid Email

Test Case	Input	Expected Result	Pass/Fail
Invalid Email Format	**name**: "Võ Phú Thịnh" **email**: "vophuthinhnc[at]gmail.com" **password**: "Vphuthinh332003@"	Displays message: “Vui lòng bao gồm ‘@’ trong địa chỉ email”	Pass

Missing '@' Symbol	**name**: "Võ Phú Thịnh" **email**: "vophuthinhncgmail.com" **password**: "Vphuthinh332003@"	Displays message: “Vui lòng bao gồm ‘@’ trong địa chỉ email”	Pass

Actual Result

** Note: Check the default email of the form.*

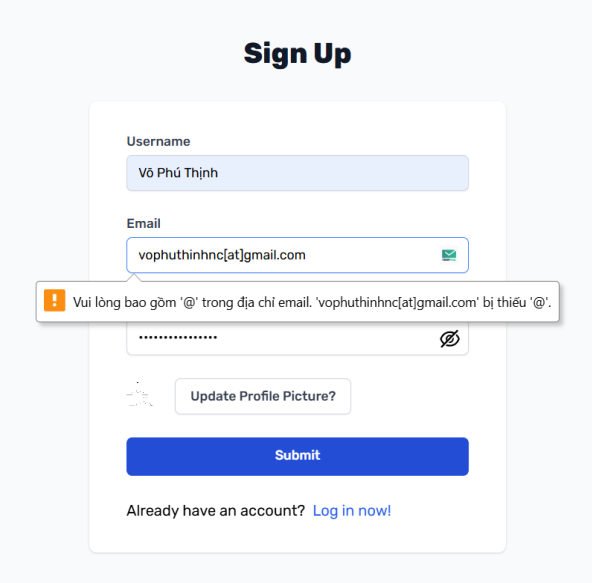

Figure 29: Screenshot of Invalid Email Format

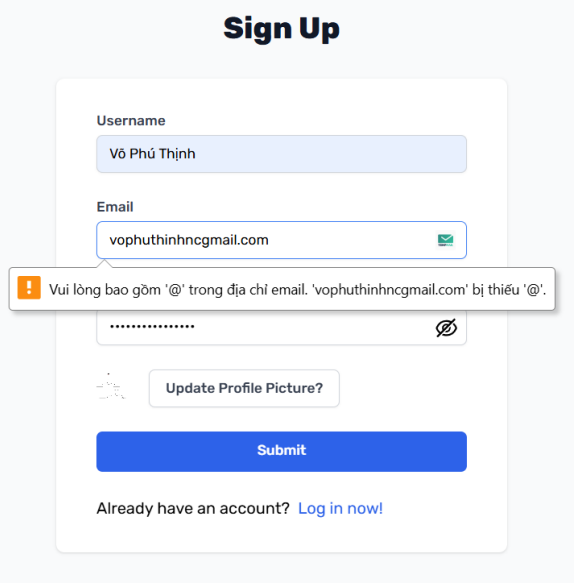

Figure 30: Screenshot of Missing '@' Symbol

Table 18: Test Case Invalid Password

Test Case	Input	Expected Result	Pass/Fail
Password Too Short	**name**: "Võ Phú Thịnh" **email**: "vophuthinhnc@gmail.com" **password**: "Vphu"	Displays message: “Password must be at least 8 characters long.”	Pass

Password Missing Special Character	**name**: "Võ Phú Thịnh" **email**: "vophuthinhnc@gmail.com" **password**: "Vphuthinh332003"	Displays message: "Password must include at least one special character."	Pass

Actual Result

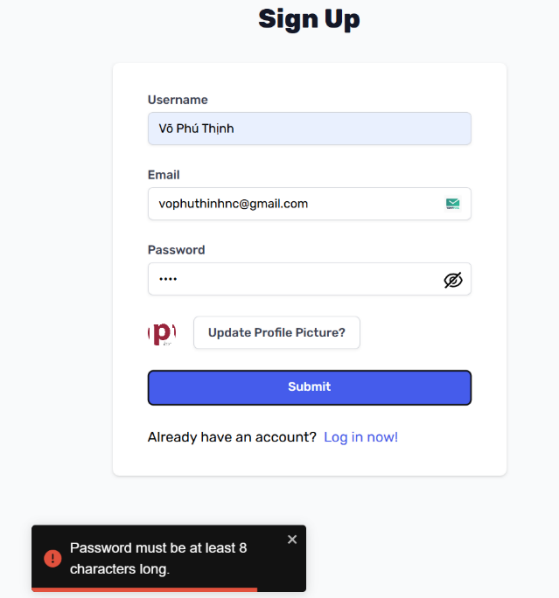

Figure 31: Screenshot of Password Too Short

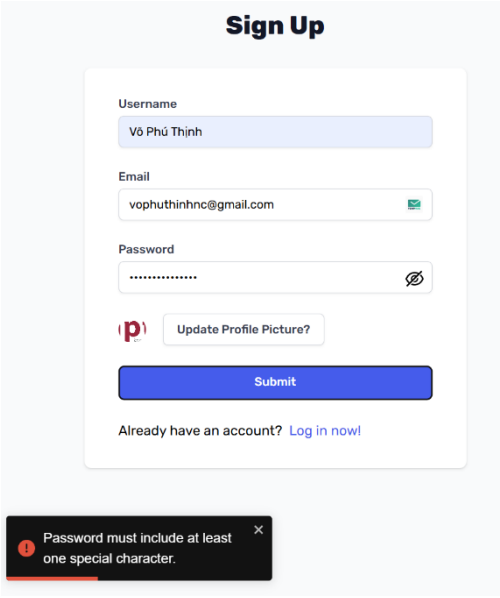

Figure 32: Screenshot of Password Missing Special Character

Table 19: Test Case Duplicate Email

Test Case	Input	Expected Result	Pass/Fail
Email Already Exists	**name**: "Võ Phú Thịnh" **email**: "vophuthinhcm@gmail.com" **password**: "Vphuthinh332003@"	Displays message: “User have existed”.	Pass

Actual Result

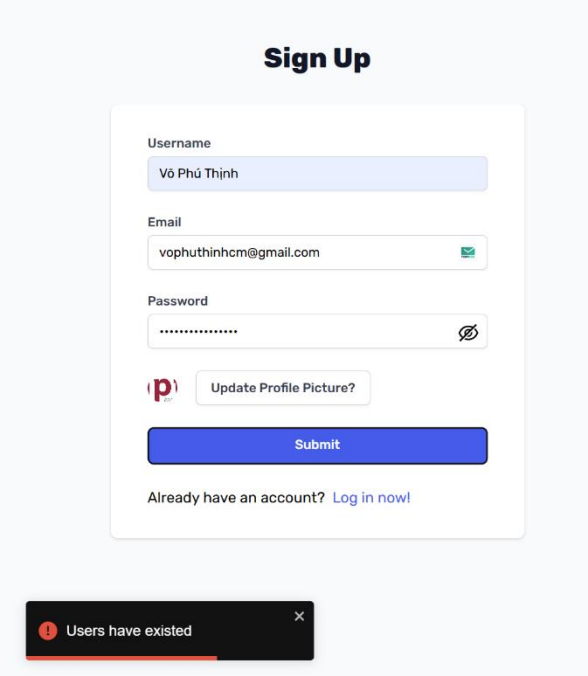

Figure 33: Screenshot of Email Already Exists

Table 20: Test Case Missing Required Fields

Test Case	Input	Expected Result	Pass/Fail
Email Not Entered	**name**: "Võ Phú Thịnh" **email**: "" **password**: "Vphu"	Displays message: “Vui lòng điền vào trường này”	Pass
Password Not Entered	**name**: "Võ Phú Thịnh" **email**: "vophuthinhnc@gmail.com" **password**: ""	Displays message: “Vui lòng điền vào trường này”	Pass

Actual Result

* *Note: Check the default email of the form.*

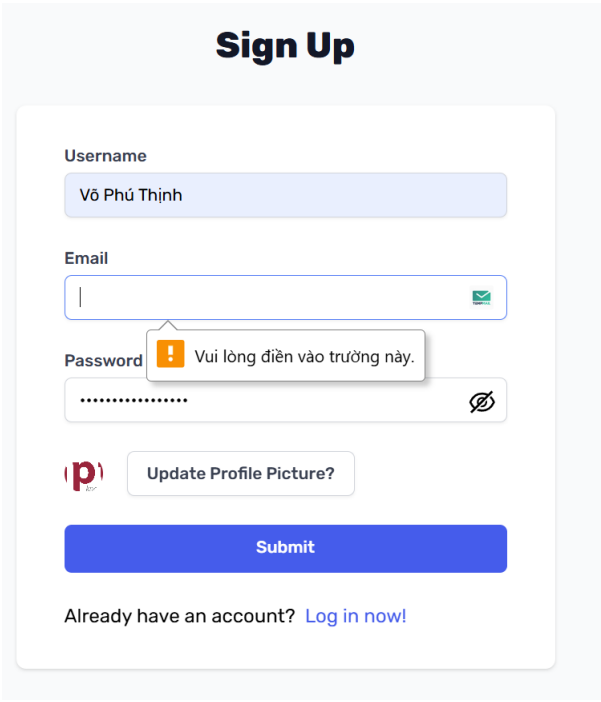

Figure 34: Screenshot of Email Not Entered

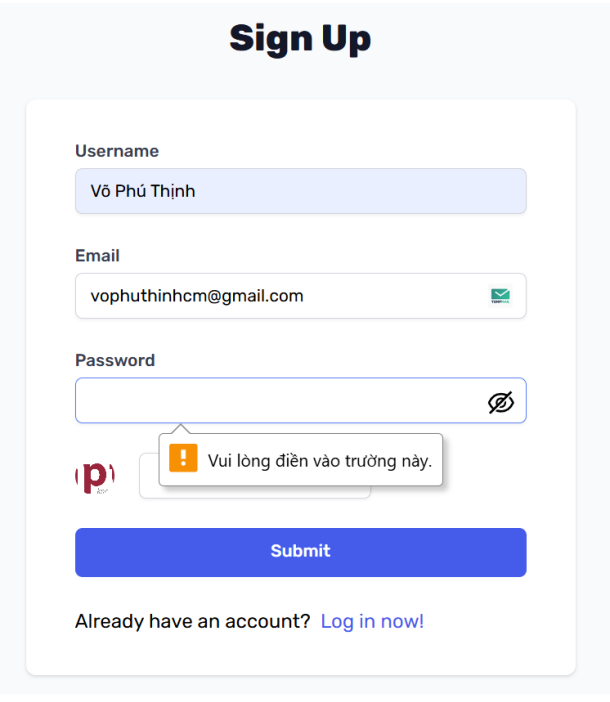

Figure 35: Screenshot of Password Not Entered

4.3.2 Test Cases for Shopping Cart Functionality

Function

Allows users to manage their shopping cart to add, view, update, or remove items and proceed with purchases.

Objective

Ensure users can successfully add products to the cart with accurate details (name, image, price).

Ensure the cart updates correctly when quantities or products are modified.

Validate the system handles edge cases like empty carts, out-of-stock products, and persistence across sessions.

Test Information

Entity: User (buyer) — a logged-in or guest user managing their shopping cart.

Prerequisites:

The user is logged in or can access the cart functionality as a guest.

Products to be added to the cart are available and in stock.

Test cases are as follows:

Table 21: Test Case Add Product to Cart

Test Case	**Input**	**Expected Result**	**Pass/Fail**
Add product to cart	Select a product and click "Add to Cart".	The cart is updated with the product's **name**, **image**, and **price** displayed correctly.	Pass

Actual Result

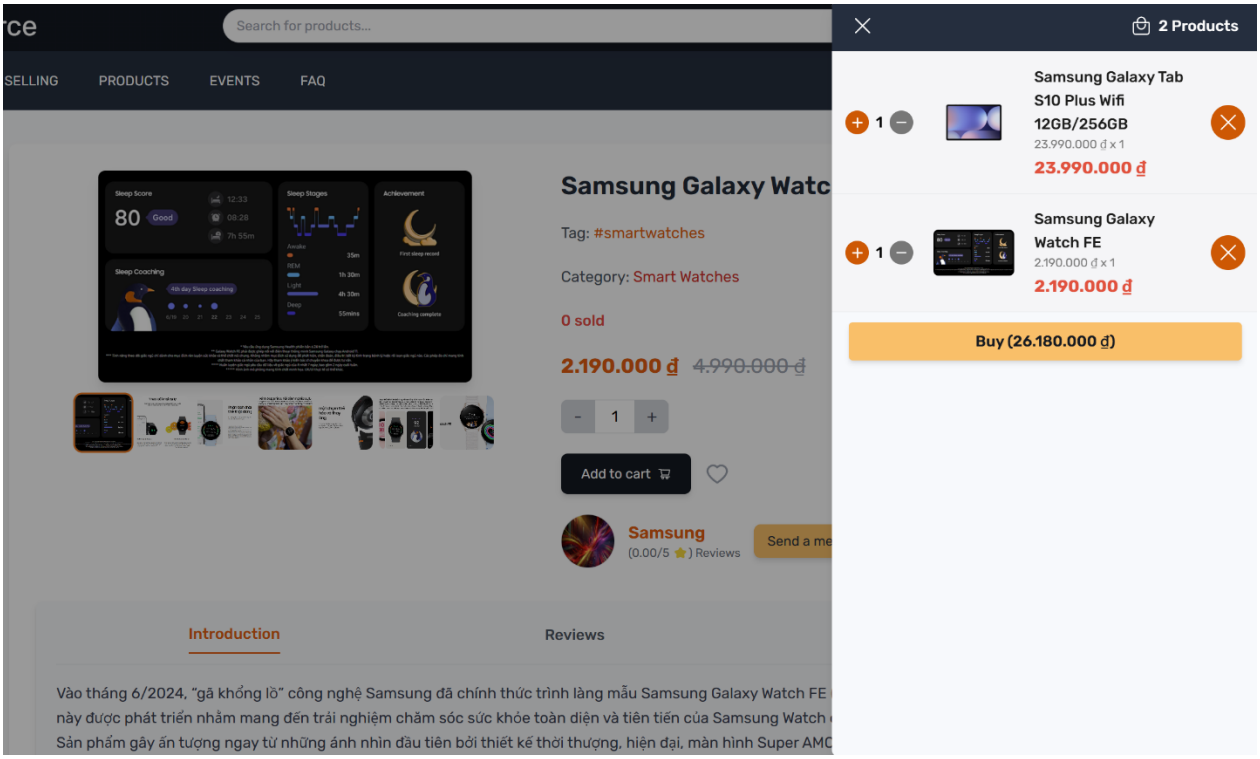

Figure 36: Screenshot of Add Product To Cart

Table 22: Test Case Manage Product Quantities in Cart

Test Case	Input	Expected Result	Pass/Fail
Add the same product multiple times	Add the same product to the cart multiple times.	Displays message: "The product is already in the cart!"	Pass

Actual Result

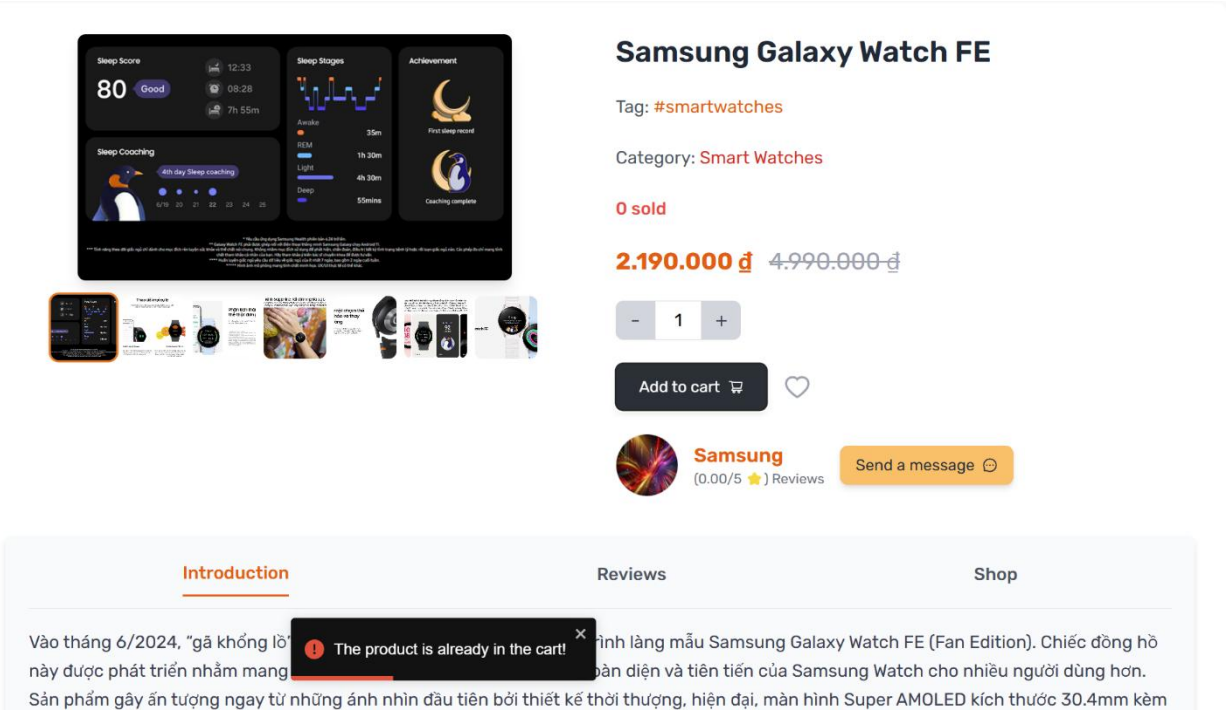

Figure 37: Screenshot of Add The Same Product Multiple Times

Table 23: Test Case Persistent Cart State

Test Case	Input	Expected Result	Pass/Fail
Retain cart state after browser restart	Add one or more products to the cart, then close and reopen the browser.	The cart retains the previously added products and displays them upon reopening the browser.	Pass

Actual Result

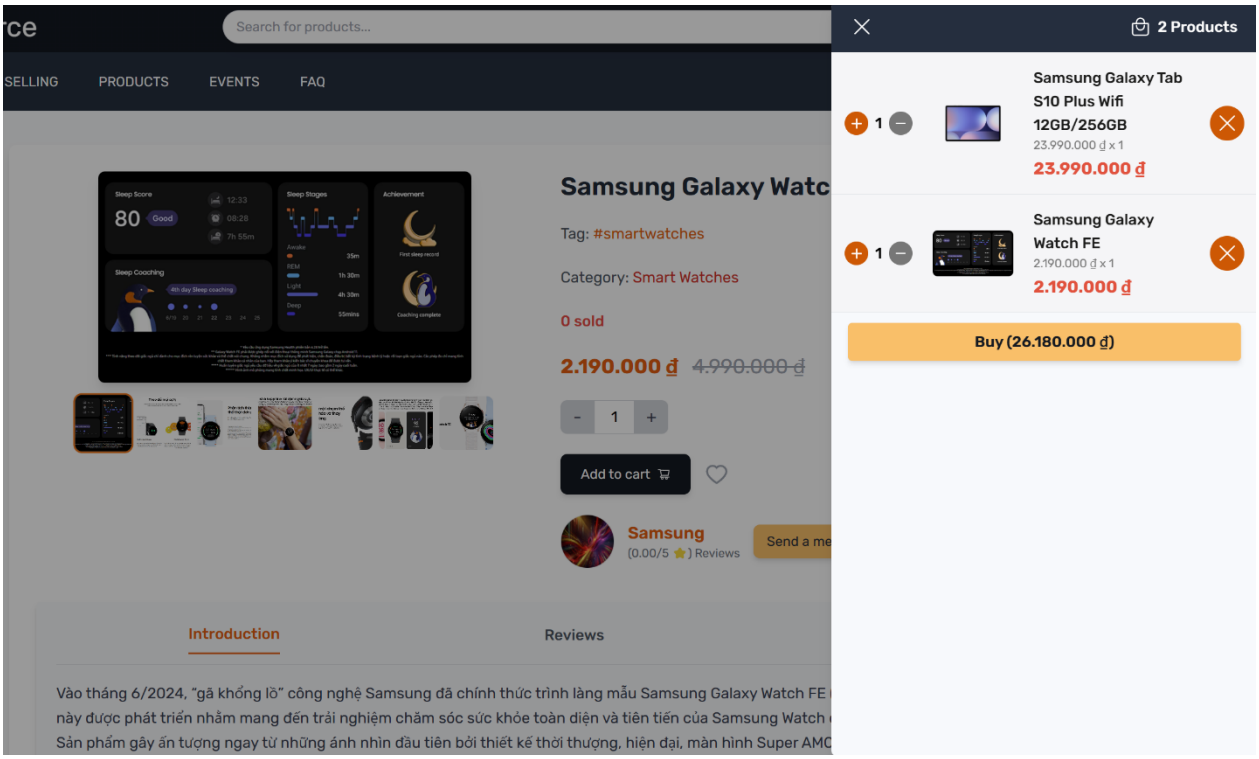

Figure 38: Screenshot of Retain Cart State After Browser Restart

4.3.3 Test Cases for Sentiment Analysis for Seller

Function

Allows sellers to analyze the sentiment of customer reviews to gauge customer satisfaction and make data-driven improvements.

Objective

Ensure the system correctly categorizes customer reviews as positive, negative, or neutral.

Verify the accuracy of sentiment scores returned by the system.

Validate the system handles edge cases such as empty reviews, unsupported languages, or ambiguous text.

Test Information

Entity: Seller — a user with seller privileges analyzing customer reviews.

Prerequisites:

The seller has access to their store's customer reviews.

Reviews are text-based and stored in a supported language for sentiment analysis.

Test cases are as follows:

Table 24: Test Case Analyze Positive Review

Test Case	Input	Expected Result	Pass/Fail
Analyze Positive Review	Review text: "This product is amazing! I love it!"	Sentiment: Positive	Pass

Actual Result

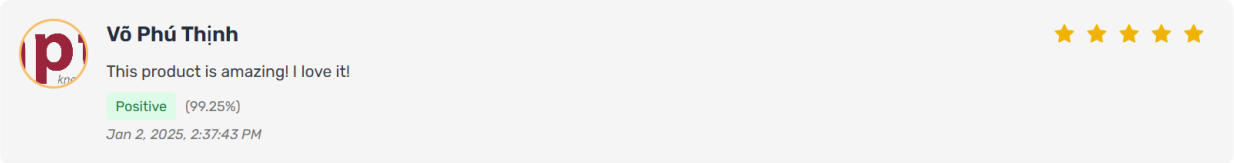

Figure 39: Screenshot of Analyze Positive Review

Table 25: Test Case Analyze Negative Review

Test Case	Input	Expected Result	Pass/Fail
Analyze Negative Review	Review text: "Terrible quality, not worth the price."	Sentiment: Negative	Pass

Actual Result

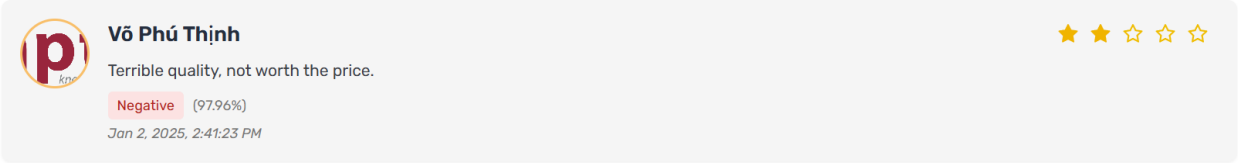

Figure 40: Screenshot of Analyze Negative Review

Table 26: Test Case Unsupported Language

Test Case	Input	Expected Result	Pass/Fail
Unsupported Language	Review text: "Tôi thích sản phẩm này"	Sentiment: Positive	Fail

Actual Result

** Note: The sentiment analysis result is incorrect due to an unsupported language.*

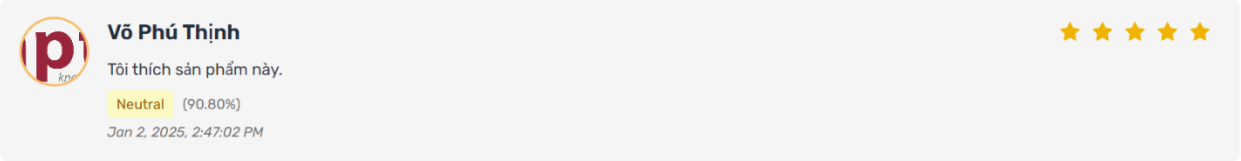

Figure 41: Screenshot of Unsupported Language

4.4 Main Function Codes

```javascript
// sentimentAnalysis.js
const axios = require('axios');

const API_TOKEN = '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';
const API_URL = 'https://api-inference.huggingface.co/models/finiteautomata/bertweet-base-sentiment-analysis';

// Checking model status
async function checkModelStatus() {
    try {
        const headers = {
            Authorization: `Bearer ${API_TOKEN}`,
        };
        const response = await axios.get(API_URL, { headers });
        if (response.data.loading) {
            console.log('Model is loading, please wait...');
            return false;
        }
        return true;
    } catch (error) {
        console.error('Unable to check model status:', error.message);
        return false;
    }
}

// Sentiment analysis for a review
async function analyzeSentiment(review) {
    try {
        const headers = {
            Authorization: `Bearer ${API_TOKEN}`,
        };
        const response = await axios.post(API_URL, { inputs: review }, { headers });
        return response.data;
    } catch (error) {
        console.error('Sentiment analysis error:', error.response ? error.response.data : error.message);
        return null;
    }
}

// Retrieving primary sentiment label
function getDominantLabel(result) {
    const labels = result[0];
    const dominant = labels.reduce((prev, curr) => (curr.score > prev.score ? curr : prev));
    return { label: dominant.label, score: dominant.score };
}

module.exports = {
    checkModelStatus,
    analyzeSentiment,
    getDominantLabel,
};

```

Figure 42: Screenshot of Code for Sentiment Analysis Processing

CHAPTER 5: CONCLUSION

5.1 Summary of Main Findings

The project successfully implemented an e-commerce platform integrated with sentiment analysis to enhance customer experience and provide businesses with actionable insights. Key achievements include

- User-friendly Features: Smooth navigation, product search, and customer care management (chat).
- Advanced Seller Tools: Management of products, orders, and promotions.
- Admin Controls: Oversight of system activities, users, and transactions.
- Sentiment Analysis: Automated analysis of customer reviews, aiding decision-making and improving service quality.
- Scalable System Design: Built with ReactJS, Node.js, and MySQL, ensuring performance, security, and extensibility.

5.2 Discussion and Implications

The integration of sentiment analysis highlights the potential for leveraging customer feedback in decision-making processes. This innovation enables businesses to:

- Personalize marketing strategies.
- Enhance user satisfaction through targeted improvements.
- Gain a competitive edge by understanding customer needs.
- The modular architecture ensures the system is scalable and adaptable for future enhancements, such as multilingual support or AI-driven product recommendations.

5.3 Limitations of the System

- Language Dependency: Sentiment analysis is primarily effective for English comments; other languages require additional training datasets.
- Lack of Advanced Features: Features like behavioral analytics and dynamic product recommendations were excluded due to time and resource constraints.
- Data Accuracy: Sentiment analysis accuracy relies on high-quality datasets, which may not fully capture nuances like sarcasm or slang.

5.4 Future Development

- Multilingual Support: Extend sentiment analysis capabilities to include non-English languages (Vietnamese).
- AI-Driven Recommendations: Implement machine learning algorithms for personalized shopping experiences.
- Enhanced User Insights: Develop tools capable of performing aspect-based sentiment analysis to provide deeper insights into specific customer feedback dimensions and trends.

REFERENCES

1. Goray, S. (2024, October 28). *Top Features and Benefits of Using React JS for Web Development*. Retrieved from webandcrafts: https://webandcrafts.com/blog/react-js-features
2. habtesoft. (2024, Sep 4). *Getting Started with Sequelize in Node.js: A Step-by-Step Tutorial*. Retrieved from habtesoft: https://habtesoft.medium.com/getting-started-with-sequelize-in-node-js-a-step-by-step-tutorial-35276b07d7b2#:~:text=Sequelize%20is%20a%20powerful%20ORM,SQLite%2C%20and%20Microsoft%20SQL%20Server.
3. RDBMS. (n.d.). *What is Relational Database Management System?* Retrieved from RDBMS: https://martech.zone/acronym/rdbms/
4. Thanh, B. (n.d.). *Phần mềm XAMPP là gì? Cách sử dụng phần mềm XAMPP như thế nào?* Retrieved from fptshop.com.vn: https://fptshop.com.vn/tin-tuc/danh-gia/phan-mem-xampp-la-gi-cach-su-dung-phan-mem-xampp-nhu-the-nao-153531