



**Daffodil**  
*International*  
**University**

Hyperstore - e-Commerce platform for small businesses

By

**Md. Zahidul Islam**

**ID: 191-35-2675**

**Department of Software Engineering  
Daffodil International University**

Supervised By

**Ms. Nadira Islam**

**Lecturer (Senior Scale)**

**Department of Software Engineering  
Daffodil International University**

A thesis submitted in partial fulfillment of the requirement for the degree  
of Bachelor of Science in Software Engineering

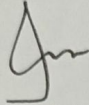
**Department of Software Engineering  
DAFFODIL INTERNATIONAL UNIVERSITY**

Fall – 2022

## APPROVAL


This thesis titled on “Hyperstore - an eCommerce platform for small businesses”, submitted by **Md. Zahidul Islam (ID: 191-35-2675)** to the Department of Software Engineering, Daffodil International University has been accepted as satisfactory for the partial fulfillment of the requirements for the degree of Bachelor of Science in Software Engineering and approval as to its style and contents.

### BOARD OF EXAMINERS



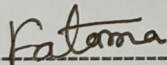
Chairman

-----  
**Dr. Imran Mahmud**  
**Head and Associate Professor**  
Department of Software Engineering  
Faculty of Science and Information  
Technology  
Daffodil International University



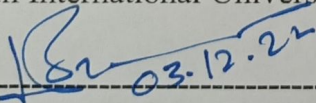
Internal Examiner 1

-----  
**Md. Maruf Hasan**  
**Associate Professor**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University



Internal Examiner 2

-----  
**Fatama Binta Rafiq**  
**Lecturer (Senior)**  
Department of Software Engineering  
Faculty of Science and Information Technology  
Daffodil International University



External Examiner

-----  
**Dr. Md. Sazzadur Rahman**  
**Associate Professor**  
Institute of Information Technology  
Jahangirnagar University

## DECLARATION

I, hereby, declare that this is my original work for the Bachelor of Science in Software Engineering and no part of it has been submitted for a degree in any other university. I was supervised by **Ms. Nadira Islam** and followed her guidance to the best of my abilities.

### Certified By:



---

Ms. Nadira Islam  
Lecturer (Senior Scale)  
Department of Software Engineering  
Daffodil International University



---

Md. Zahidul Islam  
ID: 191-35-2675  
Department of Software Engineering  
Daffodil International University

## ACKNOWLEDGEMENT

It has been a tough long journey. And I could not do it all by myself. First of all, I would like to thank the Almighty Allah who has given me the strength to keep going. Without his grace, this project would have never seen the light. And my parents, who were always there for me no matter what and supported me in every situation.

I feel lucky for the opportunity to study at Daffodil International University. I would like to thank **Prof. Dr. Imran Mahmud**, Head of the Department of Software Engineering. I can not thank my supervisor, **Ms. Nadira Islam**, enough for her guidance. Without her help and support, it would not have been possible.

Every single faculty member here at the Department of Software Engineering at Daffodil International University has helped me throughout my journey and the knowledge that I have gained from them is priceless, so I would like to sincerely thank them.

And last but not least, I would like to thank my friends and classmates who cheered me up all the time and helped me with my studies and believed in me.

## **ABSTRACT**

This project aims to help small businesses to create an online shop within a few minutes without any developers or coding knowledge. Developing an e-commerce website from scratch is time-consuming and expensive, and because of that, it is usually not worth it for small businesses. Hyperstore aims to solve this very problem by making it free and extremely fast to create an e-commerce website. It is not meant for large businesses that already have the resources to develop their own platform, rather it is targeted toward smaller businesses that could be benefited from an online store but do not have the resources to develop one. The platform does not provide a highly customizable website, instead, it provides a fixed layout with a customizable business logo and colors. It also provides a core set of features like showing product information, search functionality, shopping cart, etc. It does not include features like SEO optimization, promotions, coupon codes, etc. that larger businesses would expect.

## TABLE OF CONTENTS

|  |          |
|--|----------|
| <b>Hyperstore - e-Commerce platform for small businesses</b> | <b>1</b> |
| <b>CHAPTER 1</b>   | <b>1</b> |
| <b>INTRODUCTION</b>  | <b>1</b> |
| 1.1 Project Overview   | 1        |
| 1.2 Background   | 1        |
| 1.3 Project Purpose  | 2        |
| 1.4 Benefits and Beneficiaries                               | 2        |
| 1.5 Stakeholders   | 3        |
| 1.6 Modules of Service Assistant                             | 3        |
| 1.7 Tech Stack   | 3        |
| 1.8 Project Schedule   | 4        |
| 1.8.1 Gantt Chart  | 5        |
| <b>CHAPTER 2</b>   | <b>6</b> |
| <b>REQUIREMENT ENGINEERING</b>                               | <b>6</b> |
| 2.1 Development Model  | 6        |
| 2.2 Functional Requirements                                  | 7        |
| 2.3 Non-functional Requirements                              | 7        |
| <b>CHAPTER 3</b>   | <b>8</b> |
| <b>SYSTEM ANALYSIS, DESIGN &amp; SPECIFICATION</b>           | <b>8</b> |
| 3.1 Use Case Diagram   | 8        |
| 3.2 Use case descriptions                                    | 9        |
| 3.3 Class diagram  | 14       |
| 3.4 Activity Diagrams  | 15       |
| 3.4.1 Registration   | 15       |
| 3.4.2 Login  | 16       |
| 3.4.3 Manage user  | 17       |
| 3.4.4 Search Products  | 18       |
| 3.4.5 Place an order   | 19       |
| 3.4.6 Make payment   | 20       |
| 3.4.7 Open a new shop  | 21       |
| 3.4.8 Manage products  | 22       |
| 3.5 Sequence Diagrams  | 23       |
| 3.5.1 Registration   | 23       |
| 3.5.2 Login  | 24       |

|                                      |           |
|--------------------------------------|-----------|
| 3.5.3 Manage users                   | 25        |
| 3.5.4 Search for products            | 26        |
| 3.5.5 Place an order                 | 27        |
| 3.5.6 Open a shop                    | 28        |
| 3.5.7 Manage products                | 29        |
| 3.5.8 Make payment                   | 30        |
| <b>CHAPTER 4</b>                     | <b>31</b> |
| <b>TESTING</b>                       | <b>31</b> |
| 4.1 Introduction                     | 31        |
| 4.2 Test Scenarios and cases         | 31        |
| 4.3 Test result                      | 34        |
| <b>CHAPTER 5</b>                     | <b>35</b> |
| <b>USER MANUAL</b>                   | <b>35</b> |
| 5.1 Shop owner                       | 35        |
| 5.1.1 Register                       | 35        |
| 5.1.2 Login                          | 36        |
| 5.1.3 Open a shop                    | 37        |
| 5.1.4 Admin dashboard (orders chart) | 38        |
| 5.1.4b Admin dashboard               | 38        |
| 5.1.5 Create a product               | 39        |
| 5.2 Customer                         | 40        |
| 5.2.1 Shop page                      | 40        |
| 5.2.2 View product                   | 41        |
| 5.2.3 Cart                           | 42        |
| 5.2.4 Payment                        | 43        |
| <b>CHAPTER 6</b>                     | <b>44</b> |
| <b>CONCLUSION</b>                    | <b>44</b> |
| 6.1 Conclusion                       | 44        |
| 6.2 Limitations                      | 44        |
| 6.3 Future improvements              | 44        |
| <b>REFERENCES</b>                    | <b>45</b> |

## TABLE OF FIGURES

|  |           |
|--|-----------|
| <b>Hyperstore - e-Commerce platform for small businesses</b> | <b>1</b>  |
| <b>CHAPTER 1</b>   | <b>1</b>  |
| <b>INTRODUCTION</b>  | <b>1</b>  |
| Figure 1.1: Most wanted features of an online shop           | 5         |
| <b>CHAPTER 2</b>   | <b>6</b>  |
| <b>REQUIREMENT ENGINEERING</b>                               | <b>6</b>  |
| Figure 2.1: Waterfall method process                         | 6         |
|  | 7         |
| <b>CHAPTER 3</b>   | <b>8</b>  |
| <b>SYSTEM ANALYSIS, DESIGN &amp; SPECIFICATION</b>           | <b>8</b>  |
| Figure 3.1: Use case diagram                                 | 14        |
| Figure 3.2: Class diagram                                    | 14        |
| Figure 3.3: Activity diagram for registration                | 16        |
| Figure 3.4: Activity diagram for login                       | 16        |
| Figure 3.5: Activity diagram for managing users              | 17        |
| Figure 3.6: Activity diagram for search products             | 18        |
| Figure 3.7: Activity diagram for placing an order            | 19        |
| Figure 3.8: Activity diagram for making payment              | 20        |
| Figure 3.9: Activity diagram for opening a new shop          | 21        |
| Figure 3.10: Activity diagram for managing products          | 22        |
| Figure 3.11: Sequence diagram for registration               | 23        |
| Figure 3.12: Sequence diagram for login                      | 24        |
| Figure 3.13: Sequence diagram for manage users               | 25        |
| Figure 3.14: Sequence diagram for search products            | 26        |
| Figure 3.15: Sequence diagram for placing an order           | 27        |
| Figure 3.16: Sequence diagram for opening a shop             | 28        |
| Figure 3.17: Sequence diagram for managing products          | 29        |
| Figure 3.18: Sequence diagram for making payment             | 30        |
| <b>CHAPTER 4</b>   | <b>31</b> |
| <b>TESTING</b>   |           |
| Figure 4.1: Sample test scenario with cases                  | 34        |



|   |           |
|---|-----------|
| Figure 4.2: Test result from the automated test run | 34        |
| <b>CHAPTER 5</b>                                    | <b>35</b> |
| <b>USER MANUAL</b>                                  | <b>35</b> |
| Figure 5.1: Registration                            | 36        |
| Figure 5.2: Login                                   | 36        |
| Figure 5.3: Open a shop                             | 37        |
| Figure 5.4: Admin dashboard (chart)                 | 38        |
| Figure 5.5: Admin dashboard                         | 38        |
| Figure 5.6: Create a product                        | 39        |
| Figure 5.7: shop page (browse products)             | 40        |
| Figure 5.8: View product                            | 41        |
| Figure 5.9: Cart                                    | 42        |
| Figure 5.10: Payment                                | 43        |

## LIST OF NOMENCLATURE

|     |                       |
|-----|-----------------------|
| JWT | JSON web token        |
| BM  | Bi-month (15 days)    |
| SC  | HTTP Status Code      |
| MVC | Model View Controller |

# **CHAPTER 1**

## **INTRODUCTION**

### **1.1 Project Overview**

Nowadays having an e-commerce website is crucial for businesses of all sizes. It helps them reach a broader audience thus increasing revenue. Hyperstore helps small businesses to develop their own e-commerce store within a few minutes.

To open a shop with Hyperstore, they have to Enter a shop name, upload a logo and pick a brand color. After that, they can add products to their store. Then customers can view those products, search for specific products, add them to their cart, and place an order. The shop owner will be able to see the orders in real time as they come in.

### **1.2 Background**

Hyperstore is made with small businesses and their customers in mind. It ensures a win-win situation for both the business and a potential customer by providing a platform for the business and an easy shopping experience for the customer. A customer can use the website easily by utilizing the search and cart feature.

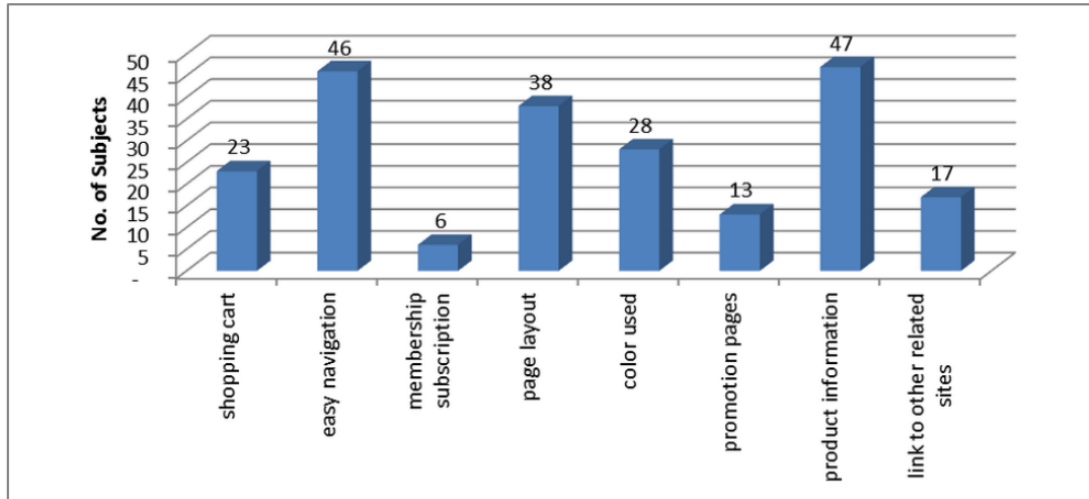


Figure 1.1: Most wanted features of an online shop [1]

Figure 1.1 shows a graph from a survey where subjects voted for their favorite features in an e-commerce store. Hyperstore includes most of the top features like shopping cart, easy navigation, colors, product information, etc.

### 1.3 Project Purpose

The key purposes of the project include

- To give small businesses an online presence.
- Help them reach a broader audience.
- Give them a platform to grow their business and identity by allowing them to upload their own logo and brand color.
- Make it easy for small business owners to create a website without any coding knowledge.

### 1.4 Benefits and Beneficiaries

The main benefits of using Hyperstore are, businesses can create their website for free and within a few minutes. It benefits both the business owner and their customer.

## 1.5 Stakeholders

Stakeholders are the people who are directly or indirectly involved with the project.

There are a few stakeholders for Hyperstore but the key stakeholders for this project include

- Shop owner
- Shop admin (can be the same person as the shop owner)
- Customers

## 1.6 Modules of Service Assistant

- User module: shop owners and customers
- Auth module: user authentication and authorization
- Shop module
- Product module
- Order module

## 1.7 Tech Stack

- Programming language
  - **Typescript:** Typescript was chosen as the main programming language because it enables type safety for Javascript, helps to catch bugs sooner rather than later.
- Backend
  - **Node JS:** The express framework on top of Node JS was chosen because it is a very lightweight framework and only includes basic features. Then more features can be added as needed. This makes it highly customizable and fast.
- Database
  - **MongoDB:** MongoDB was chosen as the primary database because it is a high performant non-relational database and enables easy and low-cost scalability. [5]

- **Redis:** I choose Redis to cache frequently accessed information like product lists. Because of Redis, the client enjoys a faster response time and the server does not get flooded with a lot of requests.
- Frontend
  - **Next JS:** It is a framework based on React JS. Next JS enables server-side rendering and easy routing. That's why I choose Next JS as my frontend framework.
  - **Tailwind CSS:** I choose tailwind CSS to style the website because it is the best way to write CSS [4]. It uses raw CSS under the hood and we can achieve the same results just faster.
- Others
  - **Socket.io:** Socket.io is a technology that enables bi-directional communication between platforms. I used it to update the order status in real time.
  - **Jsonwebtoken:** I used JWT for user authentication. I choose it because it is a secure way to send information between 2 parties. If even one letter is changed, the whole token gets invalidated. It also includes a signature, which we can use to verify that the token is coming from the right sender.

## 1.8 Project Schedule

The project was done with a tight schedule. A Gantt chart was followed to ensure it is finished within the deadline.

### 1.8.1 Gantt Chart

Table 1.1: Gantt Chart

| Activity         | BM1 | BM2 | BM3 | BM4 | BM5 | BM6 | BM7 | BM8 |
|------------------|-----|-----|-----|-----|-----|-----|-----|-----|
| Research         |     |     |     |     |     |     |     |     |
| Select topic     |     |     |     |     |     |     |     |     |
| Specifications   |     |     |     |     |     |     |     |     |
| Presentation     |     |     |     |     |     |     |     |     |
| Planning         |     |     |     |     |     |     |     |     |
| Design           |     |     |     |     |     |     |     |     |
| Backend          |     |     |     |     |     |     |     |     |
| Frontend         |     |     |     |     |     |     |     |     |
| Testing          |     |     |     |     |     |     |     |     |
| Bug fixes        |     |     |     |     |     |     |     |     |
| Prepare for prod |     |     |     |     |     |     |     |     |

## CHAPTER 2

### REQUIREMENT ENGINEERING

#### 2.1 Development Model

For the development of Hyperstore I have chosen the waterfall model because I am the sole developer of this project and the requirements were concrete from the very beginning. I did not have to change anything in the middle and the goal was clear from the very beginning. Keeping all of these things in mind, I think the waterfall model is the perfect fit for my project.

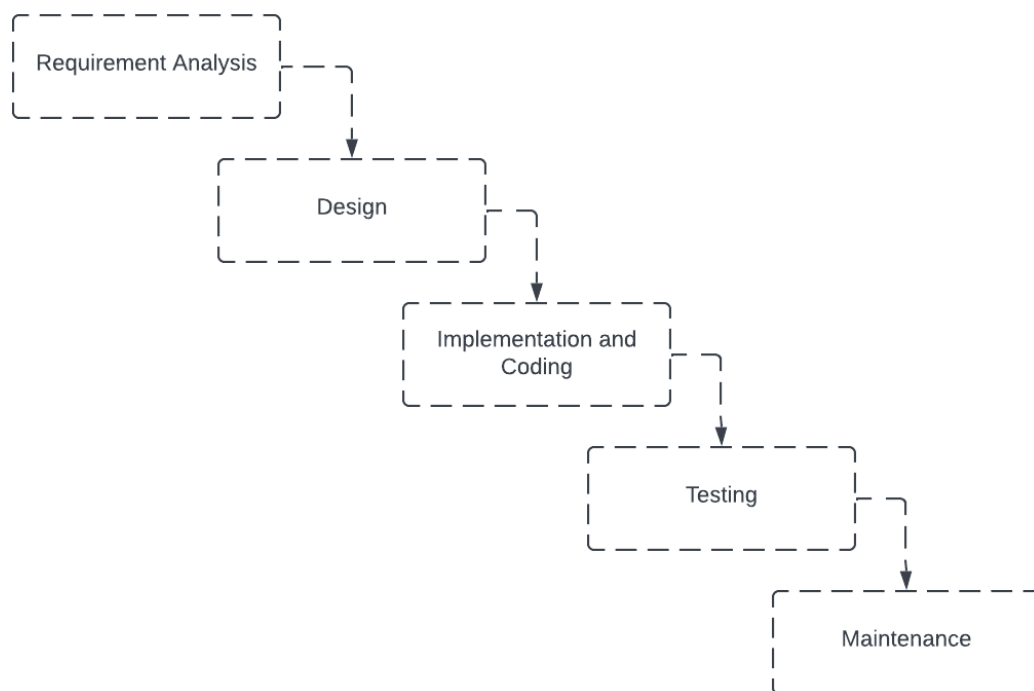


Figure 2.1: Waterfall method process



## 2.2 Functional Requirements

- Manage customer and store profiles.
- Customer and store Authentication.
- Create a store.
- Manage products.
- Search for products.
- Add products to the cart.
- Place orders.
- Pay with a card.

## 2.3 Non-functional Requirements

- **Security:** Implement a secure payment flow and role-based authorization.
- **Usability:** Core features should be easy to use. Any shop owner should be able to create a store and customers should be able to make a purchase easily.
- **Performance:** The website should have tolerable loading times.

## **CHAPTER 3**

### **SYSTEM ANALYSIS, DESIGN & SPECIFICATION**

#### **3.1 Use Case Diagram**

In the use case diagram, we have 4 actors: customer, shop admin, payment gateway, and admin. Here, the customer is the primary actor. They can log in, register as well as view products, place an order, make payments, etc. The shop admin is responsible to update products and processing orders. The payment gateway handles everything related to payment. And the admin can manage every user including shop admins and customers.

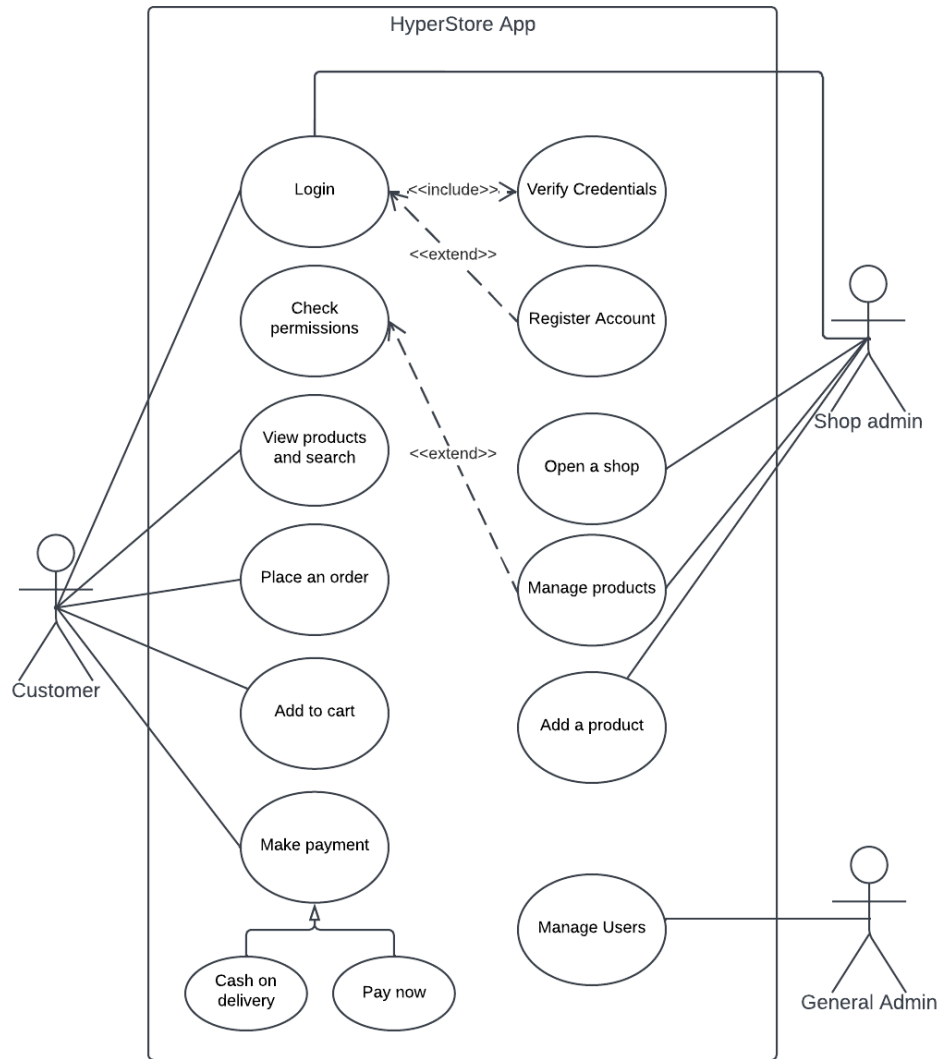


Figure 3.1: Use case diagram

### 3.2 Use case descriptions

Table 3.1: Use case description for Register

|                       |  |
|-----------------------|--|
| <b>Use case</b>       | Register   |
| <b>Pre-condition</b>  | 1. N/A   |
| <b>Actors</b>         | Shop owner, customer                               |
| <b>Triggers</b>       | User clicks on “Don’t have an account? Create one” |
| <b>Post-condition</b> | 1. User account is created                         |

|                         |  |
|-------------------------|--|
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter full name</li> <li>2. Enter email address</li> <li>3. Enter password</li> <li>4. Enter phone number</li> <li>5. Enter address</li> <li>6. Validate information</li> <li>7. Validation successful</li> <li>8. Click “Register”</li> </ol> |
| <b>Alternative path</b> | N/A  |

Table 3.2: Use case description for Login

|                         |  |
|-------------------------|--|
| <b>Use case</b>         | Login  |
| <b>Pre-condition</b>    | <ol style="list-style-type: none"> <li>1. Have created an account</li> </ol>   |
| <b>Actors</b>           | Shop owner, customer   |
| <b>Triggers</b>         | User clicks on “Log in”  |
| <b>Post-condition</b>   | <ol style="list-style-type: none"> <li>1. User is logged in</li> </ol>   |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter email address</li> <li>2. Enter password</li> <li>3. Validate and verify email and password</li> <li>4. Click “Login”</li> </ol> |
| <b>Alternative path</b> | N/A  |

Table 3.3: Use case description for View products and search

|                         |  |
|-------------------------|--|
| <b>Use case</b>         | View products and search   |
| <b>Pre-condition</b>    | <ol style="list-style-type: none"> <li>1. User is logged in</li> </ol>   |
| <b>Actors</b>           | Customer   |
| <b>Triggers</b>         | User visits a shop page  |
| <b>Post-condition</b>   | <ol style="list-style-type: none"> <li>1. User is viewing filtered products</li> </ol>   |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter search query</li> <li>2. Check the database for matches</li> <li>3. Found products based on the query</li> </ol> |
| <b>Alternative path</b> | <ol style="list-style-type: none"> <li>1. In step 2: No products found for the specified query</li> <li>2. No products found</li> </ol>                          |

Table 3.4: Use case description for placing an order

|                         |  |
|-------------------------|--|
| <b>Use case</b>         | Place an order   |
| <b>Pre-condition</b>    | 1. User is logged in   |
| <b>Actors</b>           | Customer   |
| <b>Triggers</b>         | User adds a product to their cart  |
| <b>Post-condition</b>   | 1. User placed an order  |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Select a product variant</li> <li>2. Add product to cart</li> <li>3. Go to cart</li> <li>4. Click on “Checkout”</li> </ol> |
| <b>Alternative path</b> | N/A  |

Table 3.5: Use case description for Make payment

|                         |  |
|-------------------------|--|
| <b>Use case</b>         | Make payment   |
| <b>Pre-condition</b>    | <ol style="list-style-type: none"> <li>1. User is logged in</li> <li>2. User placed an order</li> </ol>  |
| <b>Actors</b>           | Customer, payment gateway  |
| <b>Triggers</b>         | User clicks on “Checkout” from the cart page   |
| <b>Post-condition</b>   | 1. Payment status of the order is “Paid”   |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter credit card number</li> <li>2. Enter Expiry date</li> <li>3. Enter CVC</li> <li>4. Enter name on card</li> <li>5. Click on “Pay”</li> <li>6. Validate card info and check balance</li> <li>7. The card is valid and has enough balance</li> <li>8. Payment successful</li> </ol> |
| <b>Alternative path</b> | <ol style="list-style-type: none"> <li>1. In step 7: Card is not valid and/or does not have enough balance</li> <li>2. Payment unsuccessful</li> </ol>   |

Table 3.6: Use case description for Open a shop

|                         |   |
|-------------------------|---|
| <b>Use case</b>         | Open a shop   |
| <b>Pre-condition</b>    | 1. User is logged in  |
| <b>Actors</b>           | Shop owner  |
| <b>Triggers</b>         | User clicks on “Open a shop”  |
| <b>Post-condition</b>   | 1. User has created their own store   |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter shop name</li> <li>2. Enter shop description</li> <li>3. Upload shop logo</li> <li>4. Pick brand color</li> <li>5. Validate information</li> <li>6. Validation successful</li> <li>7. Shop created</li> </ol> |
| <b>Alternative path</b> | <ol style="list-style-type: none"> <li>1. In step 6: validation unsuccessful</li> <li>2. Shop not created</li> </ol>  |

Table 3.7: Use case description for Add a product

|                         |   |
|-------------------------|---|
| <b>Use case</b>         | Add a product   |
| <b>Pre-condition</b>    | <ol style="list-style-type: none"> <li>1. User is logged in</li> <li>2. User is the owner of the shop that owns this product</li> </ol>   |
| <b>Actors</b>           | Shop owner  |
| <b>Triggers</b>         | User clicks on “Add product”  |
| <b>Post-condition</b>   | 1. A new product is added to the shop   |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter product name</li> <li>2. Enter product description</li> <li>3. Upload product image</li> <li>4. Enter product price</li> <li>5. Click on “Create”</li> <li>6. Validate information</li> <li>7. Validation successful</li> <li>8. Product created</li> </ol> |
| <b>Alternative path</b> | <ol style="list-style-type: none"> <li>1. In step 11: validation unsuccessful</li> <li>2. Product not created</li> </ol>  |

Table 3.8: Use case description for Manage a product

|                         |   |
|-------------------------|---|
| <b>Use case</b>         | Manage a product  |
| <b>Pre-condition</b>    | <ol style="list-style-type: none"> <li>1. User is logged in</li> <li>2. User is the owner of the shop that owns this product</li> </ol>   |
| <b>Actors</b>           | Shop owner  |
| <b>Triggers</b>         | User clicks on “Edit” or “Delete” next to a product   |
| <b>Post-condition</b>   | <ol style="list-style-type: none"> <li>1. Product is updated or deleted</li> </ol>  |
| <b>Basic path</b>       | <ol style="list-style-type: none"> <li>1. Enter new product name</li> <li>2. Enter new product description</li> <li>3. Upload a new image</li> <li>4. Enter new price</li> <li>5. Click on “Done”</li> <li>6. Product is updated</li> </ol> |
| <b>Alternative path</b> | <ol style="list-style-type: none"> <li>1. Click on “Delete”</li> <li>2. Click “Yes” in the confirmation dialogue</li> <li>3. Product is deleted</li> </ol>  |

### 3.3 Class diagram

Class diagram showing all the modules and the relationships between them of the application. Here we have four modules: User, Shop, Order, and Product. A user can zero to many shops but a shop can have only one owner. A shop can have many products but a product can only belong to one shop. A product can be in many orders and an order can have many products. An user can have many orders but an order can only belong to one user.

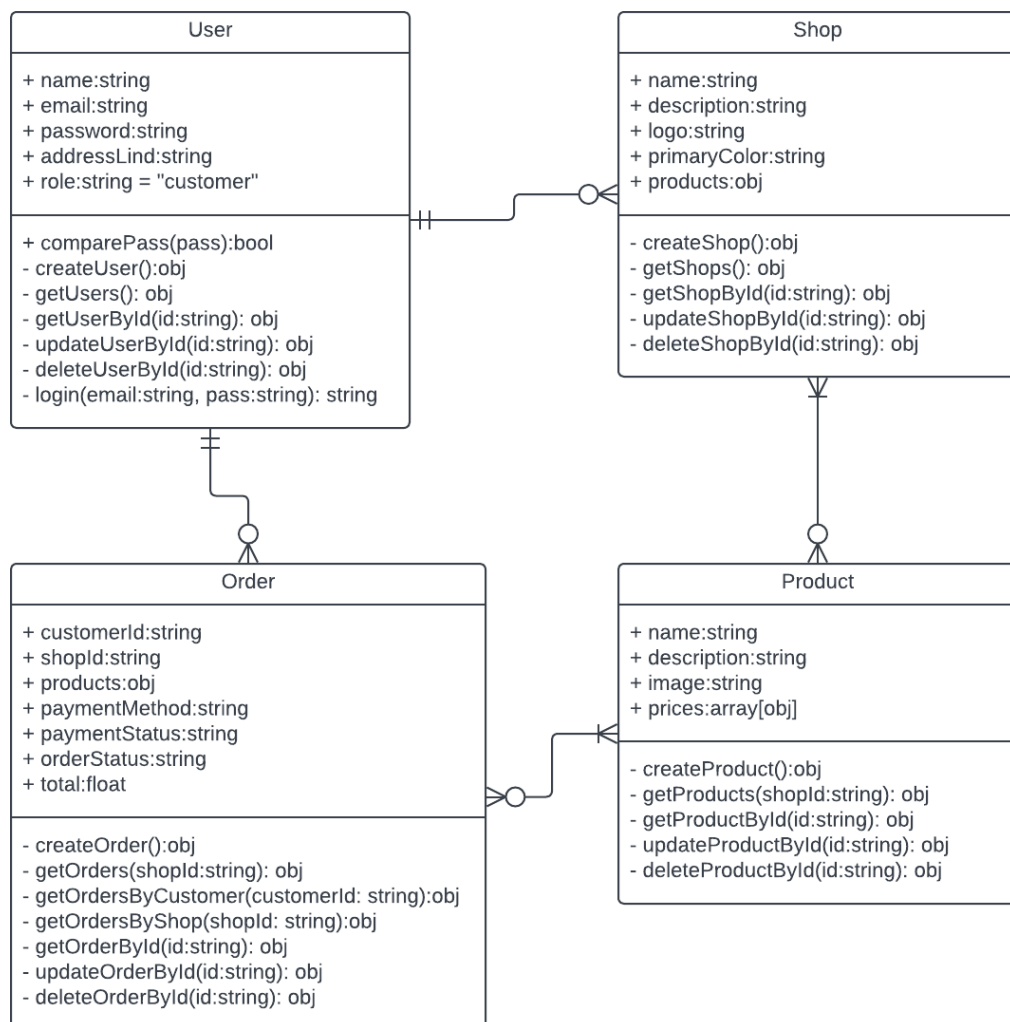


Figure 3.2: Class diagram



### 3.4 Activity Diagrams

#### 3.4.1 Registration

Enter full name, email address, password, phone number, and address. Then this information is validated, if they are valid, the account gets created, otherwise, the user is asked to enter their information again.

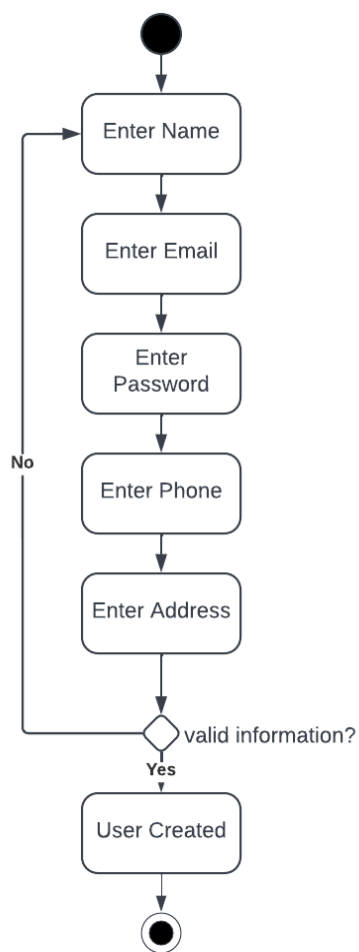


Figure 3.3: Activity diagram for registration

Figure 3.3 shows the activity diagram for user registration.

### 3.4.2 Login

Enter email and password. Then the email and hashed password are checked against the database. If a match is found, the user gets logged in, otherwise, they are asked to enter the information again.

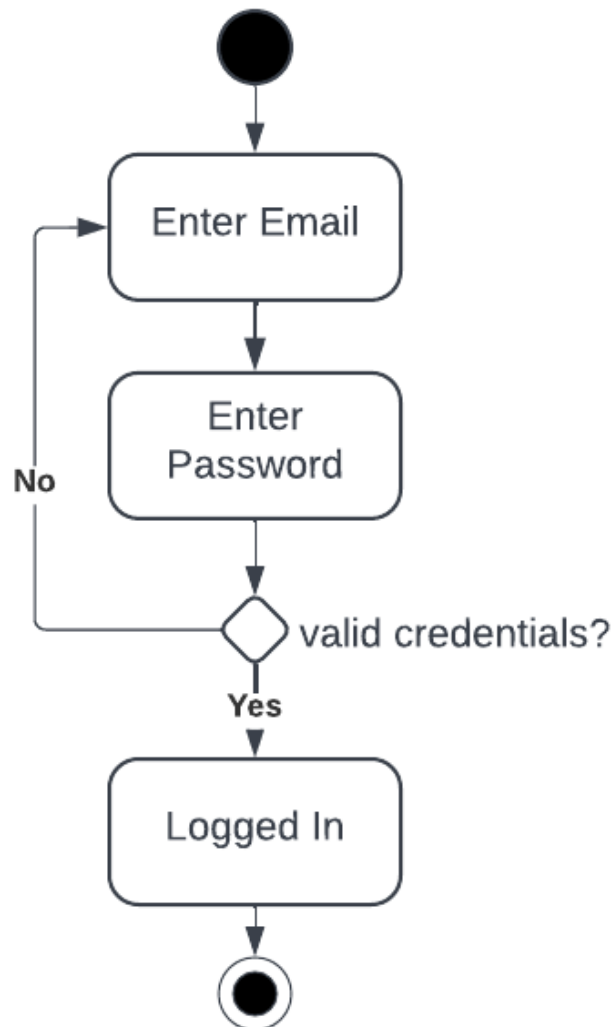


Figure 3.4: Activity diagram for login

Figure 3.4 shows the activity diagram for user login.

### 3.4.3 Manage user

First, check if the currently authenticated user is an admin. If not, check if the currently logged-in user is authorized to edit/delete the account. If not, the user is not allowed to perform this action. If the user is an admin, then they are allowed to edit/delete the user.

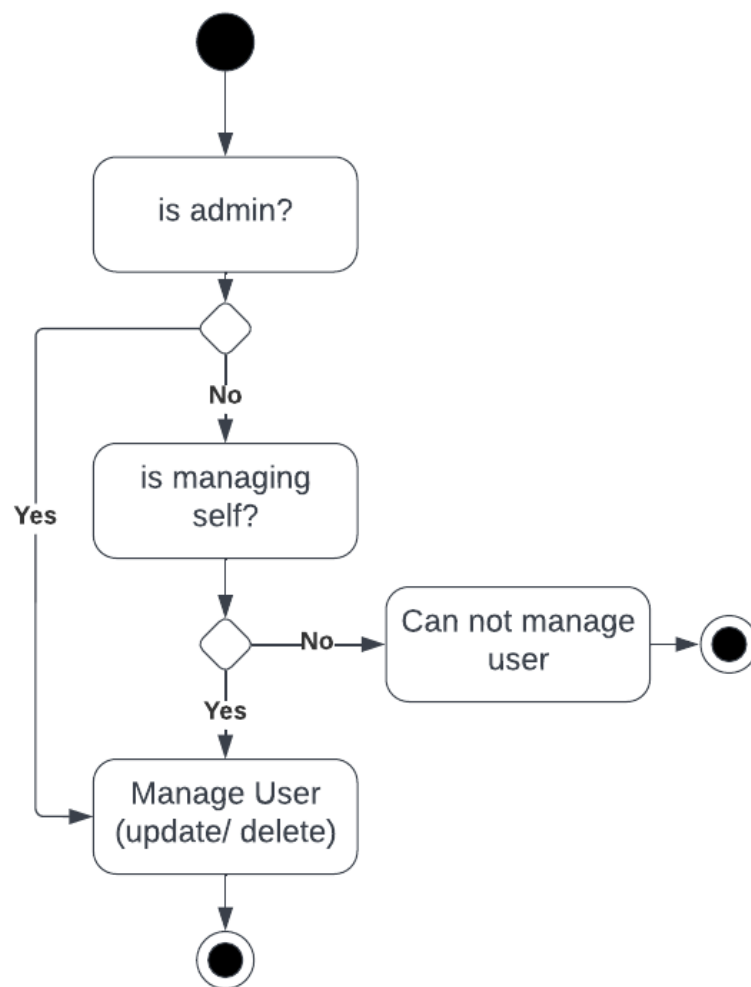


Figure 3.5: Activity diagram for managing users

Figure 3.5 shows the activity diagram for managing users.

### 3.4.4 Search Products

First, the user enters a search query, then a query is run on the database against the query. If at least one product is found, show the product, otherwise show “no product found”.

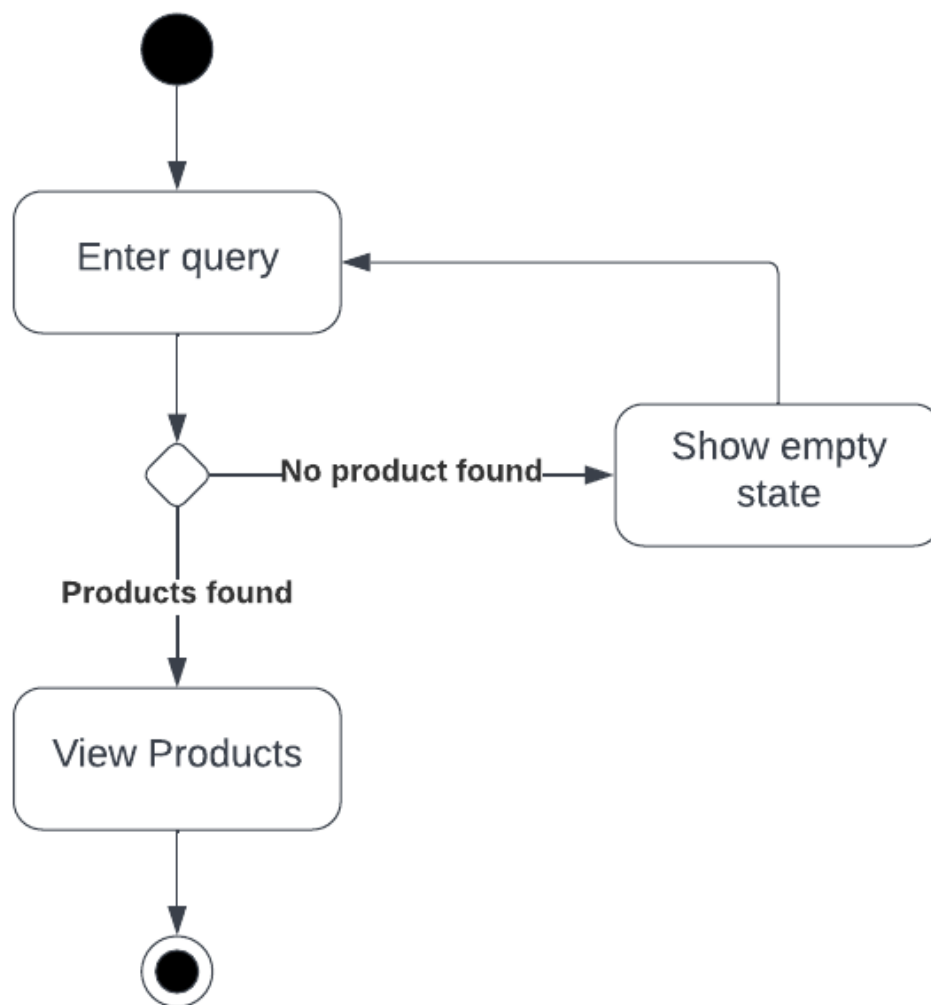


Figure 3.6: Activity diagram for search products

Figure 3.6 shows the activity diagram for searching for products.

### 3.4.5 Place an order

User selects a variant they like and adds the product to the cart. Then they can either pay with a card or pay with cash on delivery. If the user selects to pay with a card, they are sent to the checkout page.

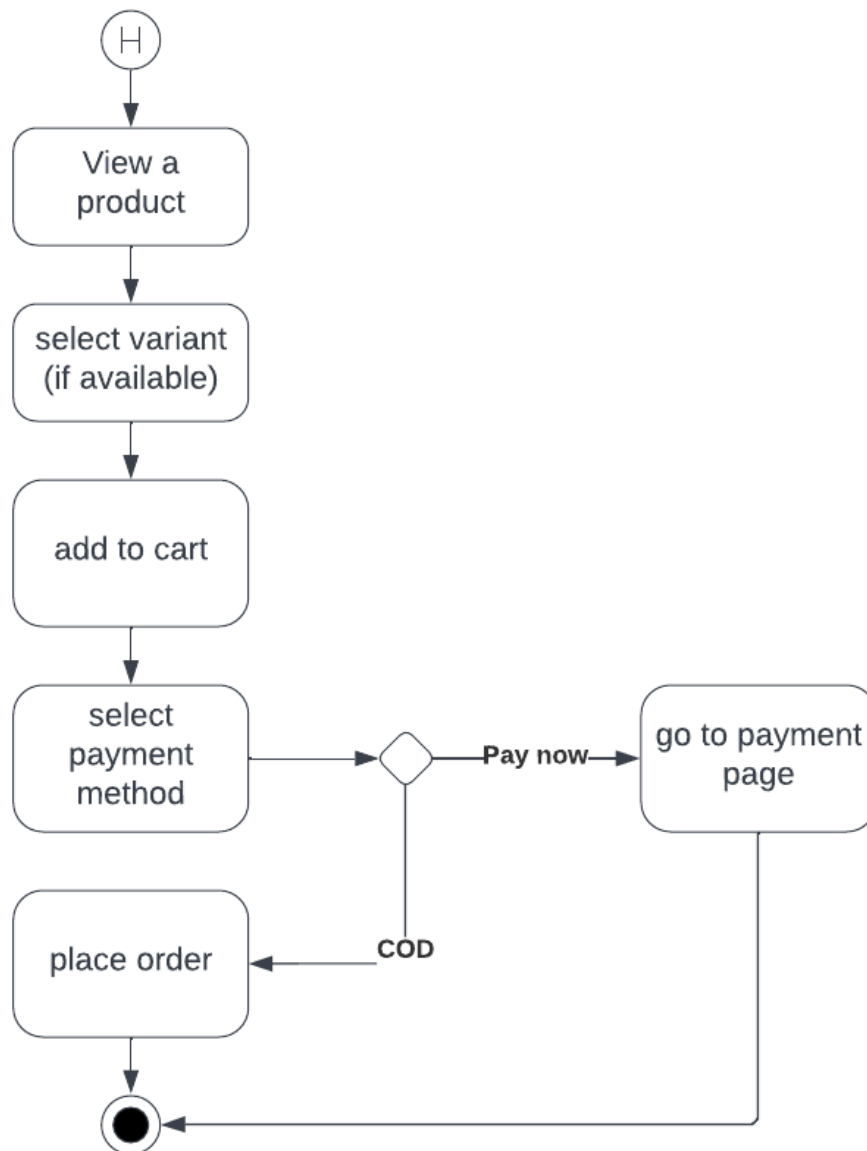


Figure 3.7: Activity diagram for placing an order

Figure 3.7 shows the activity diagram for placing an order.

### 3.4.6 Make payment

On the checkout page, the user enters their credit card number, expiry date, CVC, and name on the card. If the card details are valid, their payment was successful, otherwise unsuccessful.

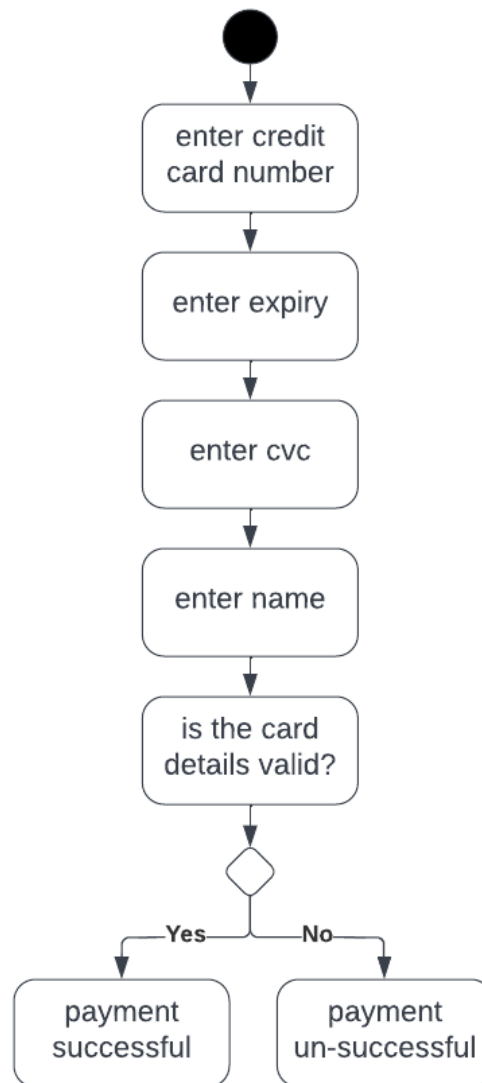


Figure 3.8: Activity diagram for making payment

Figure 3.8 shows the activity diagram for making payment.

### 3.4.7 Open a new shop

To open a shop, first enter a shop name, and shop description, upload the business logo, and pick a brand color. If all information is correct, a shop gets created in the database, otherwise, the user is asked to enter the information again.

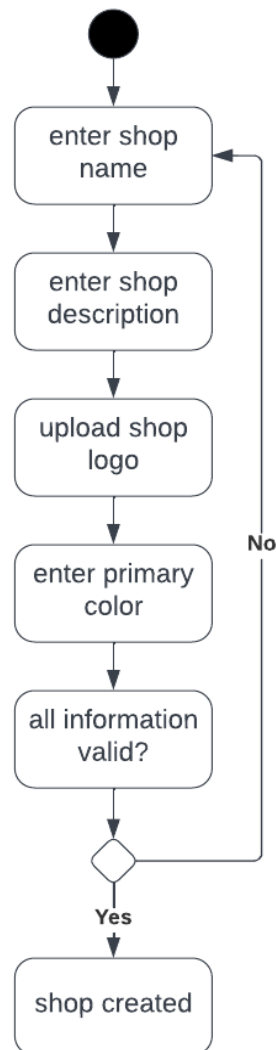


Figure 3.9: Activity diagram for opening a new shop

Figure 3.9 shows the activity diagram for opening a shop.

### 3.4.8 Manage products

To edit/delete a product, first check if the user is an admin. If yes, they are allowed to update the product. If not, check if the user is the owner of the store that this product belongs to. If yes, they are allowed to update the product, otherwise not.

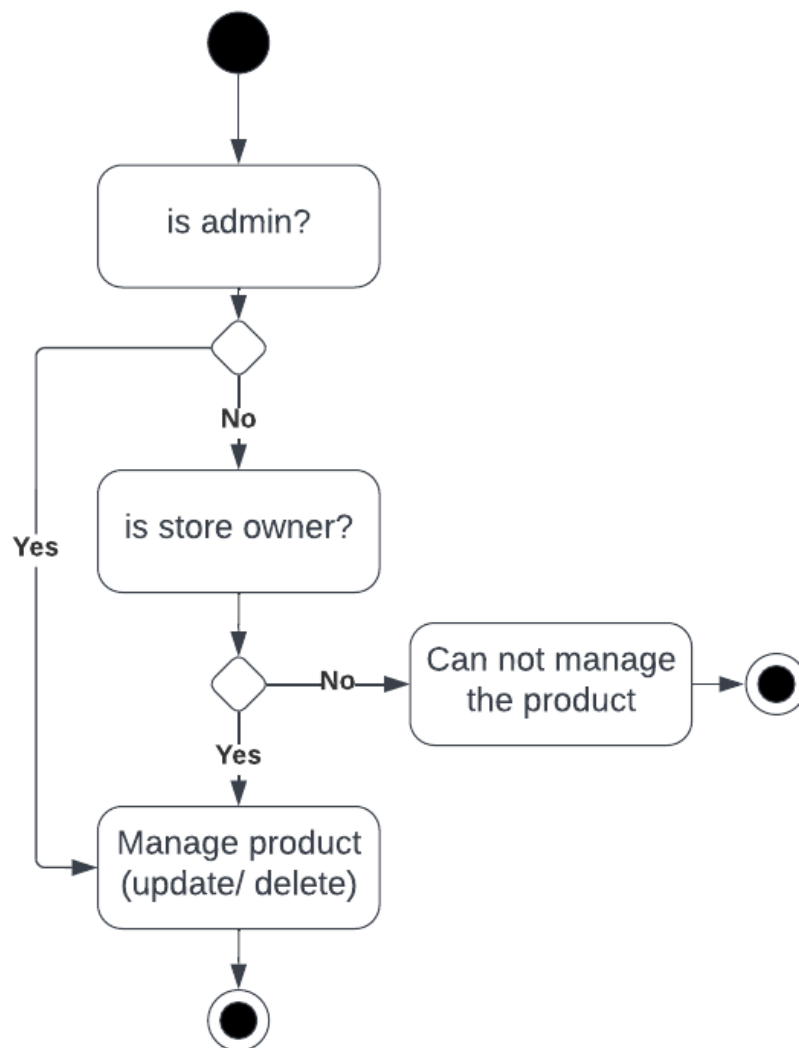


Figure 3.10: Activity diagram for managing products

Figure 3.10 shows the activity diagram for managing products.



### 3.5 Sequence Diagrams

#### 3.5.1 Registration

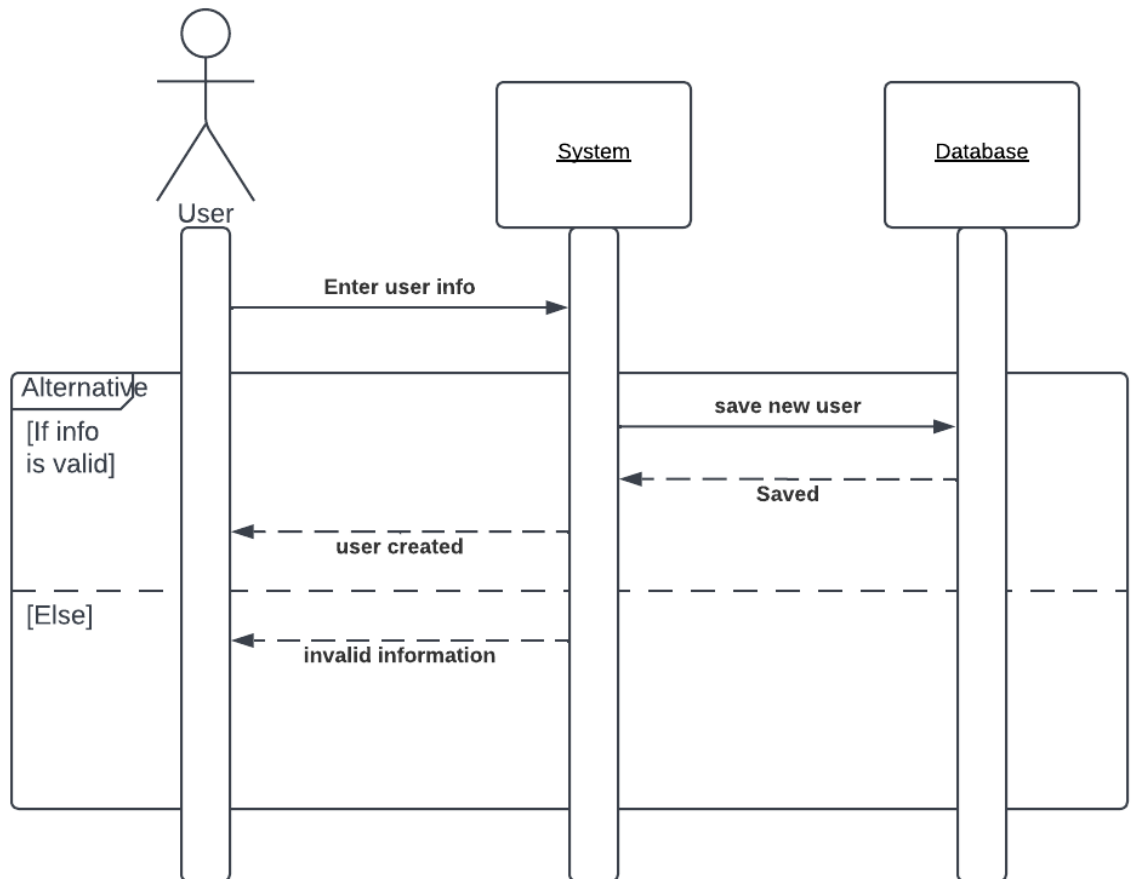


Figure 3.11: Sequence diagram for registration

Figure 3.11 shows the sequence diagram for user registration.

### 3.5.2 Login

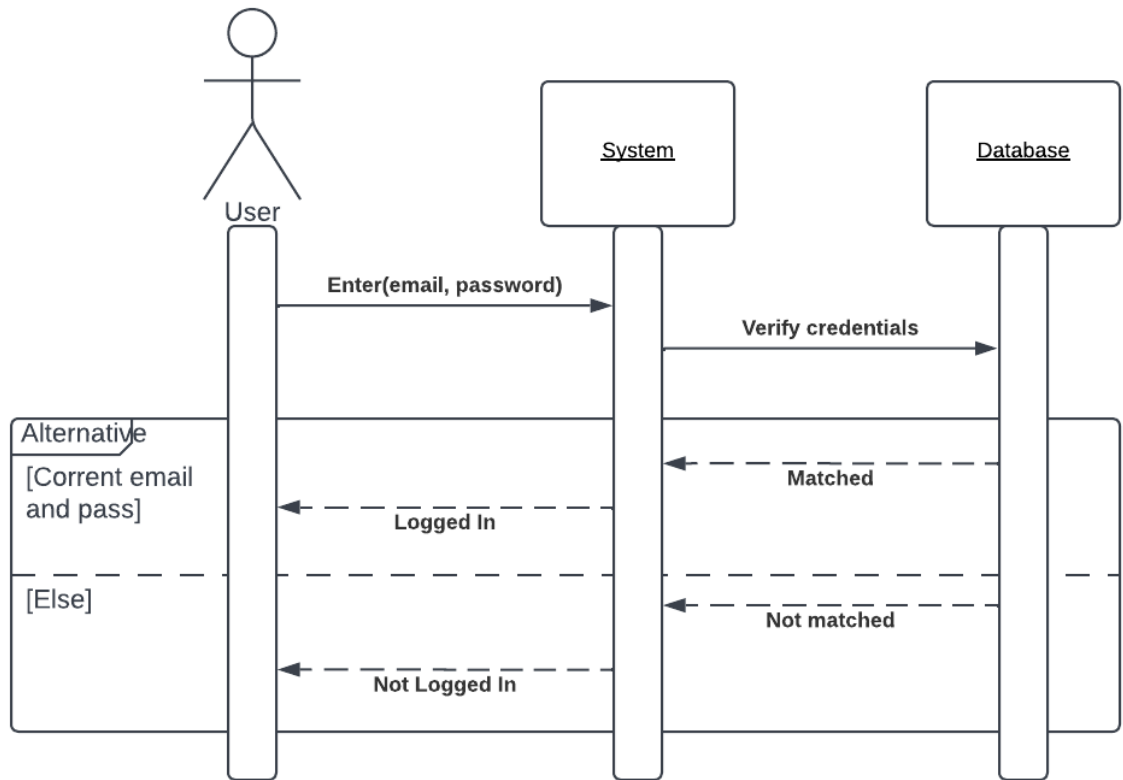


Figure 3.12: Sequence diagram for login

Figure 3.12 shows the sequence diagram for login.

### 3.5.3 Manage users

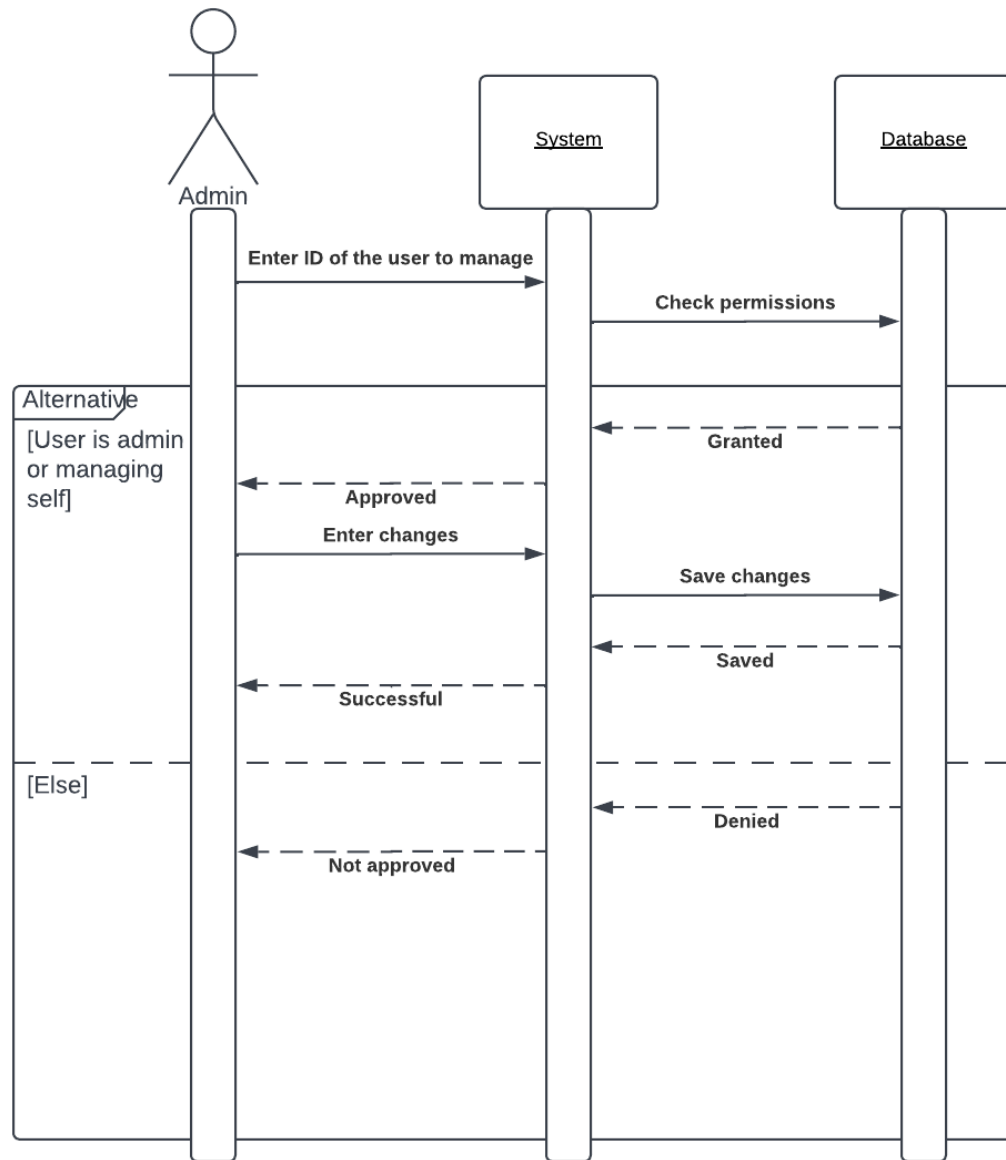


Figure 3.13: Sequence diagram for manage users

Figure 3.13 shows the sequence diagram for managing users.

### 3.5.4 Search for products

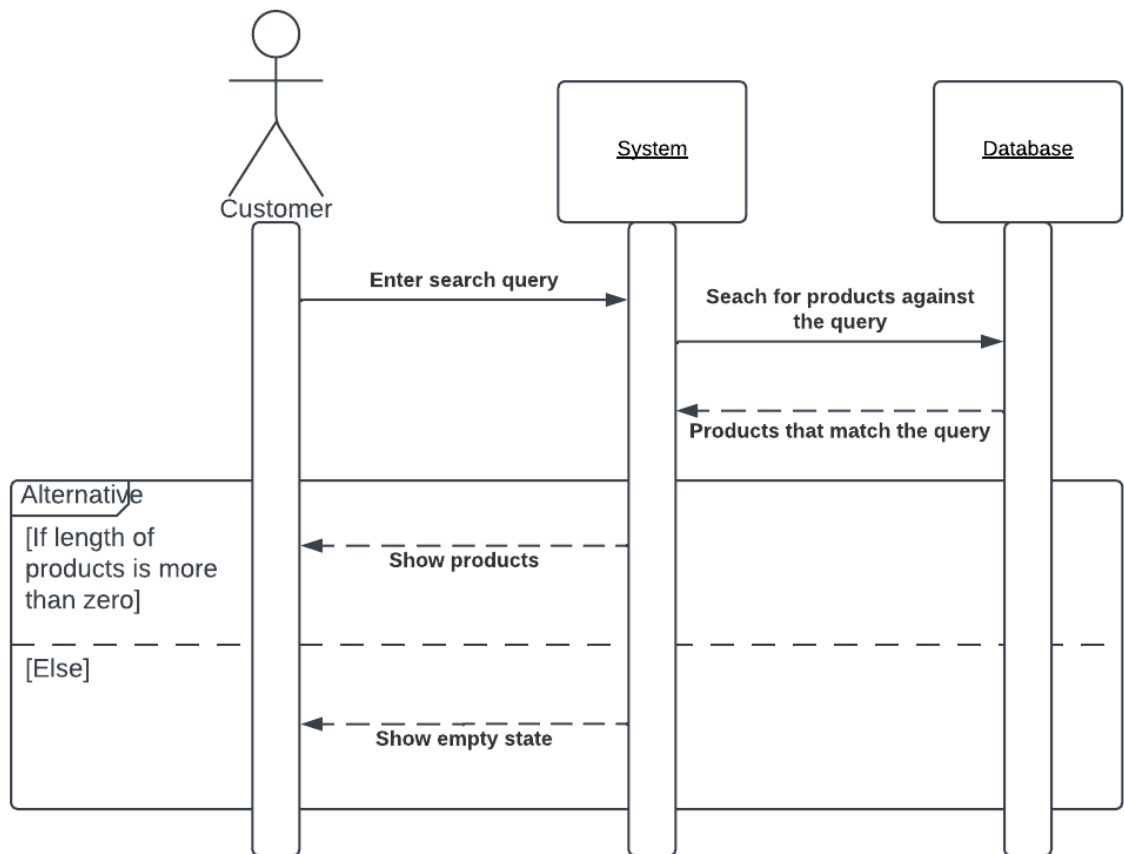


Figure 3.14: Sequence diagram for search products

Figure 3.14 shows the sequence diagram for search products.

### 3.5.5 Place an order

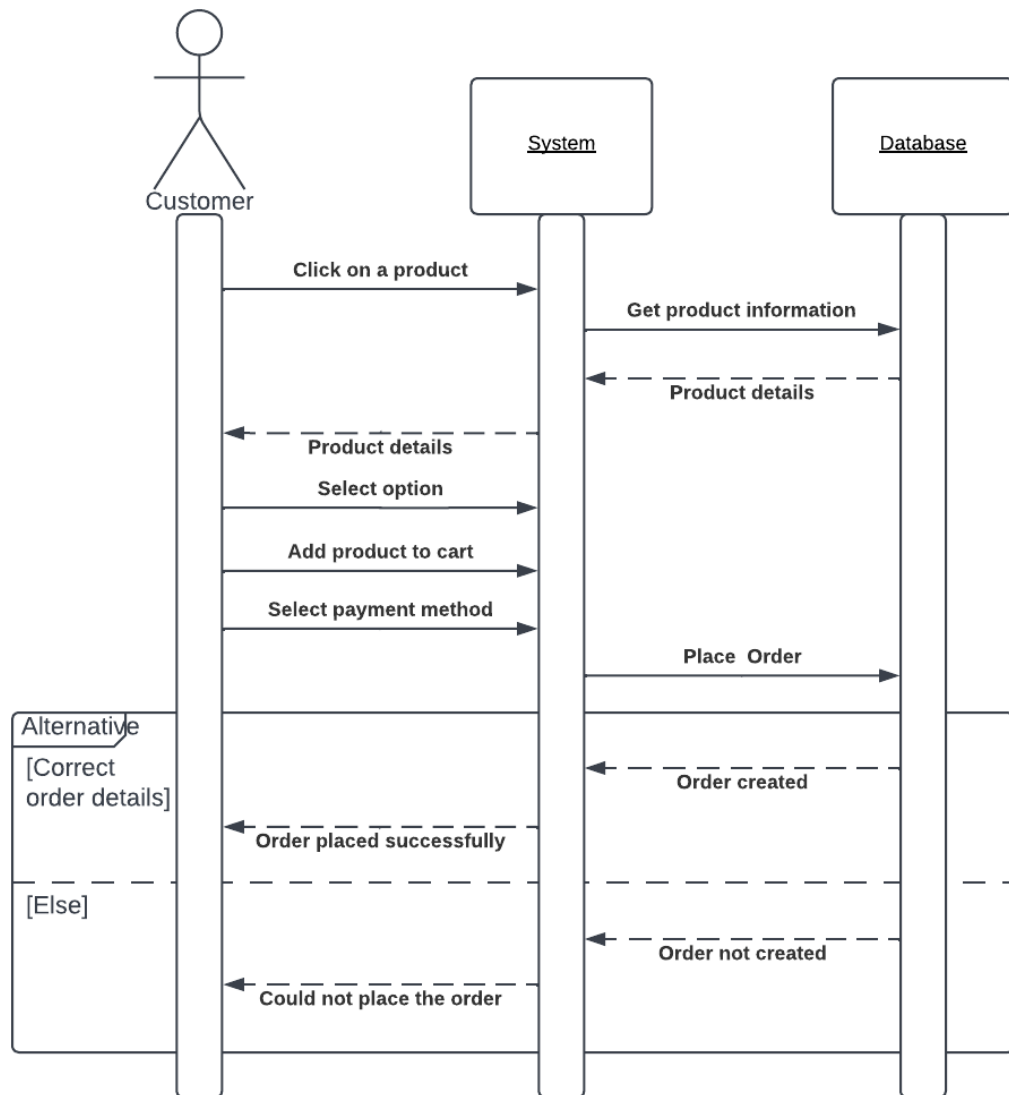


Figure 3.15: Sequence diagram for placing an order

Figure 3.15 shows the sequence diagram for placing an order.

### 3.5.6 Open a shop

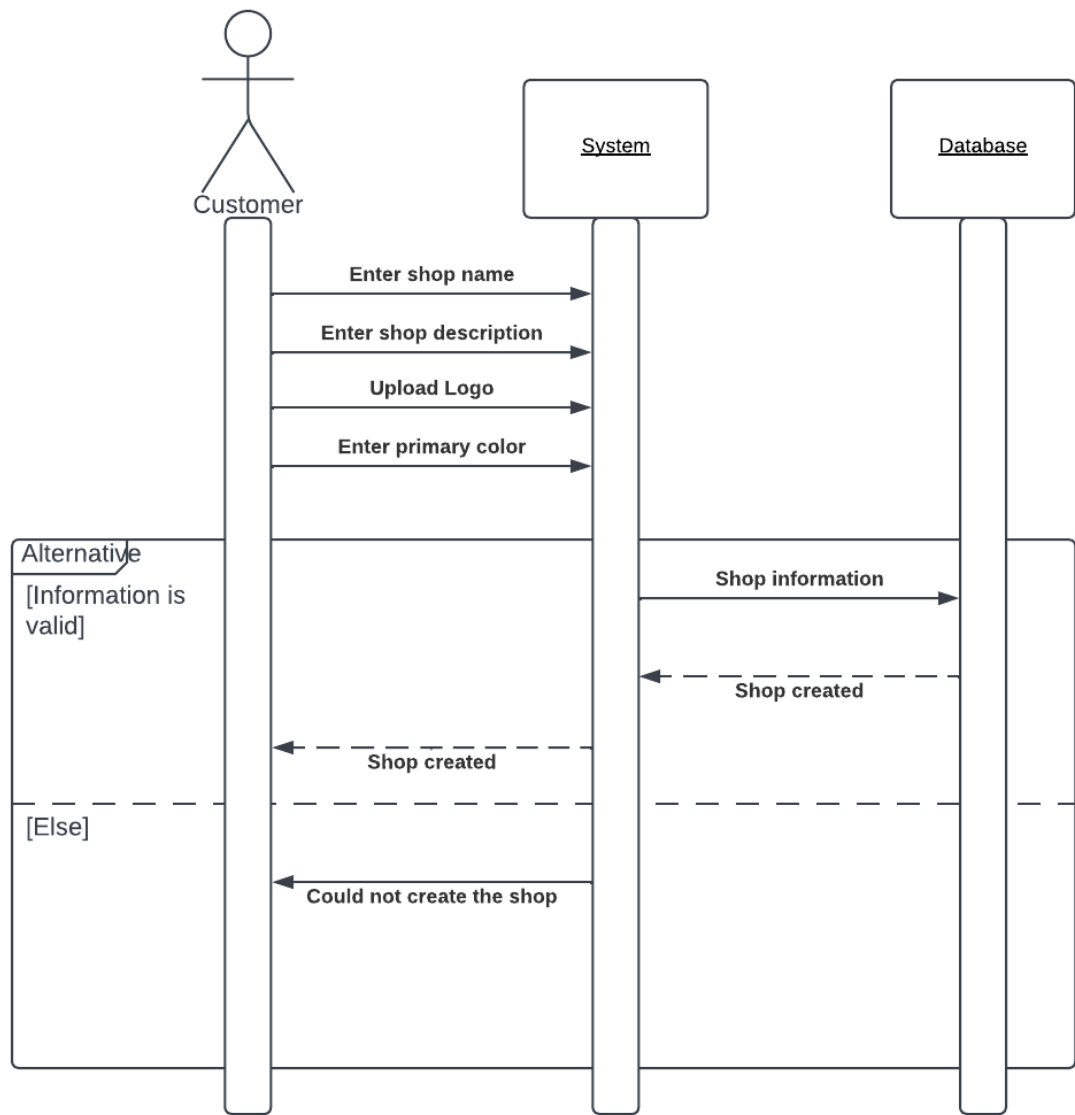


Figure 3.16: Sequence diagram for opening a shop

Figure 3.16 shows the sequence diagram for opening a shop.

### 3.5.7 Manage products

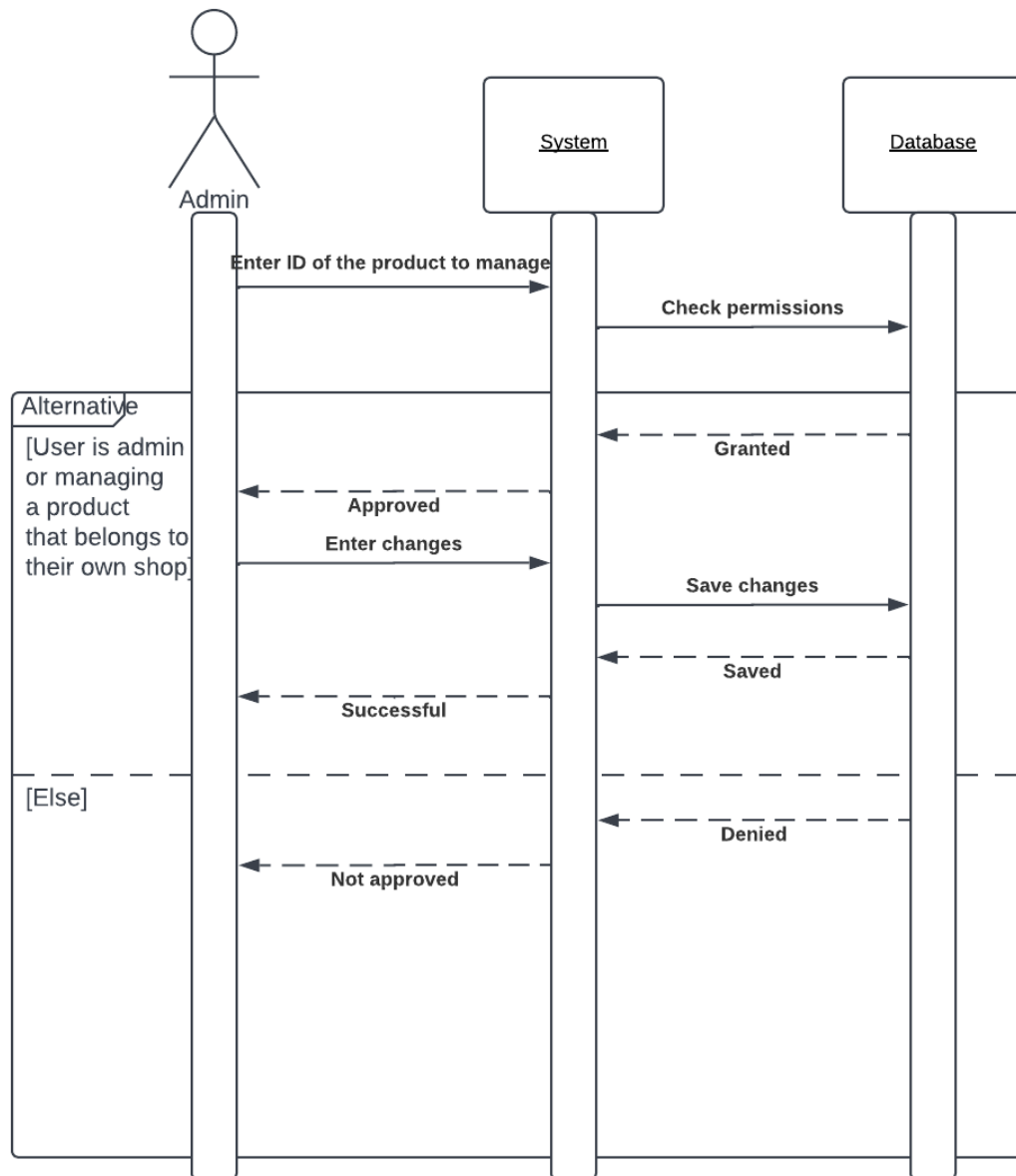


Figure 3.17: Sequence diagram for managing products

Figure 3.17 shows the sequence diagram for managing products.

### 3.5.8 Make payment

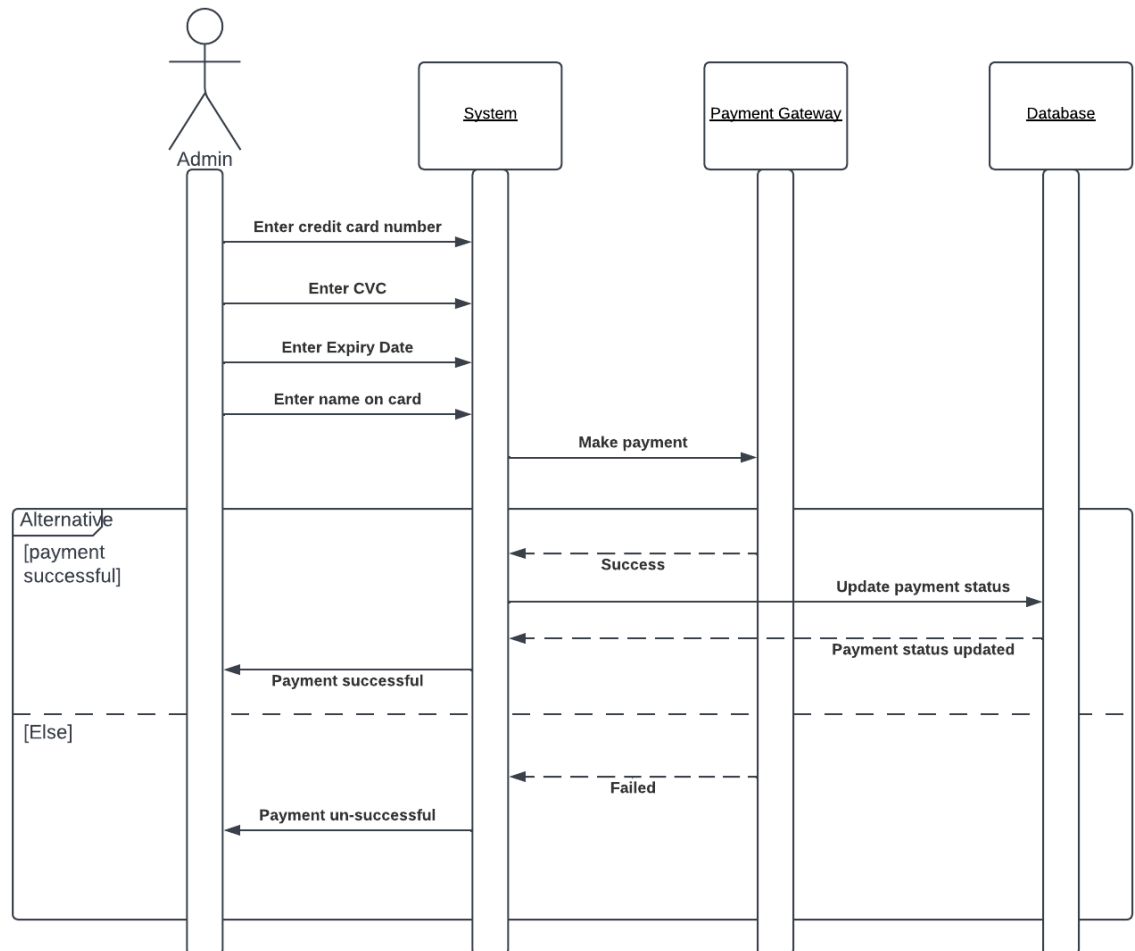


Figure 3.18: Sequence diagram for making payment

Figure 3.18 shows the sequence diagram for making payment.



## CHAPTER 4

### TESTING

#### 4.1 Introduction

Both automated and manual testing was done during the development of Hyperstore to ensure everything works as expected.

#### 4.2 Test Scenarios and cases

Tests were written based use cases. Here is a scenario where a user tries to create a product for their shop:

```
create product
  POST /api/v1/product/
  given user is not authenticated
    ✓ should return a 401 (279 ms)
  given authorized user is not an admin
    ✓ should return a 403 (282 ms)
  given authorized user is an admin but didn't provide name and price info
    ✓ should return a 400 and not create a product (43 ms)
  given authorized user is an admin and provided name and price info
    ✓ should return a 201 and create a product (64 ms)
```

Figure 4.1: Sample test scenario with cases

Table 4.1: Test scenarios and cases

| Scenario    | Case  | Expected result with SC                   |
|-------------|---|---|
| create user | given name, email, password is not provided | should return a 400 and not create a user |
| create user | given name, email, password is provided     | should return a 201 and create a user     |
| create user | given extra unaccepted fields are           | should return a 400                       |

|                               |  |   |
|-------------------------------|--|---|
|                               | provided   |   |
| create user                   | given duplicate email provided   | should return a 409   |
| get users                     | given user is not authenticated  | should return a 401   |
| get users                     | given the user is not an admin   | should return a 403   |
| get users                     | given an admin is authenticated and some users exist                                       | should return a 200 and less than or equal to 'limit' number of users |
| get specific user             | given user is not authenticated  | should return a 401   |
| get specific user             | given the user is not an admin   | should return a 403   |
| get specific user             | given a user with a specific id don't exist  | should return a 404   |
| get specific user             | given a user with a specific id exist  | should return a 200 and the user                                      |
| Update a specif user          | given the user is not authenticated  | should return a 401 and not update the user                           |
| Update a specif user          | given the user is trying to update someone else's profile                                  | should return a 403   |
| Update a specif user          | given the user is authorized as the correct user but trying to change to a duplicate email | should return a 409   |
| Update a specif user          | given the user is authorized as the correct user   | should return a 200 and update the user                               |
| Delete a specific user        | given the user is not authenticated  | should return a 401 and not delete the user                           |
| Delete a specific user        | given the user is trying to delete someone else's profile                                  | should return a 403   |
| Delete a specific user        | given the user is authorized as the correct user   | should return a 200 and delete the user                               |
| Get orders of a specific user | given the user is not authenticated  | should return a 401   |
| Get orders of a specific user | given the user have placed 2 orders  | should return a 200 and 2 orders                                      |
| Login                         | given email and password is not provided   | should return a 400   |

|                        |   |  |
|------------------------|---|--|
| Login                  | given incorrect email is provided   | should return a 404  |
| Login                  | given incorrect password is provided  | should return a 401  |
| Login                  | given correct email and password is provided                                | should return a 200 and send an accessToken                              |
| Get current user       | given no user is logged in  | should return a 401  |
| Get current user       | given a user is logged in   | should return a 200 and the user   |
| Get a new access token | given no refresh token is provided  | should return a 400  |
| Get a new access token | Given a valid refresh token is provided                                     | should return a 200 and a new accessToken                                |
| create product         | given user is not authenticated   | should return a 401  |
| create product         | given authenticated user is not an admin                                    | should return a 403  |
| create product         | given authenticated user is an admin but didn't provide name and price info | should return a 400 and not create a product                             |
| create product         | given authenticated user is an admin and provided name and price info       | should return a 201 and create a product                                 |
| get products           | given no product exists   | should return a 200 and an empty array                                   |
| get products           | given some products do exist  | should return a 200 and less than or equal to 'limit' number of products |
| Get a specific product | given no product with that id doesn't exist                                 | should return a 404  |
| Get a specific product | given a product with that id do exist                                       | should return a 200 and the product                                      |
| Update a product       | given user is not authenticated   | should return a 401 and not update the product                           |
| Update a product       | given authenticated user is not an admin                                    | should return a 403 and not update the product                           |
| Update a               | given authenticated user is an  | should return a 200 and  |

|                           |  |  |
|---------------------------|--|--|
| product                   | admin                                    | update the product                             |
| Delete a specific product | given user is not authenticated          | should return a 401 and not delete the product |
| Delete a specific product | given authenticated user is not an admin | should return a 403 and not delete the product |
| Delete a specific product | given authenticated user is an admin     | should return a 200 and delete the product     |

### 4.3 Test result

```

Test Suites: 5 passed, 5 total
Tests:      51 passed, 51 total
Snapshots:  0 total
Time:       29.115 s
Ran all test suites.
Done in 29.85s.

```

Figure 4.2: Test result from the automated test run

Figure 4.2 shows the result from an automated test run. It includes 5 test suites each representing a module. There is a total of 51 test cases, all of which are passing. Each test case represents a unique scenario.

## CHAPTER 5

### USER MANUAL

There are two types of users, shop owners, and customers. User manuals are prepared for both user types covering all the use cases.

#### 5.1 Shop owner

Shop owner is a user who owns a shop. Below is the process for a shop owner on how to use the application with screenshots.

##### 5.1.1 Register

The screenshot shows the registration page for Hyperstore. At the top left is the logo 'Hyperstore' and at the top right are links for 'Login' and 'Open a shop'. The registration form consists of several input fields: 'Name', 'Email', 'Password', 'Phone', and 'Address'. Below these fields is a purple 'Register' button. At the bottom of the form, there is a link for 'Log In' with the text 'Already have an account?'. Red arrows point from each input field and the 'Register' button to explanatory text: 'Enter your full name' for Name, 'Enter your valid email address' for Email, 'Enter a password' for Password, 'Enter a phone number' for Phone, 'Enter your address' for Address, and 'Finally click here to create an account' for the Register button. A final arrow points from the 'Log In' link to the text 'Click here if you already have an account'. The footer contains 'Powered By Hyperstore' and 'Copyright © Hyperstore 2022. All rights reserved.'

Figure 5.1: Registration

- Step 1: Enter full name
- Step 2: Enter email address
- Step 3: Enter password
- Step 4: Enter phone number
- Step 5: Enter address
- Step 6: Click on “Register”

### 5.1.2 Login

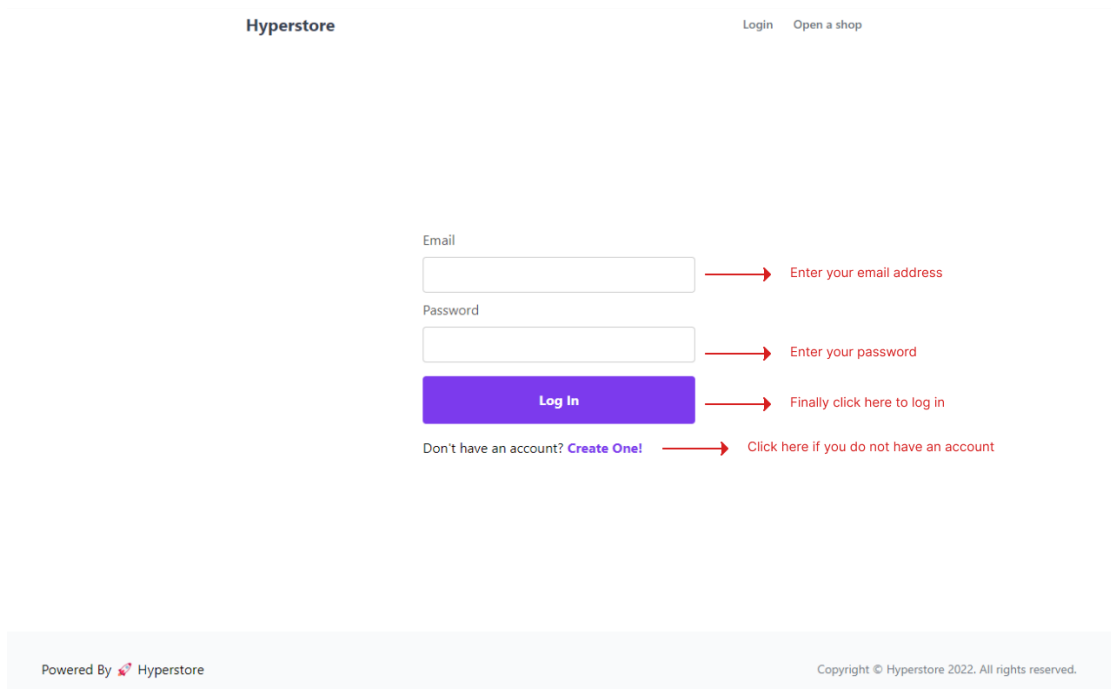


Figure 5.2: Login

- Step 1: Enter email address and password
- Step 2: Click on “Log In”

### 5.1.3 Open a shop

The screenshot shows the Hyperstore website interface for opening a shop. At the top left is the 'Hyperstore' logo, and at the top right are links for 'Login' and 'Open a shop'. The main heading is 'Get started today' in purple, followed by the text 'Open a store with hyperstore to and take your business online in minutes.' Below this is a form with four sections: 'Store name \*' with a text input field and a red arrow pointing to the placeholder 'Enter your shop or business name'; 'Description' with a text input field and a red arrow pointing to the placeholder 'Enter a short description (optional)'; 'Logo' with a 'Choose File' button, 'No file chosen' text, and a red arrow pointing to the instruction 'Upload your business logo (64x64 png for best results)'; and 'Brand color' with a black color swatch and a red arrow pointing to the instruction 'Pick the primary color for your shop'. At the bottom of the form is a large purple 'Create store' button with a red arrow pointing to the text 'Click here to finally create your shop'. The footer contains 'Powered By Hyperstore' on the left and 'Copyright © Hyperstore 2022. All rights reserved.' on the right.

Figure 5.3: Open a shop

Step 1: Enter shop name

Step 2: Enter shop description

Step 3: Upload logo

Step 4: Pick brand color

Step 5: Click on “Create store”

### 5.1.4 Admin dashboard (orders chart)

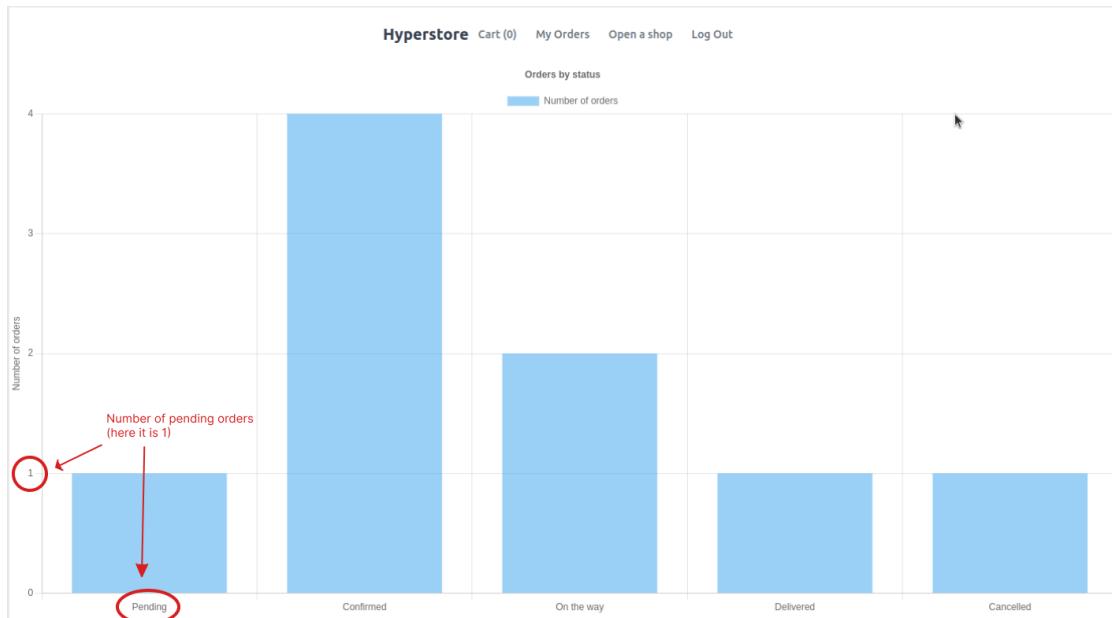


Figure 5.4: Admin dashboard (chart)

### 5.1.4b Admin dashboard

**Orders** → List of incoming orders (sorted by new to old)

| Order ID                 | Customer | Products                            | Total | Payment Method | Payment Status | Order Status | Delete |
|--------------------------|----------|-------------------------------------|-------|----------------|----------------|--------------|--------|
| 637d1e2e07742b66294698ca | user     | cat toy - medium<br>cat toy - small | 42    | card           | paid           | confirmed    | Delete |

Change order status    Delete this order

**Products** → List of all products

| Product ID               | Name      | Edit                     | Delete                       |
|--------------------------|-----------|--------------------------|------------------------------|
| 637fa5f86c9307294860deba | fish cake | Edit → Edit this product | Delete → Delete this product |
| 637d17fe07742b6629469843 | cat toy   | Edit                     | Delete                       |

Add a new Product → Click here to add a new product to your store

Powered By Hyperstore    Copyright © Hyperstore 2022. All rights reserved.

Figure: 5.5: Admin dashboard



Step 1: Click on the status dropdown menu

Step 2: Select a status

### 5.1.5 Create a product

The screenshot shows a form for creating a product. It includes the following elements:

- Name:** A text input field with a red arrow pointing to it and the text "Enter product name".
- Description:** A text area with a red arrow pointing to it and the text "Enter a short description of the product".
- Image:** A "Choose File" button (highlighted with a purple box), the text "No file chosen", and a red arrow pointing to it with the text "Upload an image of the product".
- Prices:** A red arrow pointing to the section with the text "Add different price based on the variant (you need to insert atleast one variant)". Below this are three input fields labeled "small", "medium", and "large".
- Create New Product:** A large purple button at the bottom.

Below the form, there is a red arrow pointing to the "Create New Product" button with the text "Finally click here to create the product".

Figure 5.6: Create a product

Step 1: Enter product name

Step 2: Enter description

Step 3: Upload product image

Step 4: Enter price

Step 5: Click on "Create New Product"

## 5.2 Customer

Customer is a user who does not own a shop but rather wants to buy products from a shop. Below is the process for a customer on how to use the application with screenshots.

### 5.2.1 Shop page

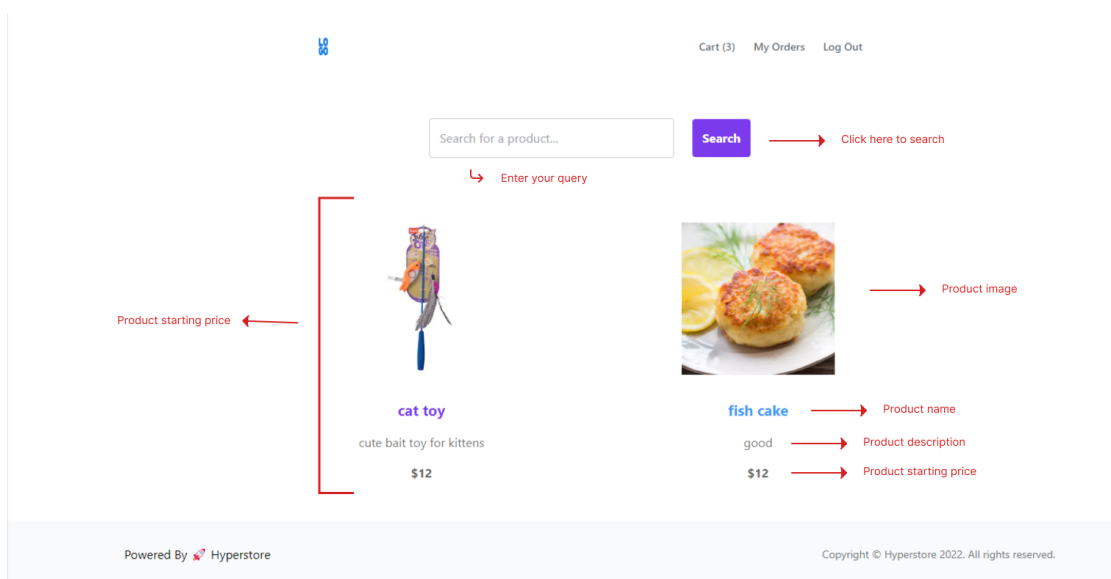


Figure 5.7: shop page (browse products)

Step 1: Enter search query

Step 2: Click on “Search”

## 5.2.2 View product

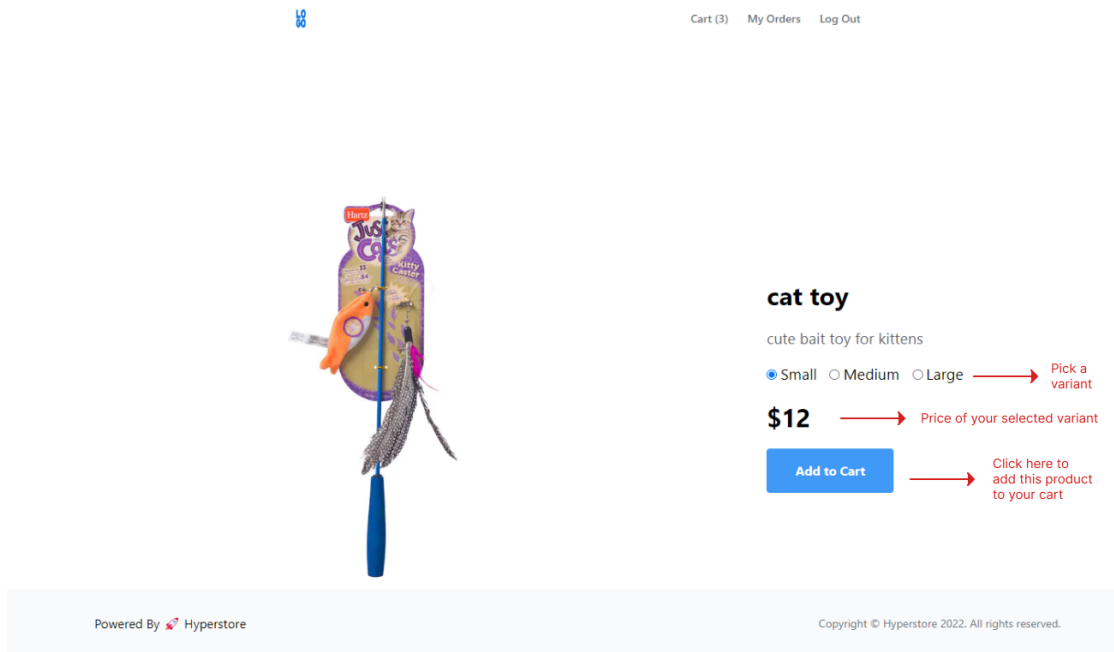


Figure 5.8: View product

Step 1: Select a variant

Step 2: Click on “Add to Cart”

## 5.2.3 Cart

The screenshot shows a shopping cart titled "Your Cart". At the top right, there are links for "Cart (3)", "My Orders", and "Log Out". The cart contains two items:

| Name    | Option | Quantity | Price | Remove                 |
|---------|--------|----------|-------|------------------------|
| cat toy | medium | - 2 +    | \$30  | <a href="#">Remove</a> |
| cat toy | small  | 1 +      | \$12  | <a href="#">Remove</a> |

Annotations include:

- A red arrow pointing to the minus sign in the quantity field of the first item: "Click here to remove an unit".
- A red arrow pointing to the plus sign in the quantity field of the first item: "Click here to add another unit".
- A red arrow pointing to the "Remove" button of the first item: "Click here to remove this product from your cart".
- A red arrow pointing to the "Checkout" button: "If you are happy with your cart content, click here to checkout".
- A red arrow pointing to the "Total: \$42" text: "Total: \$42".
- A red arrow pointing to the "Products currently in your cart" text: "Products currently in your cart".

Figure 5.9: Cart

Step 1: Increase/decrease product amount

Step 2: Click on "Checkout"

## 5.2.4 Payment

← pizza\_app TEST MODE

Product  
US\$42.00

Enter your credit card number →

Enter expiry date →

Enter name that is on your card →

Select the country where your card was issued at →

Enter a 3 digit code, usually found in the other side of your credit card →

Finally click here to complete the purchase →

Pay with card

Email owner@test.com

Card information

1234 1234 1234 1234 VISA

MM / YY CVC

Name on card

Country or region

Bangladesh

Save my info for secure 1-click checkout  
Pay faster on pizza\_app and thousands of sites.

Pay

Figure 5.10: Payment

Step 1: Enter credit card number

Step 2: Enter Expiry date

Step 3: Enter name on card

Step 4: Enter CVC

Step 5: Click on “Pay”

## **CHAPTER 6**

### **CONCLUSION**

#### **6.1 Conclusion**

I completed this project from start to finish all by myself within a tight deadline. It still needs a few works and features, and maintenance but the MVP is done and ready to be used by shop owners and customers alike.

#### **6.2 Limitations**

- The website is not mobile and SEO friendly.
- Email verification is not set up.
- Fixed layout for products.
- No way to run promotions or add coupon codes.

#### **6.3 Future improvements**

- Add email verification to reduce spam.
- Add the ability to customize the layout.
- Add a promotions page.
- Add support for coupon codes.
- Show related products when viewing a product.
- Show stock information.

## REFERENCES

- [1] Changchit, C., & Klaus, T. (2015). "An exploratory study on small business website creation and usage." *Journal of Electronic Commerce in Organizations (JECO)*, 13(1), 1-14.
- [2] Dushnitsky, Gary, and Bryan K. Stroube. "Low-code entrepreneurship: Shopify and the alternative path to growth." *Journal of Business Venturing Insights* 16 (2021): e00251.
- [3] Kim, Hana, Daeho Lee, and Min Ho Ryu. "An optimal strategic business model for small businesses using online platforms." *Sustainability* 10.3 (2018): 579.
- [4] Ihechikara Abba. "How to Use Tailwind CSS to Rapidly Develop Snazzy Websites." *Kinsta blog* (2022)
- [5] Software testing help. "SQL Vs NoSQL Exact Differences And Know When To Use NoSQL And SQL" (2022)
- [6] Abhishek Dubey. "Redis Best Practices and Performance Tuning" (2019)