**VIETNAM NATIONAL UNIVERSITY, HANOI**

**UNIVERSITY OF ENGINEERING AND TECHNOLOGY**

# INITIAL DESIGN DOCUMENT FOR ONLINE ECOMMERCE WEBSITE

**SEMESTER 232**

## GROUP 10


| | | |
|---|---|---|
| **Group members:** | **Phan Anh Tú** | **Student ID: 22028238** |
| | **Mai Anh Tuấn** | **Student ID: 22028144** |
| | **Nguyễn Ngọc Hưng** | **Student ID: 22028142** |
| | **Nguyễn Đức Phát** | **Student ID: 22028298** |
| | **Nguyễn Nhật Phong** | **Student ID: 22028272** |


| | |
|---|---|
| **Course:** | Software Engineering |
| **Course ID:** | INT2208E 23 |
| **Instructor:** | **Assoc. Prof. Đặng Đức Hạnh** |
| | **Bsc. Kiều Văn Tuyên** |

**HÀ NỘI – 2024**

# Initial Design Document

**for**

# Online Ecommerce Website

**Version 3.0 approved**

**Prepared by Group 10**

**University of Engineering and Technology**

**April 30th, 2024**

| Date | Document version | Description | Author |
|---|---|---|---|
| **24th, April, 2024** | 0.1 | **Construct the skeleton of the Content table (What we will write)** | Phan Anh Tú |
| **27th, April, 2024** | 1.0 | **Subsections details authors:**<br><br>**1. Introduction**<br><br>**2. System Architecture**<br><br>**3. Sequence Diagrams**<br><br>**4. Class Diagrams**<br><br>**5. User Interface Prototype**<br><br>**6. Database Diagrams**<br>6.1. EER Diagram<br>6.2. Details of tables in EER Diagram<br><br>**7. Data Structures and Algorithms**<br>**7**.1. Data Structures<br>*7.1.1. Arrays – Store products*<br>*7.1.2. Object – Represent each product*<br>7.2. Algorithms<br><br>**8. APIs and Dependencies** | <br><br>Phan Anh Tú<br><br>Phan Anh Tú<br><br>Phan Anh Tú<br><br>Nguyễn Ngọc Hưng<br><br>Nguyễn Đức Phát<br><br><br>Mai Anh Tuấn<br>Phan Anh Tú<br><br>Nguyễn Ngọc Hưng<br><br>Nguyễn Ngọc Hưng<br><br>Phan Anh Tú<br><br>Nguyễn Ngọc Hưng<br><br>Nguyễn Đức Phát |
| **3st , May, 2024** | 2.0 | **Subsections review:**<br><br>**4. Class Diagrams: Add Figure indices**<br><br>**5. User Interface Prototype: Add Figure indices**<br><br>**Reformat document** | <br><br>Phan Anh Tú<br><br>Phan Anh Tú<br><br>Phan Anh Tú |
| **8th , May, 2024** | 2.1 | **Subsections review:**<br><br>**3. Sequence Diagrams: Review Sequence diagrams**<br><br>**4. Class Diagrams: Review Class diagrams** | <br><br>Phan Anh Tú<br><br>Phan Anh Tú |
| **9th , May, 2024** | 3.0 | **Ultimate review** | Phan Anh Tú |

# Table of Contents


1. Introduction ..... 9
   1.1. Motivation ..... 9
   1.2. Purpose ..... 9
   1.3. Scope of users ..... 9
   1.4. References ..... 10
2. System Architecture ..... 11
   2.1. Important Abstractions ..... 11
      2.1.1. Abstraction List ..... 11
      2.1.2. Abstraction Details ..... 11
   2.2. System Architecture ..... 13
      2.2.1. Architectural Model: Three Layered Package Diagram – MVC Structure ..... 13
      2.2.2. Definition of Packages ..... 14
3. Sequence Diagrams ..... 16
   3.1. Account Registration ..... 16
   3.2. Account Login ..... 17
   3.3. Search items ..... 17
   3.4. Add item to cart ..... 17
   3.5. View cart ..... 18
   3.6. Proceed to buy ..... 18
   3.7. Upload listings ..... 19
4. Class Diagrams ..... 20
   4.1. Account Registration ..... 20
   4.2. Account Login ..... 20
   4.3. Search items ..... 20
   4.4. Add item to cart ..... 21
   4.5. View cart ..... 21
   4.6. Proceed to buy ..... 22
   4.7. Upload listings ..... 22
5. User Interface Prototype ..... 24
   5.1. Register an account ..... 24
   5.2. Login ..... 25
   5.3. Edit account information ..... 26
   5.4. Search items ..... 27
   5.5. View items ..... 28
   5.6. Add items to cart ..... 28
   5.7. View cart ..... 29

5.8. Buy items ..... 30
5.9. Shipping tracking ..... 31
5.10. Order cancellation ..... 31
5.11. Order returns ..... 32
5.12. Order confirmation and reviews ..... 32
5.13. Upload listings ..... 33
5.14. Edit items ..... 34
5.15. Items classification ..... 35
5.16. View order from customers ..... 36
6. Database Diagram ..... 36
6.1. EER Diagram ..... 36
6.2. Details of the tables in EER Diagram ..... 38
6.2.1. User table ..... 38
6.2.2. Product table ..... 38
6.2.3. Orders table ..... 39
6.2.4. Orders_detail table ..... 39
6.2.5. Type table ..... 40
6.2.6. Carts table ..... 40
6.2.7. Reviews table ..... 40
6.2.8. Posses_product table ..... 40
6.2.9. Image_product table ..... 41
7. Data Structures and Algorithms ..... 41
7.1. Data Structures ..... 41
7.1.1. Arrays – Store products ..... 41
7.1.2. Object – Represent each product ..... 42
7.2. Algorithms ..... 44
7.2.1. Heap Sort - Arrange items according to the user's filter ..... 44
8. APIs and Dependencies ..... 45
8.1. API ..... 45
8.2. Dependencies ..... 45
8.2.1. React ..... 45
8.2.2. React-dom ..... 45
8.2.3. Vite ..... 45
8.2.4. Tailwind ..... 45
8.2.5. Express ..... 45
8.2.6. MySql ..... 45
8.2.7. Bcrypt ..... 46

# Table of Figures

Figure 1. List of important abstractions ..... 11
Figure 2. Architectural Model ..... 13
Figure 3. Account Registration Sequence Diagram ..... 16
Figure 4. Account Login Sequence Diagram ..... 17
Figure 5. Search items Sequence Diagram ..... 17
Figure 6. Add item to cart Sequence Diagram ..... 18
Figure 7. View cart Sequence Diagram ..... 18
Figure 8. Proceed to buy Sequence Diagram ..... 19
Figure 9. Upload Listings Sequence Diagram ..... 19
Figure 10. Account Registration Class Diagram ..... 20
Figure 11. Account Login Class Diagram ..... 20
Figure 12. Search items Class Diagram ..... 21
Figure 13. Add item to cart Class Diagram ..... 21
Figure 14. View cart Class Diagram ..... 22
Figure 15. Proceed to buy Class Diagram ..... 22
Figure 16. Upload listings Class Diagram ..... 23
Figure 17. Register an account ..... 24
Figure 18. Login Form ..... 25
Figure 19. Edit Account Information ..... 26
Figure 20. Search items ..... 27
Figure 21. View items ..... 28
Figure 22. View cart ..... 29
Figure 23. Buy items ..... 30
Figure 24. Shipping tracking ..... 31
Figure 25. Order Cancellation ..... 31
Figure 26. Order return ..... 32
Figure 27. Upload listings ..... 33
Figure 28. Edit items ..... 34
Figure 29. Items classification ..... 35
Figure 30. View order from customers ..... 36
Figure 31. EER Diagram of the system ..... 37

# Table of Tables

Table 1. Abstraction Details ..... 11
Table 2. Definition of Packages ..... 14
Table 3. EER Diagram - User table ..... 38
Table 4. EER Diagram - Product table ..... 38
Table 5. EER Diagram - Orders table ..... 39
Table 6. EER- Diagram - Order_detail table ..... 39
Table 7. EER Diagram - Type table ..... 40
Table 8. EER Diagram - Carts table ..... 40
Table 9. EER- Diagram - Reviews table ..... 40
Table 10. EER Diagram - Possess_product table ..... 41
Table 11. EER Diagram - Image_product table ..... 41

# 1. Introduction

## *1.1. Motivation*

In today's competitive and convenience-driven society, consumers prefer the ease of shopping remotely rather than venturing out. This shift is evident, with Southeast Asia recording the highest mobile Internet usage in the region, averaging 3.6 hours per day per person, according to Google and Temasek's research.

Acknowledging this trend, our team proposes the development of an online ecommerce platform to cater to this demand. Key features include robust search functionality for easy product discovery, high-quality images, and detailed descriptions to aid decision-making. Additionally, seamless product management and payment processes, along with review and rating features, will enhance user experience and credibility.

User account registration and management will allow for order tracking and purchase history access. Statistical analysis tools will provide insights into customer behavior for continuous improvement, while diverse payment methods will offer flexibility. Furthermore, customer support features will be integrated to address inquiries promptly.

In line with regulations, our platform will prioritize security and privacy to ensure user data protection. Through these efforts, our ecommerce website, Market Swift, aims to provide users with a convenient, secure, and efficient online shopping experience.

## *1.2. Purpose*

The purpose of this software requirement specification is to offer a comprehensive outline of the necessities for an online ecommerce website. This document delineates the primary use cases for web customers engaging in online purchases, encompassing both the client and seller components.

The online ecommerce website serves as a virtual marketplace for a diverse array of goods, facilitating global transactions. It operates as an internet-based enterprise with an extensive inventory spanning categories such as books, electronics, groceries, and more.

Sellers utilize this platform to broaden their reach to a global audience, providing enhanced flexibility and market access. The convenience of online shopping has surged in popularity over the past decade, allowing customers to easily explore various products, brands, and price points from the comfort of their homes. This trend has solidified the online shopping website's position as a dominant force in the digital marketplace.

## *1.3. Scope of users*

The e-commerce platform functions as a hub for the sale of diverse goods and facilitates secure transactions between sellers and customers worldwide. Operating as a virtual entity, it offers customers an extensive selection of products for online browsing and purchase, while providing sellers with a platform to showcase and sell their products. Its key benefits include the convenience of remote shopping, the ability to compare prices, brands, and customer feedback, and the opportunity for retailers to expand their reach to a global audience.

The platform features a robust database that stores customer information and purchase histories. Its review section promotes transparency by showcasing product performance and user experiences through ratings and detailed reviews, including optional product images, fostering trust and community engagement.

Ensuring secure transactions and timely product delivery are core components. Driving sales growth is prioritized, necessitating the implementation of various strategies such as cost-effective management, enhancing customer loyalty and satisfaction, and harnessing internet resources for technological advancements, marketing, and business expansion. Continuous research and development efforts, particularly in logistics, are essential for sustained business development. Exploring new markets and investing in innovative ventures are also viable strategies.

Ultimately, the goal of the e-commerce website is to offer a seamless and user-friendly platform that caters to the needs of both customers and sellers, enabling convenient product selection from a wide range of offerings.

## *1.4. References*


[1] Ian Sommerville, *Software Engineering 10th Edition*, Pearson, 2016

[2] Data Structures in JavaScript with examples - freeCodeCamp.org

[3] Use Case Analysis Document - Students' document in previous courses (Resourses in MSTeams)

[4] Geeksforgeeks

[5] *Lectures during courses*, Dang Duc Hanh

# 2. System Architecture

## *2.1. Important Abstractions*

### *2.1.1. Abstraction List*

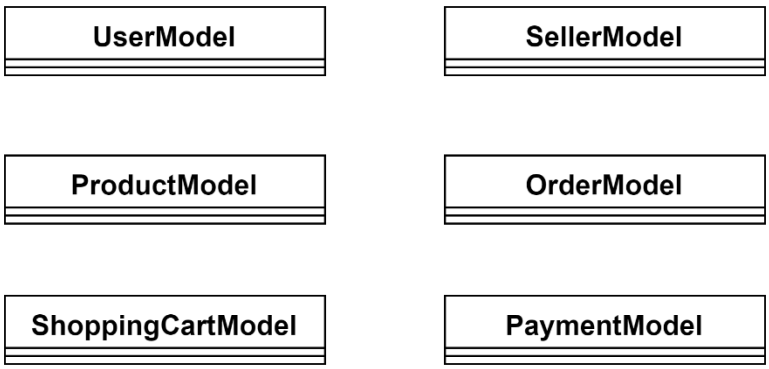


*Figure 1. List of important abstractions*

### *2.1.2. Abstraction Details*

*Table 1. Abstraction Details*

| No. | Name | Function |
|---|---|---|
| **1** | UserModel | User accounts represent individual customers who interact with the ecommerce platform. These abstractions store essential information about users, including personal details, purchase history, preferences, and authentication credentials. Upon discovering a product of interest, customers can delve into its detailed description, examining its features, specifications, and pricing. They may explore various product images from different angles to gain a comprehensive understanding of its appearance and functionality. Customers can also utilize filters and sorting options to refine their search results, narrowing down their options based on criteria such as price range, brand, ratings, and availability. Additionally, customers may read reviews and ratings from other buyers to gauge the product's quality, reliability, and suitability for their needs. Once satisfied with their selection, customers can add the product to their shopping cart, proceeding to checkout for secure payment processing and order confirmation. |
| **2** | SellerModel | Seller Models encapsulate information about individual merchants or businesses operating within the marketplace. Each profile includes details such as seller name, logo, description, contact information, ratings, reviews, and product listings. Sellers can upload listings, includes details such as product name, description, images, price, variations, and availability. Sellers upload |

| | | |
|---|---|---|
| | | and manage their listings, ensuring accurate representation and timely updates. Seller profiles enable customers to evaluate the credibility and reputation of sellers before making purchases. |
| **3** | ProductModel | Products are fundamental abstractions representing items available for sale on the ecommerce platform. Each product has attributes such as name, description, price, availability, images, and metadata. Products may also belong to categories or have variations (such as size or color). |
| **4** | OrderModel | Orders represent completed transactions initiated by users. This abstraction captures details such as the list of purchased items, shipping information, payment method, order status, and timestamps. Order management functionalities, including processing, fulfillment, and tracking, are typically associated with this abstraction. |
| **5** | ShoppingCartModel | The shopping cart abstraction holds items selected by the user for purchase during their browsing session. It tracks quantities, prices, and other relevant details before the user proceeds to checkout. The shopping cart may exist both on the client-side (in the user's browser) and server-side. |
| **6** | PaymentModel | The payment abstraction facilitates transactions by handling payment-related operations securely. It encapsulates payment gateways, methods, and processing logic. Payment abstractions ensure the security and integrity of financial transactions, including authorization, validation, and settlement. |

## 2.2. *System Architecture*

### 2.2.1. *Architectural Model: Three Layered Package Diagram – MVC Structure*

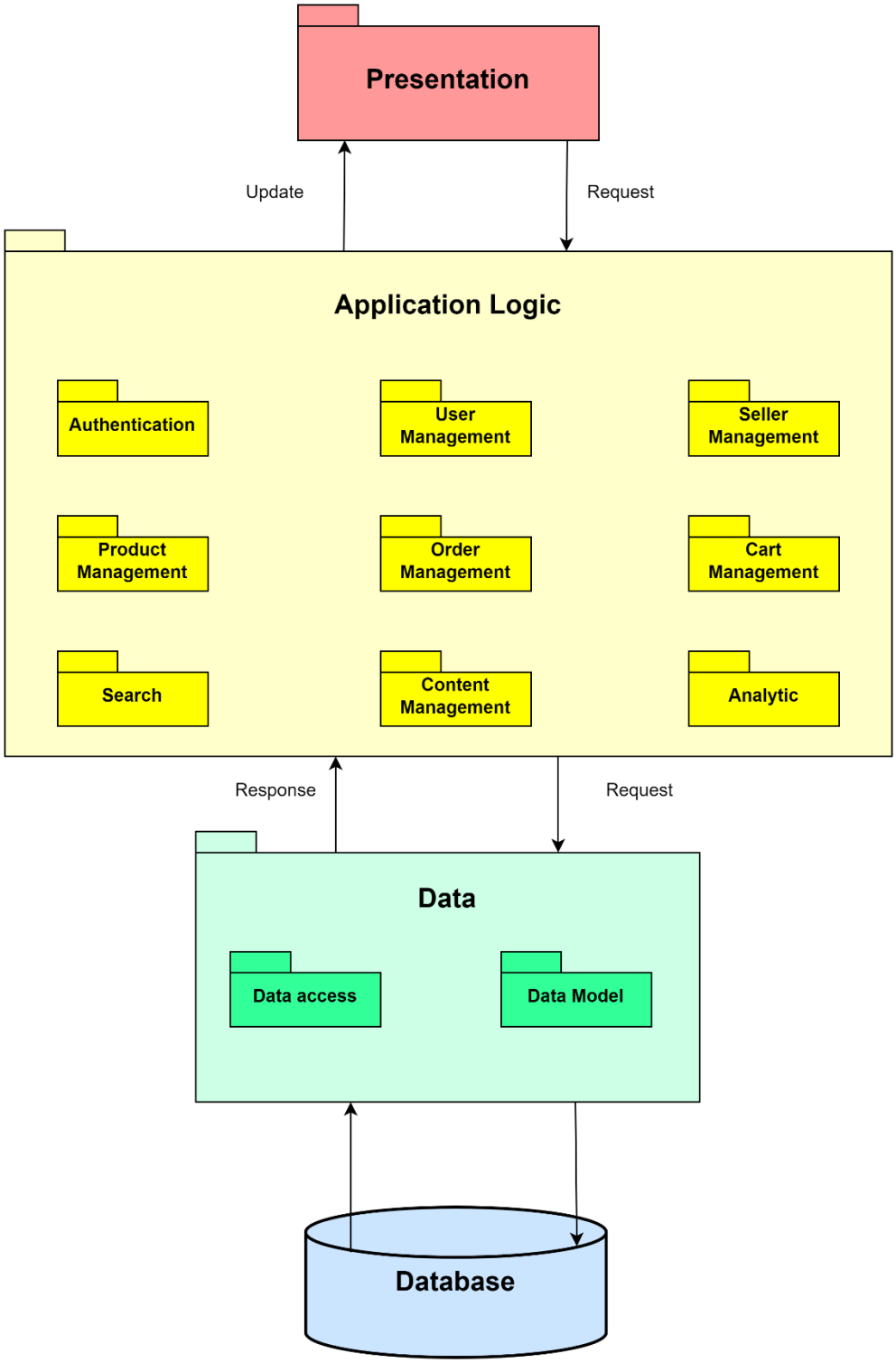


*Figure 2. Architectural Model*

Figure 2 describes the high-level organization of the system. The system is built with a three-tier architecture consisting of three layers: *Presentation Layer*, *Logic Layer*, and *Data Layer*.

- **Presentation Layer**

  This layer's main responsibility is to interact with users. It consists of interface components (such as winform, webform, etc.) and performs tasks such as data input, data display, and data validation before calling the Business Logic Layer (BLL).

- **Logic Layer**

  This layer is divided into two main tasks:

  **1.** It responds to data manipulation requests from the GUI layer, processes the data from the Presentation Layer before passing it down to the Data Layer, and saves it to the database management system (DBMS).

  **2.** It checks data constraints, integrity, and validity, performs calculations, and handles business logic requests before returning the results to the Presentation Layer.

- **Data Layer**

  This layer is responsible for communicating with the database management system (DBMS), such as performing tasks related to storing and querying data (searching, adding, deleting, updating, etc.).

### *2.2.2. Definition of Packages*

*Table 2. Definition of Packages*

| No. | Name | Description |
|---|---|---|
| **1** | **Authentication** | The Authentication package plays a pivotal role in ensuring the security and integrity of user interactions within the ecommerce platform. In addition to managing user authentication and authorization processes, it also oversees user signup and sign-in functionalities. This includes validating user credentials during the signup process, such as verifying email addresses or phone numbers, and securely storing login information. Furthermore, it may maintains user sessions and permissions, enabling secure access to the platform's features and resources while safeguarding against unauthorized access. By integrating robust authentication mechanisms, the package enhances the platform's security posture and ensures a trustworthy environment for users to engage in ecommerce activities. |
| **2** | **User Management** | The User Management package assumes a central role in fostering a secure and compliant environment within the ecommerce platform. Beyond conventional user account management tasks, it delves into the intricacies of user behavior to uphold the platform's rules of engagement. Moreover, it administers account-related activities such as password resets and account deactivations, safeguarding against unauthorized access and misuse. Furthermore, through sophisticated role-based permissions, it delineates distinct access levels for users based on their roles (e.g., customers, sellers), bolstering |

| | | |
|---|---|---|
| | | platform security and user trust. By actively monitoring user behavior and enforcing compliance with platform policies, the User Management package plays a pivotal role in cultivating a resilient and inclusive ecommerce ecosystem. |
| **3** | **Seller Management** | The Seller Management package within the ecommerce platform serves as a pivotal hub, orchestrating various aspects of sellers' operations with meticulous attention to detail. From streamlining registration and verification processes to empowering sellers with intuitive tools for product listing management, it ensures a seamless and engaging experience. Moreover, it facilitates performance tracking and optimization by providing insightful analytics, fostering continuous improvement and growth. In essence, the Seller Management package embodies empowerment, enabling sellers to thrive and excel within the dynamic ecommerce landscape. |
| **4** | **Product Management** | The Product Management package is responsible for managing the lifecycle of products within the ecommerce platform. It includes functionalities for product catalog management, such as adding new products, updating product information, managing product categories and attributes, and handling product variations (e.g., sizes, colors). Product Management also encompasses inventory management, pricing strategies, and product recommendation systems. It reserved for sellers. |
| **5** | **Order Management** | The Order Management package handles the processing and fulfillment of orders placed by customers. It includes functionalities for order creation, order tracking, inventory reservation, order status updates, and communication with customers regarding order fulfillment and delivery. Additionally, Order Management manages returns, exchanges, and refunds processes to ensure customer satisfaction. |
| **6** | **Cart Management** | The Cart Management package manages the shopping cart functionality for customers, allowing users to add, remove, and modify items before completing their purchases. It maintains the state of users' carts across sessions, calculates order totals, applies discounts or promotions, and handles cart abandonment scenarios. Cart Management ensures a seamless and intuitive shopping experience for customers. |
| **7** | **Search** | The Search package provides robust search functionalities to help users discover products efficiently. It includes features such as keyword search, faceted search, filtering options (by category, price range, etc.), and sorting mechanisms (by relevance, price, rating, etc.). Search utilizes indexing and search algorithms to deliver accurate and relevant search results to users. |
| **8** | **Content Management** | The Content Management package oversees the creation, organization, and presentation of non-product-related content on the platform. This includes managing static pages, blog posts, promotional banners, FAQs, and other informational or marketing content. Content |

| | | |
|---|---|---|
| | | Management systems often include features for content versioning, scheduling, localization, and SEO optimization. |
| **9** | **Analytics** | The Analytics package collects, analyzes, and interprets data to gain insights into user behavior, market trends, and platform performance. It tracks key metrics such as website traffic, user engagement, conversion rates, and sales performance. Analytics tools and dashboards enable stakeholders to make data-driven decisions, optimize strategies, and improve the overall effectiveness of the ecommerce platform. |
| **10** | **Data Access** | The Data Access package provides interfaces and methods for accessing and manipulating data stored in the underlying database management system (DBMS). It abstracts database interactions, handles database connections, executes database queries, and manages transactions. Data Access ensures efficient and secure access to data while maintaining data integrity and consistency. |
| **11** | **Data Model** | The Data Model package defines the structure and relationships of the data entities within the ecommerce system. It includes entity-relationship diagrams, database schemas, and data models that represent entities such as users, products, orders, and transactions. The Data Model serves as a blueprint for organizing and storing data efficiently, supporting the system's functionalities and business requirements. |

# 3. Sequence Diagrams

## *3.1. Account Registration*

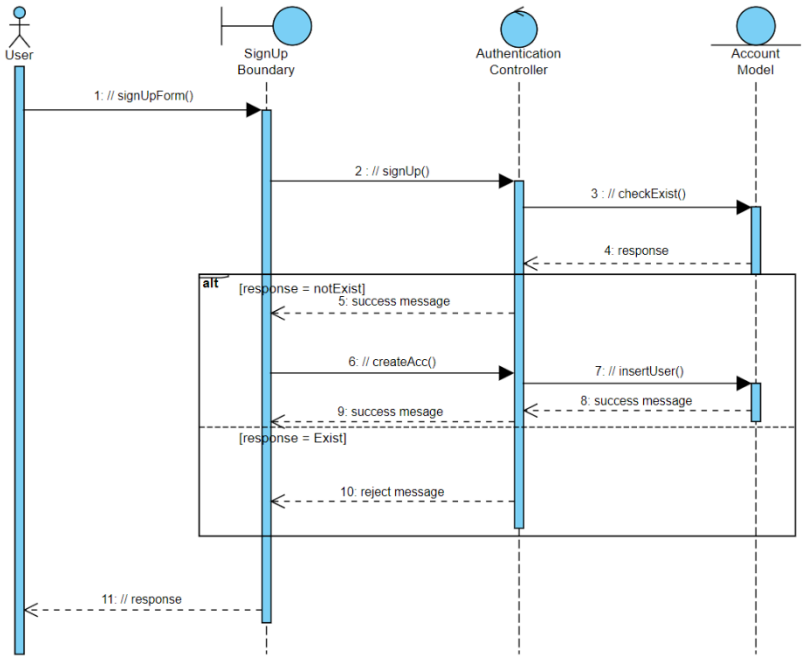


*Figure 3. Account Registration Sequence Diagram*

## 3.2. Account Login

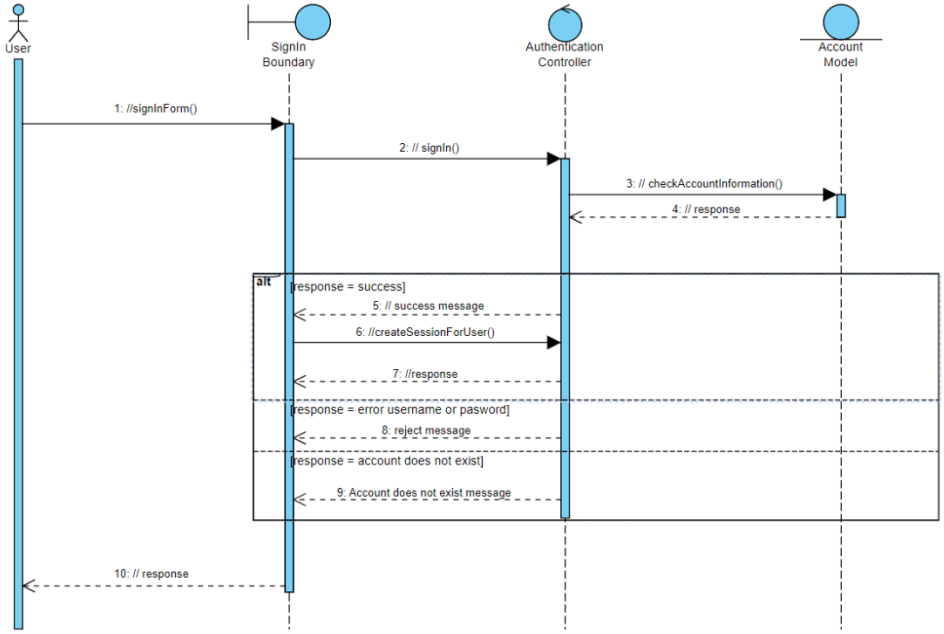


*Figure 4. Account Login Sequence Diagram*

## 3.3. Search items

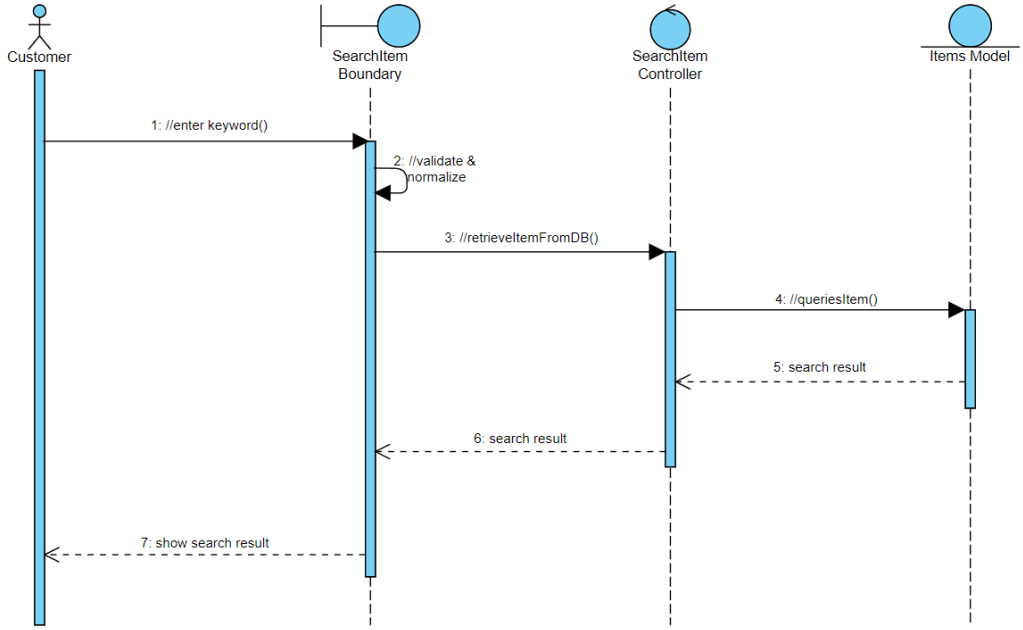


*Figure 5. Search items Sequence Diagram*

## 3.4. Add item to cart

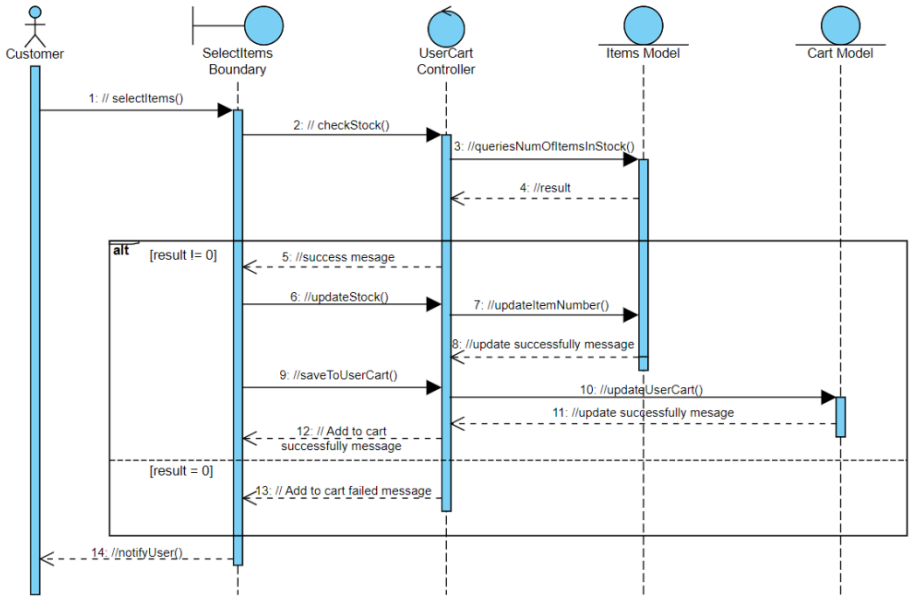


*Figure 6. Add item to cart Sequence Diagram*

## 3.5. View cart

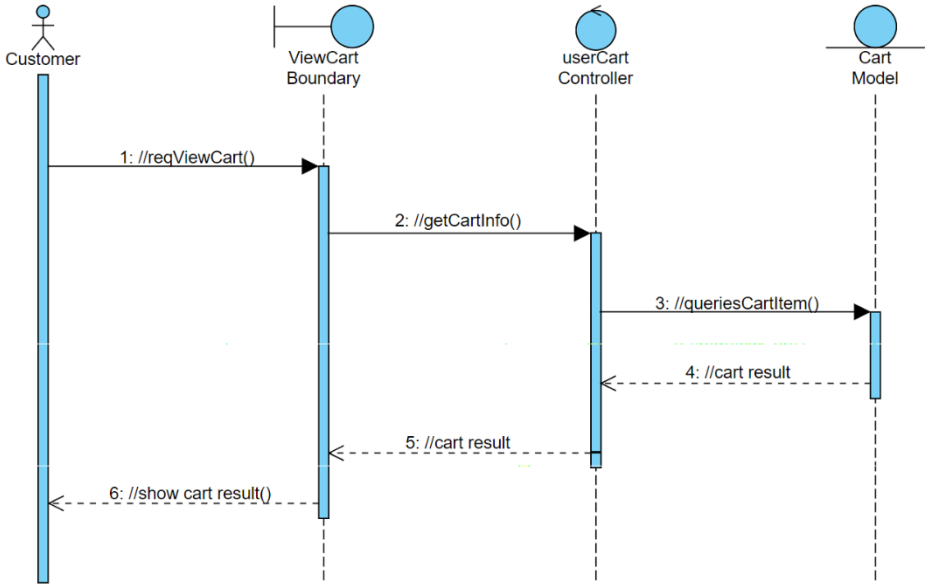


*Figure 7. View cart Sequence Diagram*

## 3.6. Proceed to buy

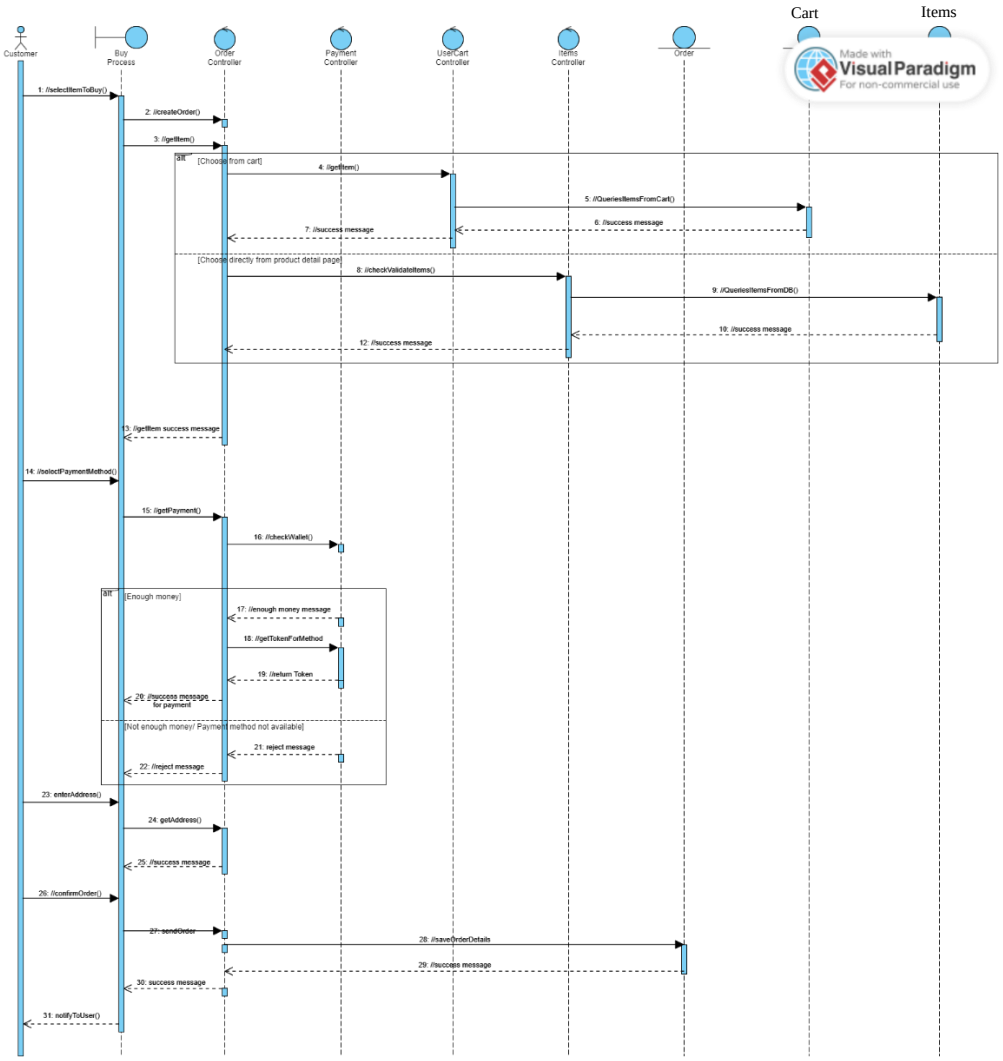


*Figure 8. Proceed to buy Sequence Diagram*

## *3.7. Upload listings*

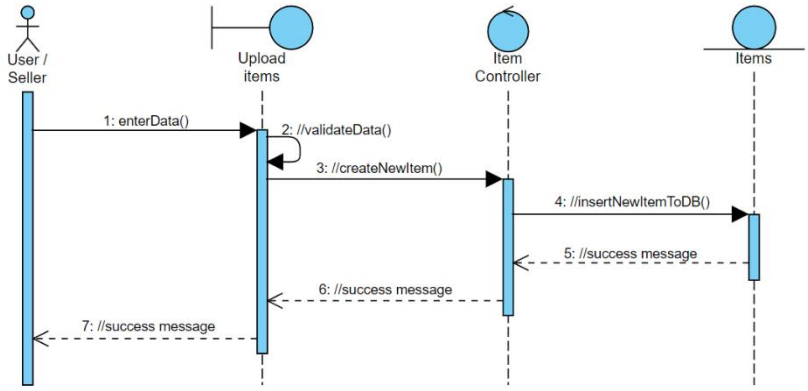


*Figure 9. Upload Listings Sequence Diagram*

# 4. Class Diagrams

## *4.1. Account Registration*

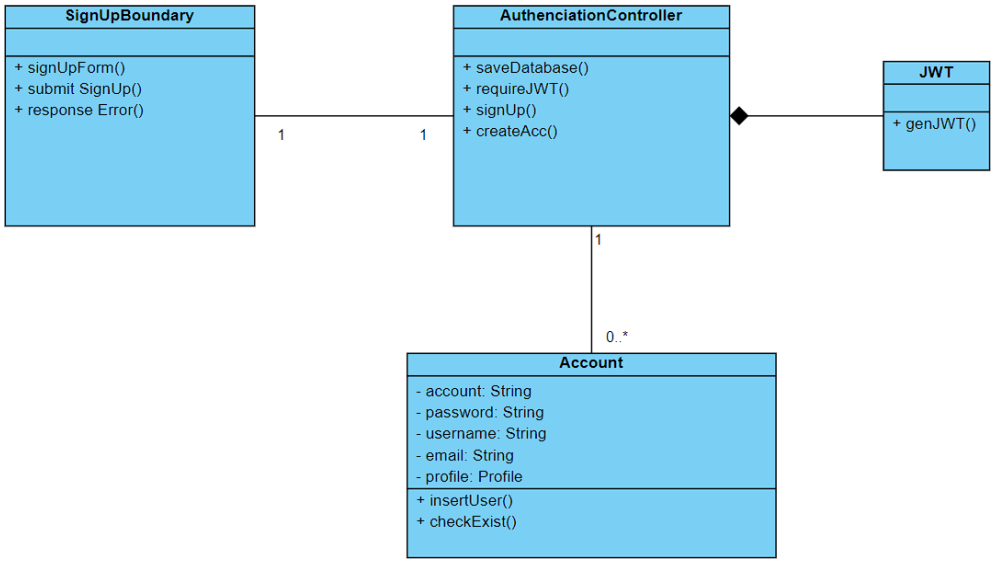


*Figure 10. Account Registration Class Diagram*

## *4.2. Account Login*

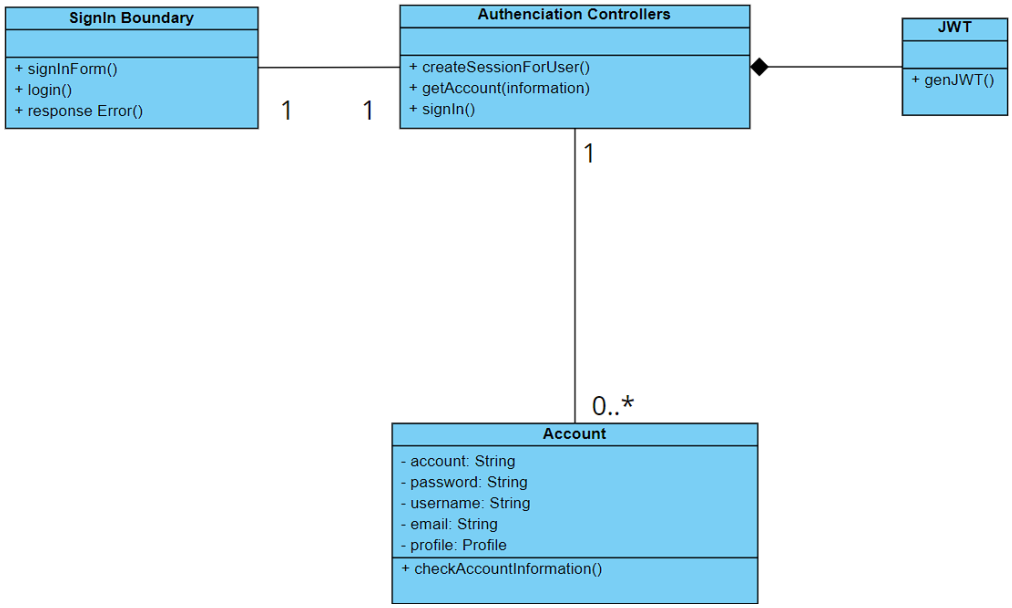


*Figure 11. Account Login Class Diagram*

## *4.3. Search items*

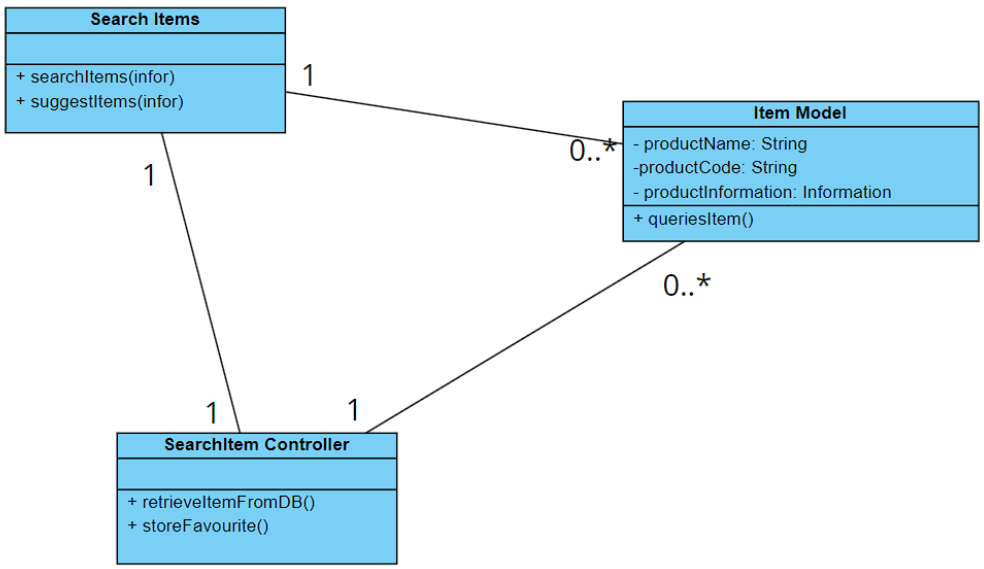


*Figure 12. Search items Class Diagram*

## 4.4. *Add item to cart*

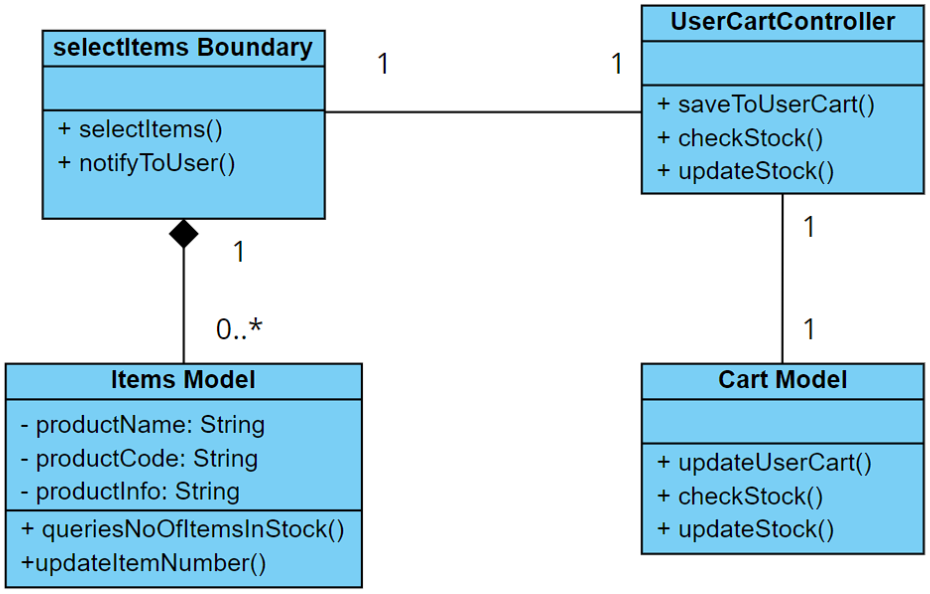


*Figure 13. Add item to cart Class Diagram*

## 4.5. *View cart*

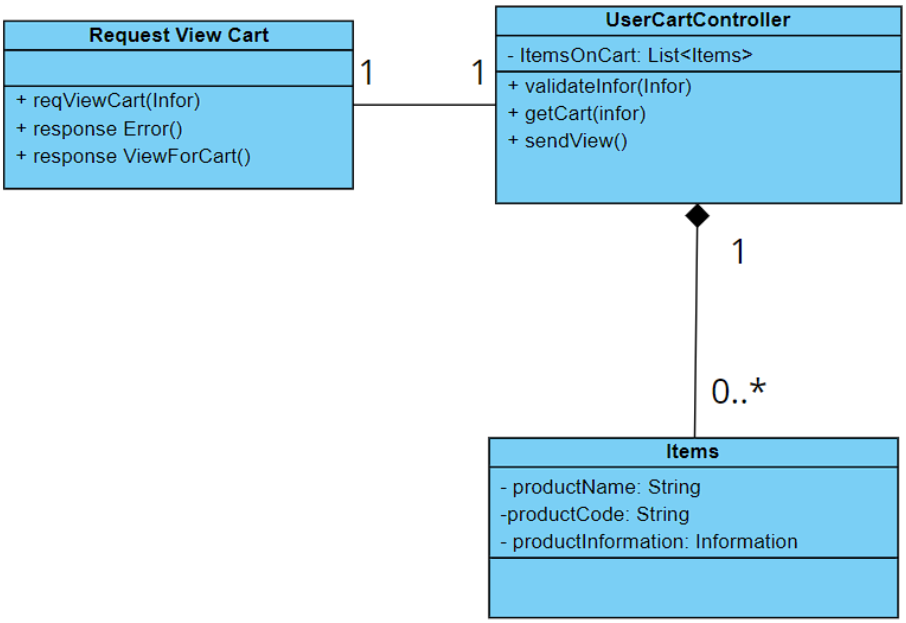


*Figure 14. View cart Class Diagram*

## *4.6. Proceed to buy*

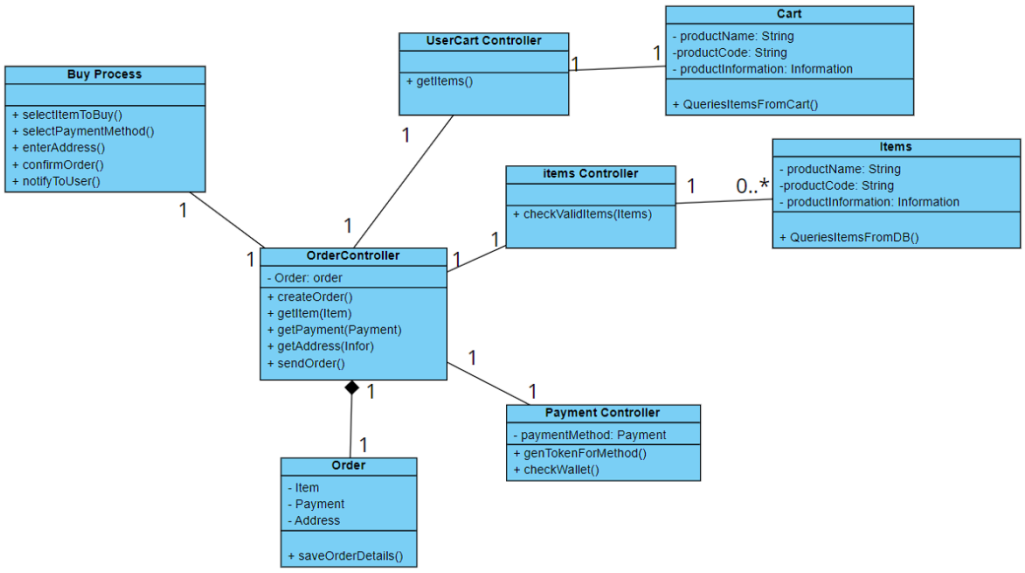


*Figure 15. Proceed to buy Class Diagram*

## *4.7. Upload listings*

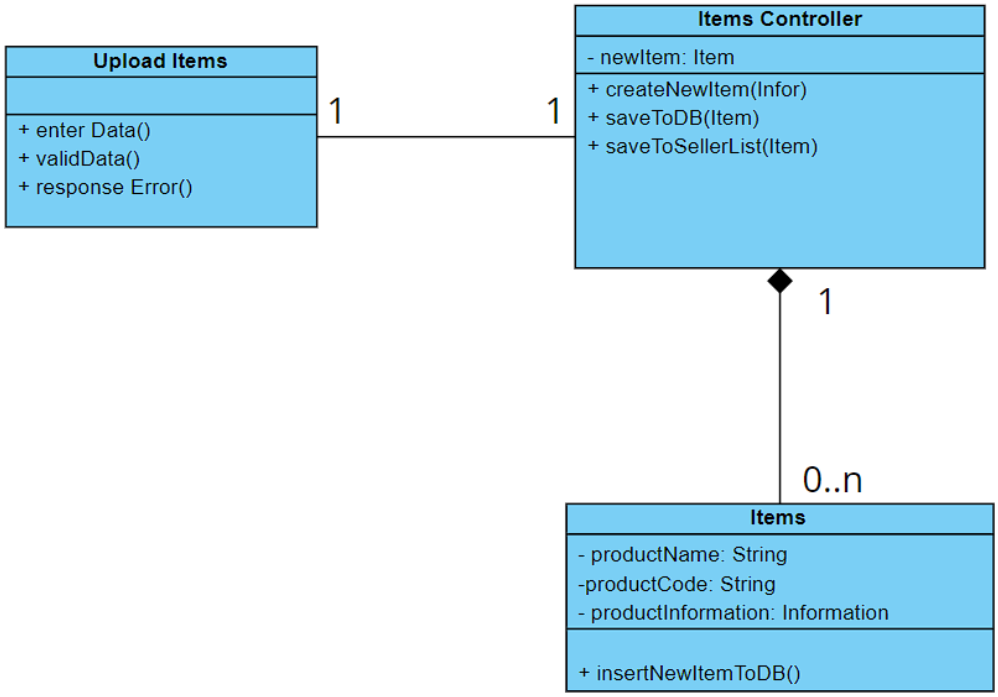


*Figure 16. Upload listings Class Diagram*

# 5. User Interface Prototype

## *5.1. Register an account*

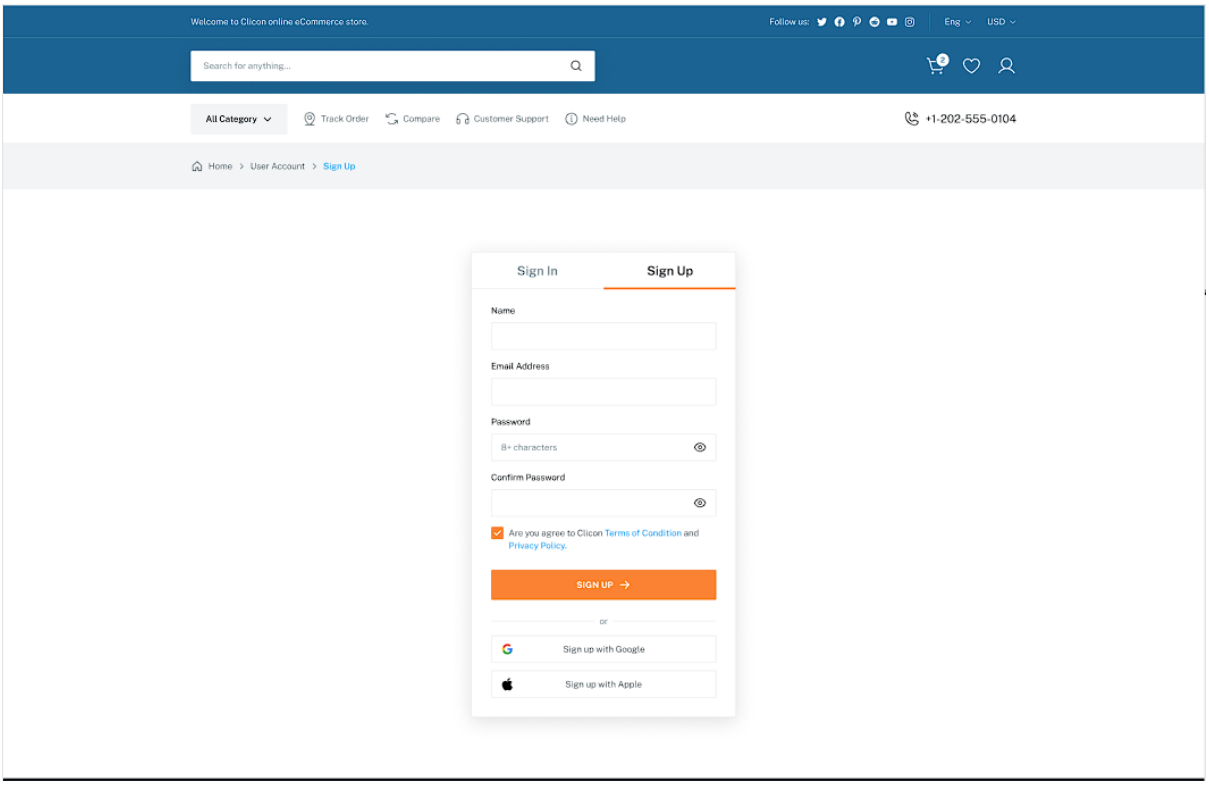


*Figure 17. Register an account*

## *5.2. Login*

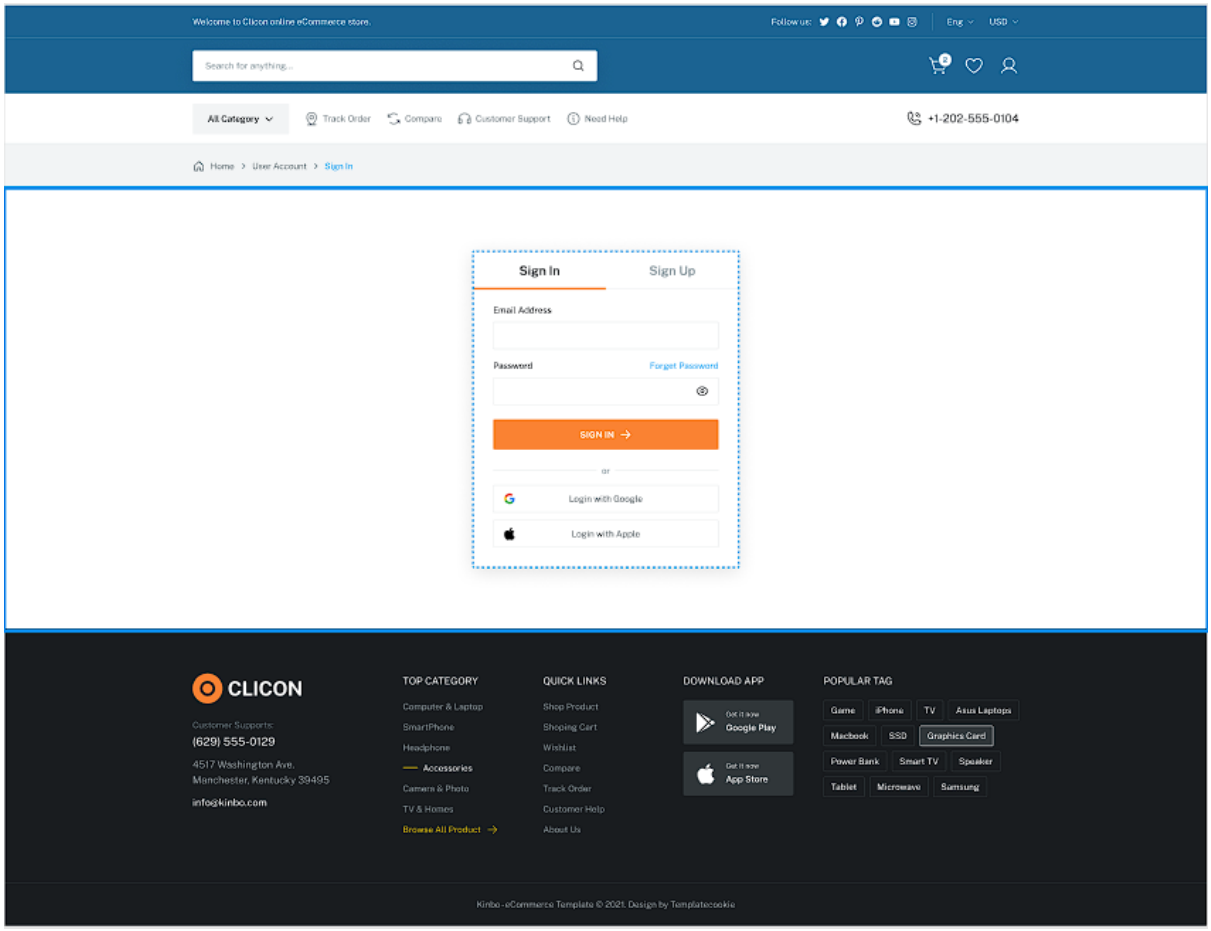


*Figure 18. Login Form*

## *5.3. Edit account information*

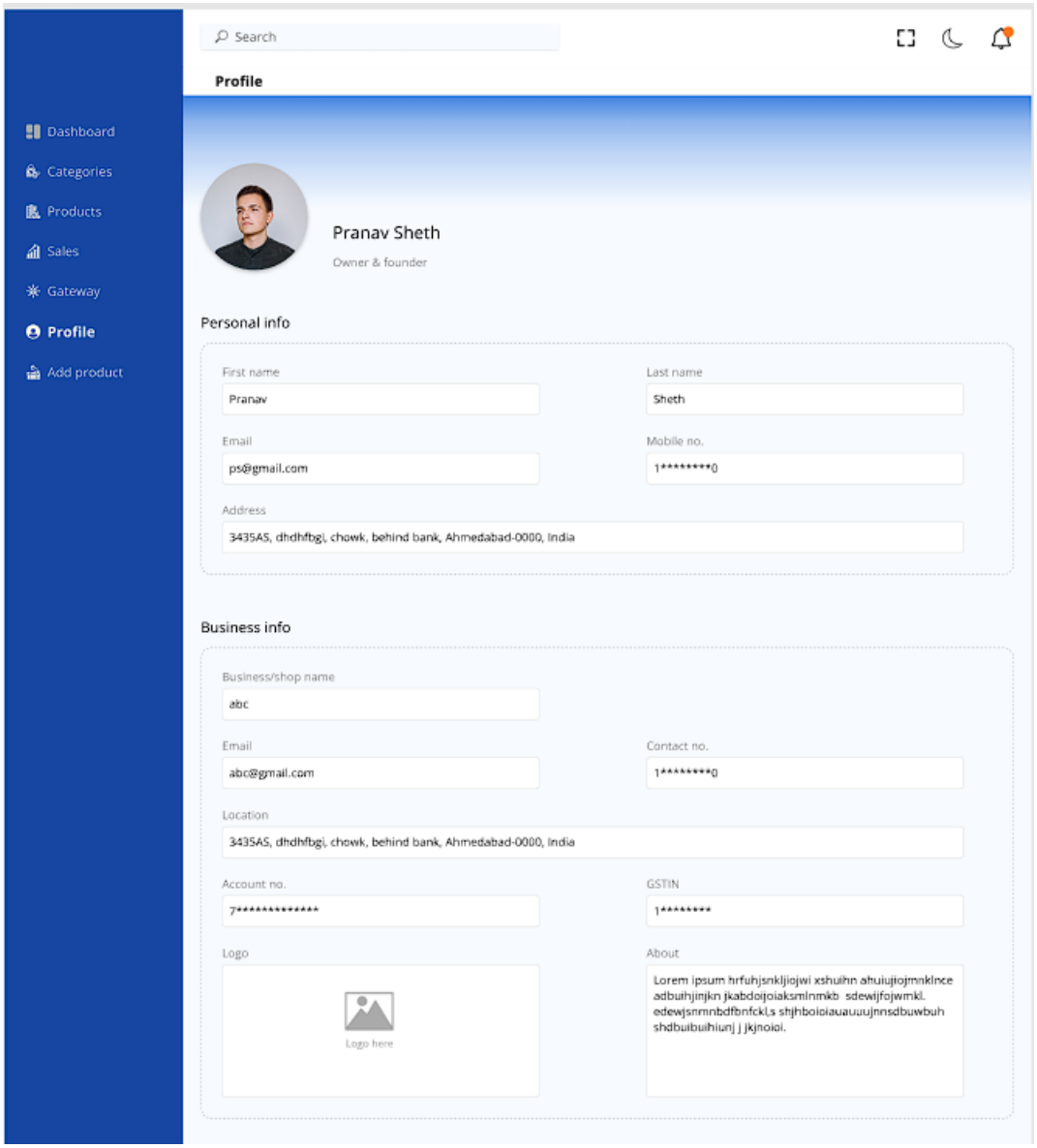


*Figure 19. Edit Account Information*

## *5.4. Search items*

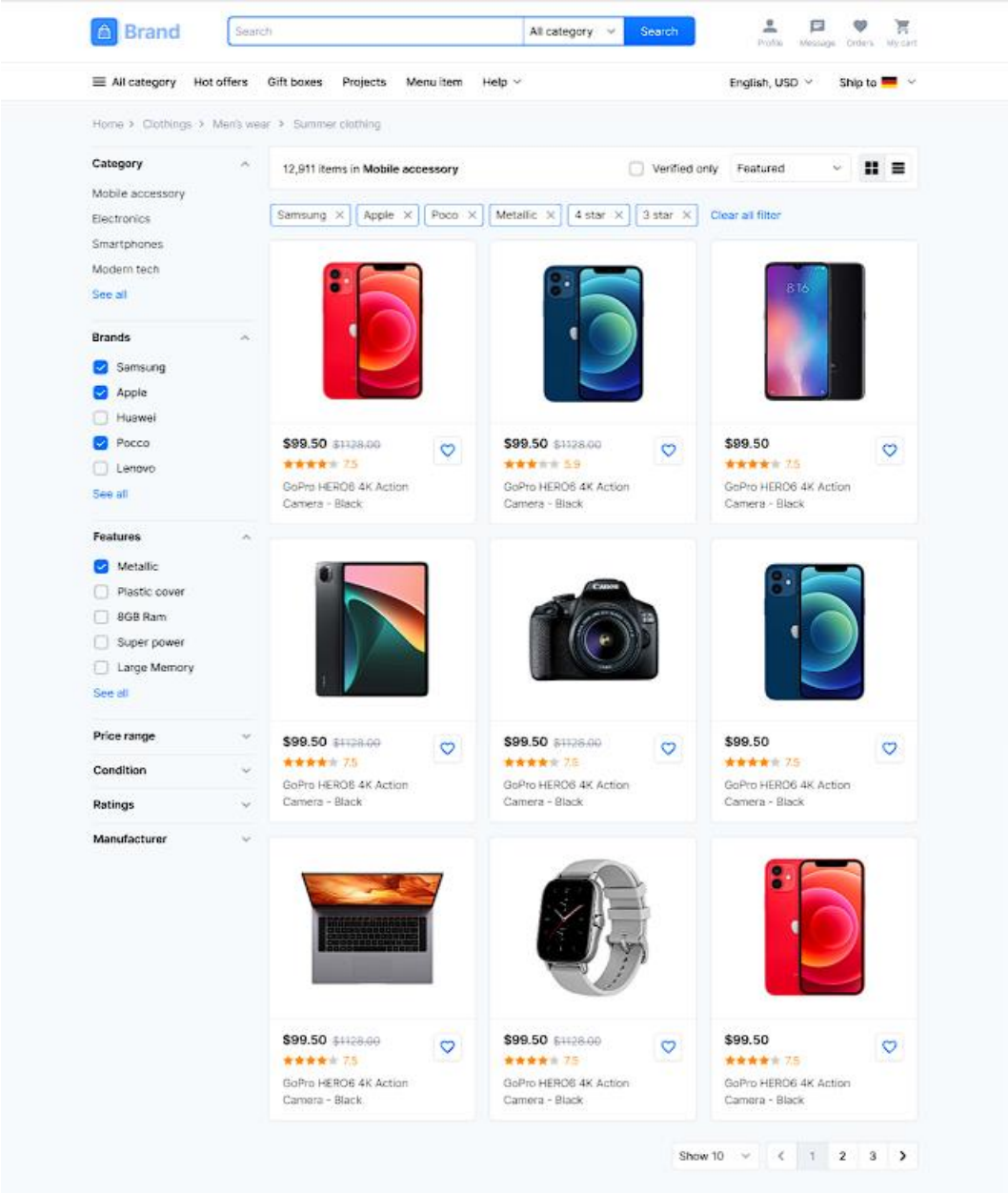


*Figure 20. Search items*

## 5.5. *View items*

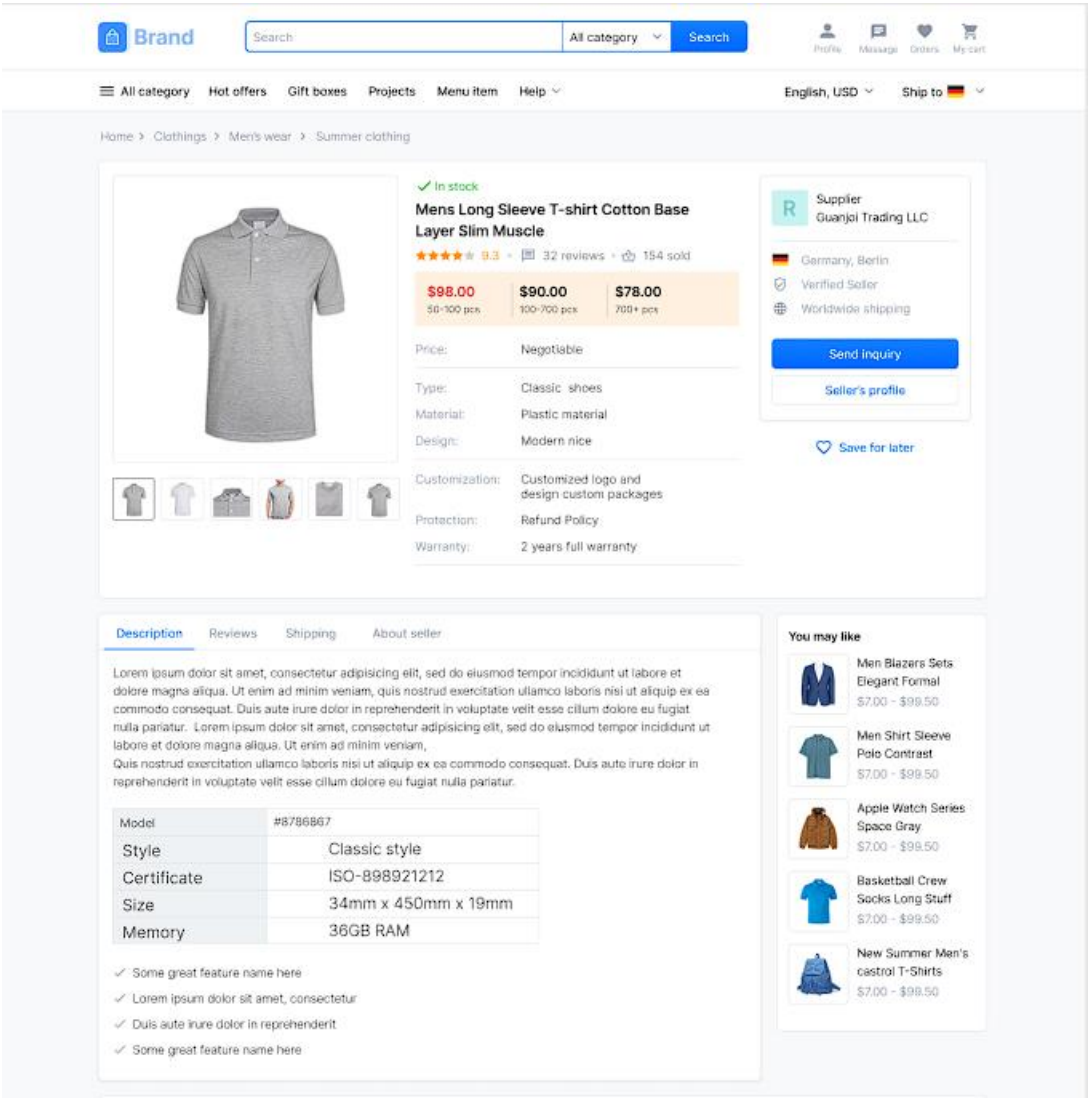


*Figure 21. View items*

## 5.6. *Add items to cart*

(a Button **"My cart"** in View Items)

## *5.7. View cart*

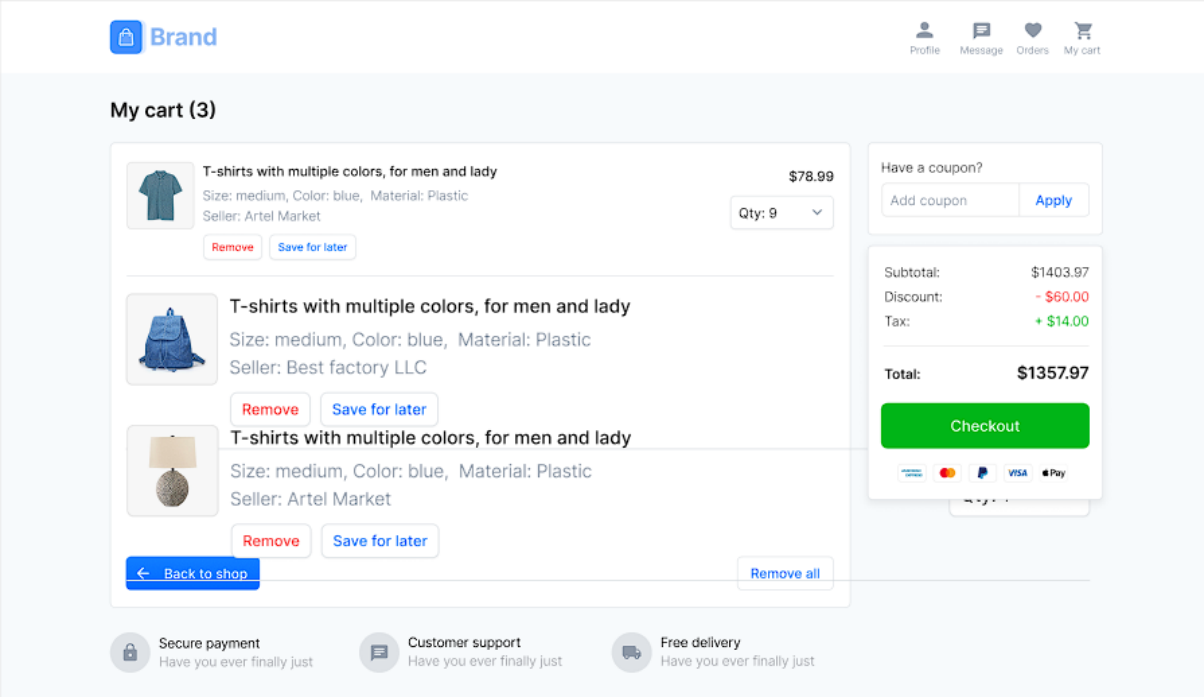


*Figure 22. View cart*

## *5.8. Buy items*

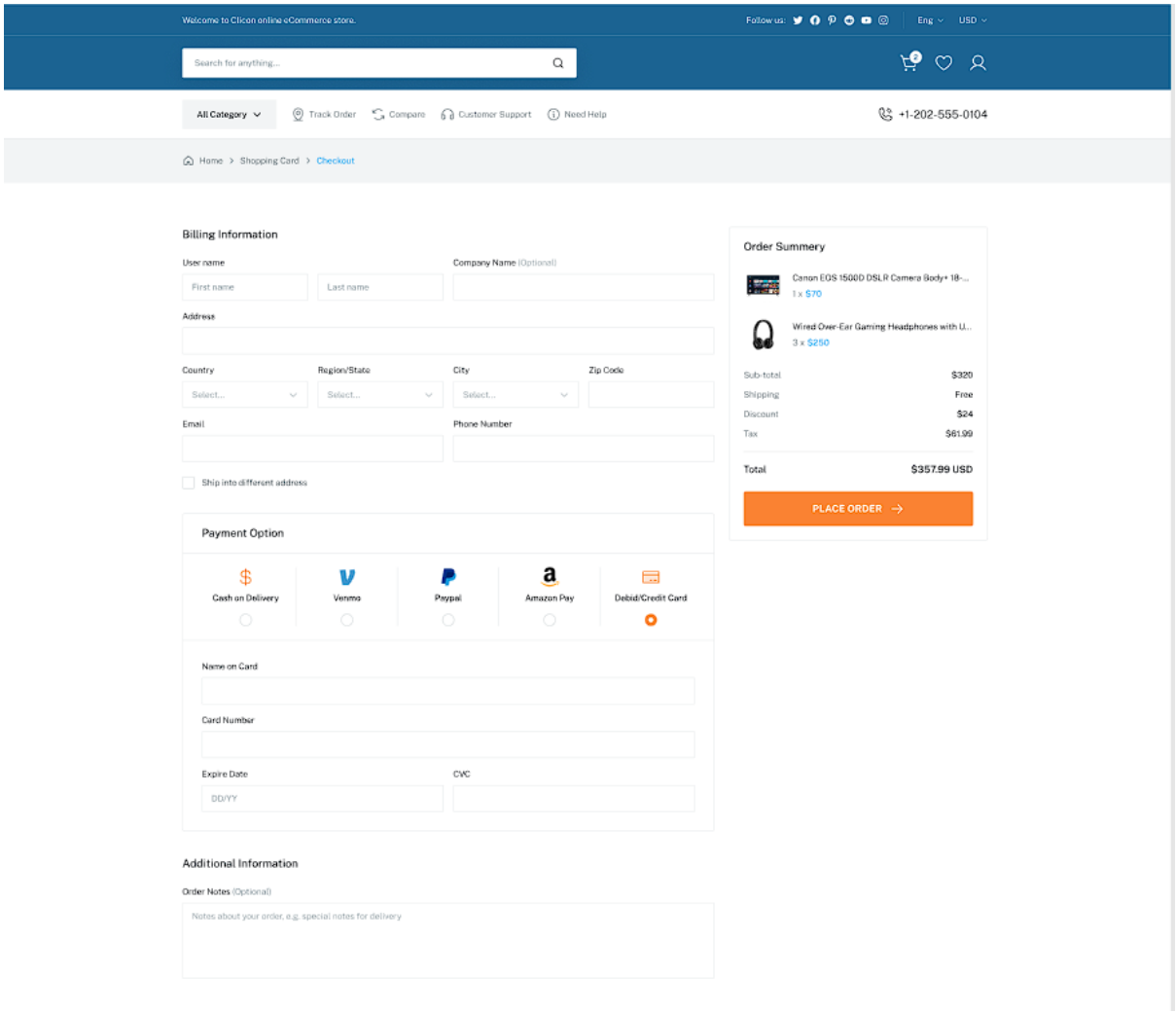


*Figure 23. Buy items*

## 5.9. Shipping tracking

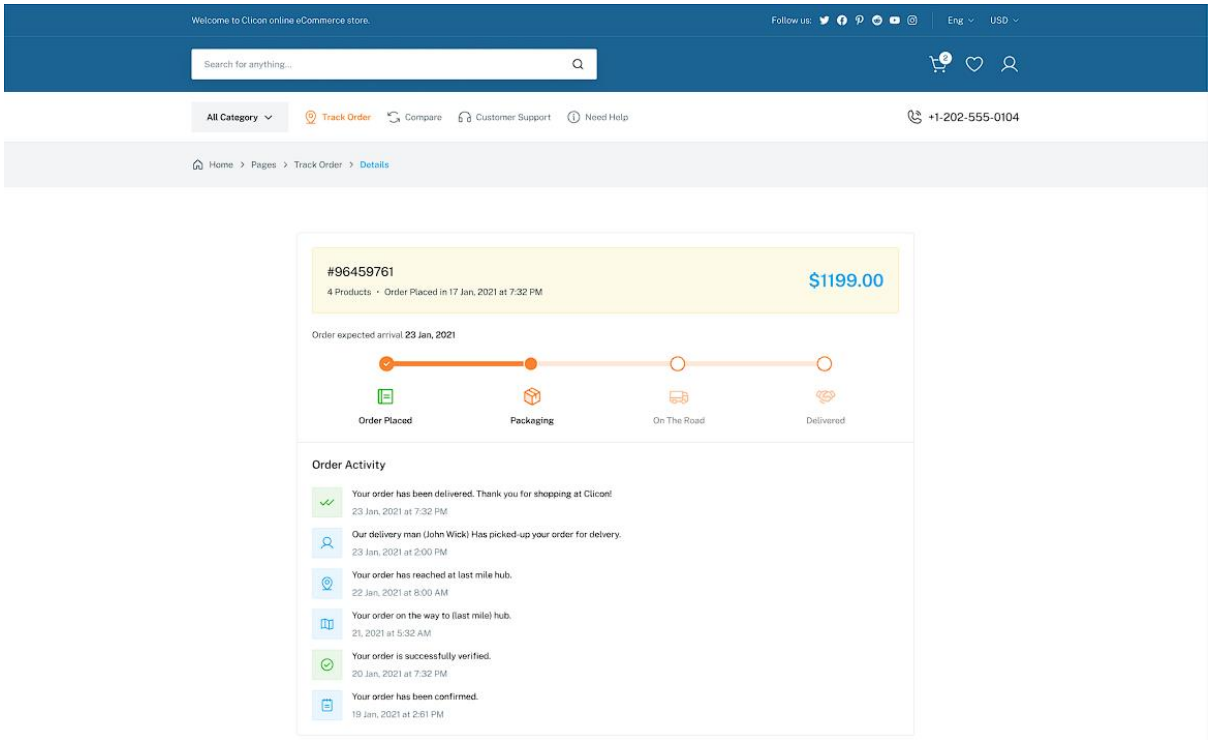


*Figure 24. Shipping tracking*

## 5.10. Order cancellation

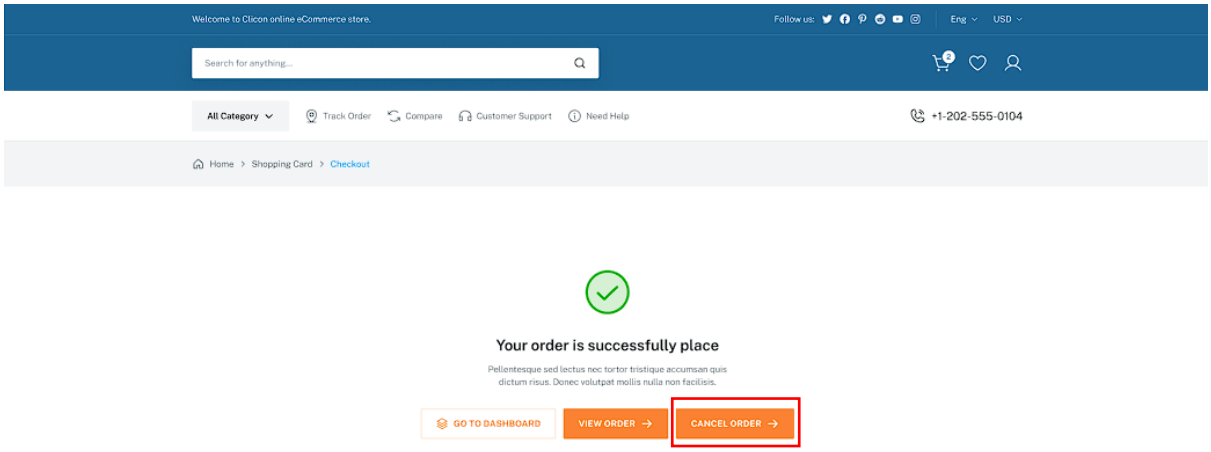


*Figure 25. Order Cancellation*

## 5.11. Order returns

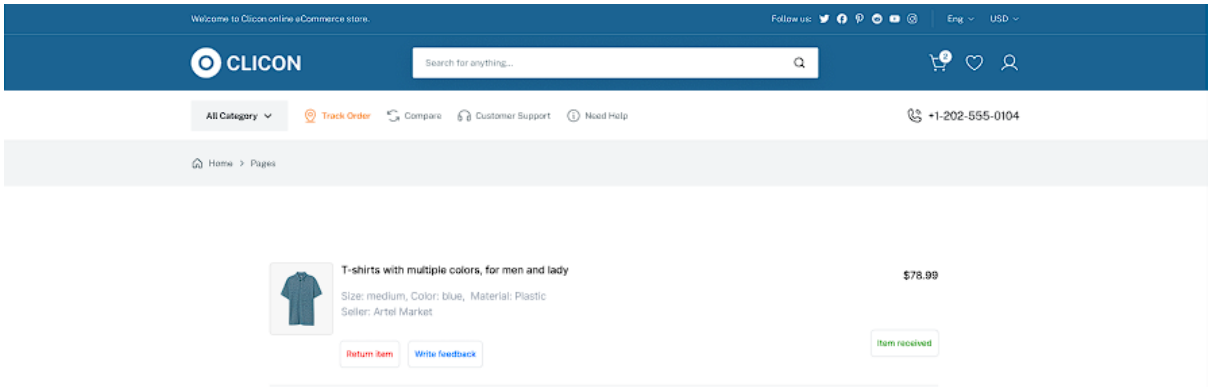


*Figure 26. Order return*

## 5.12. Order confirmation and reviews

(In 2.11. Order return we also have options to confirm and write feedback)

## 5.13. *Upload listings*

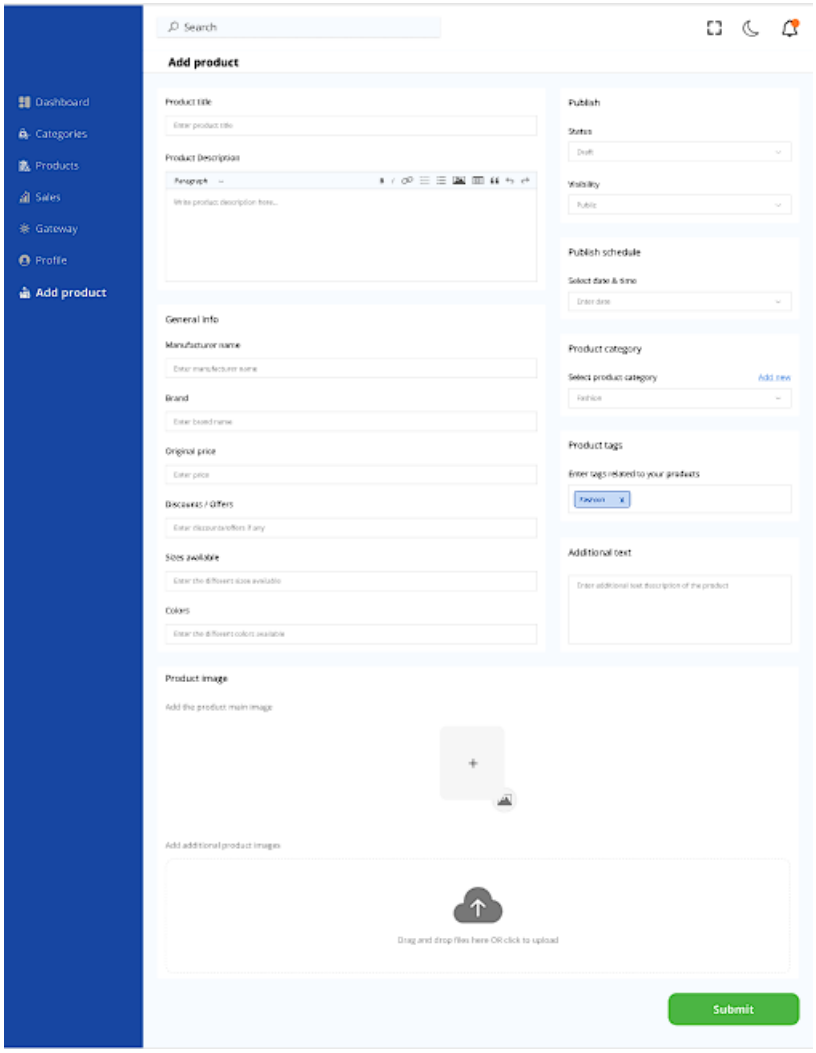


*Figure 27. Upload listings*

## *5.14. Edit items*

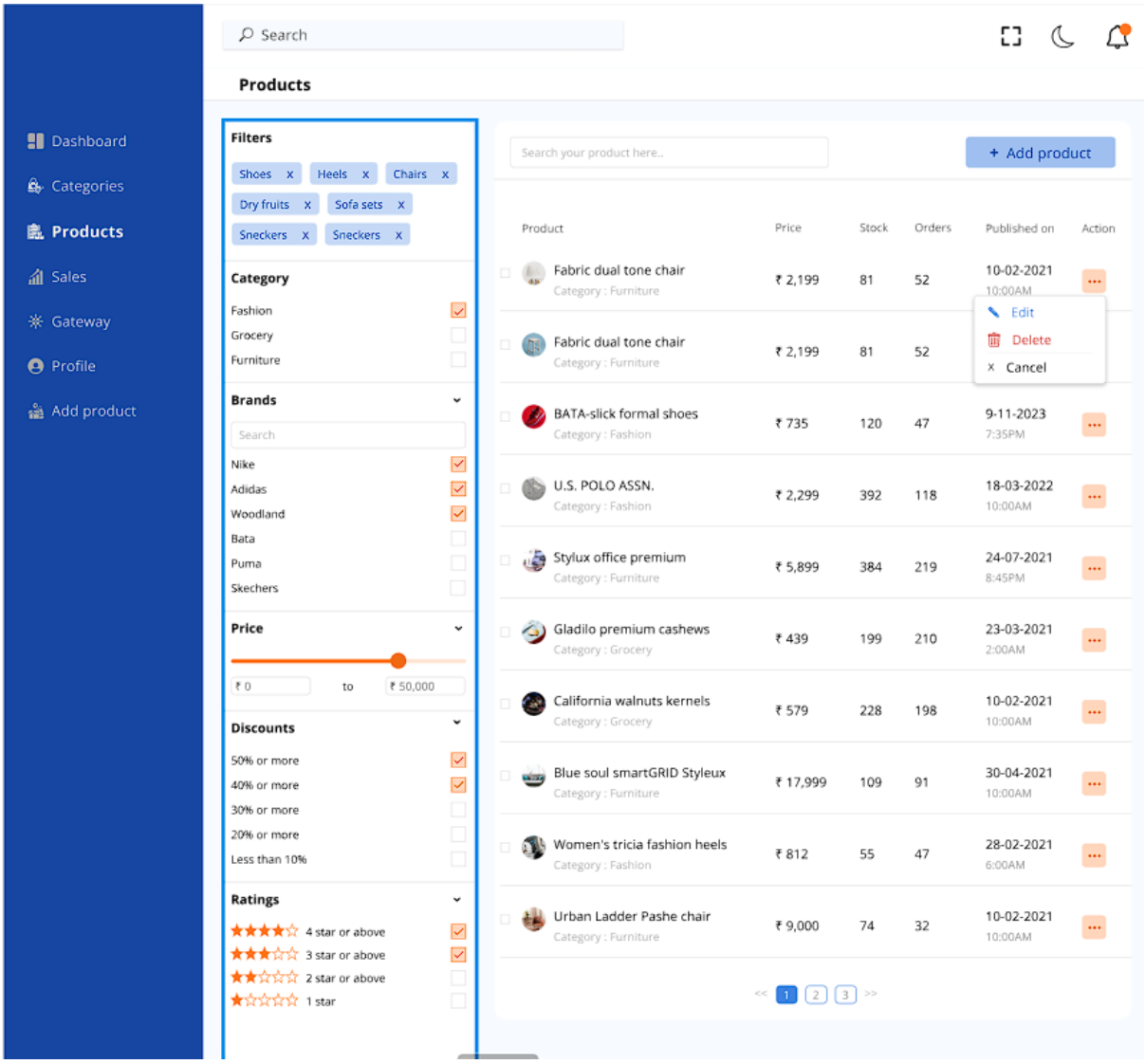


*Figure 28. Edit items*

## 5.15. *Items classification*

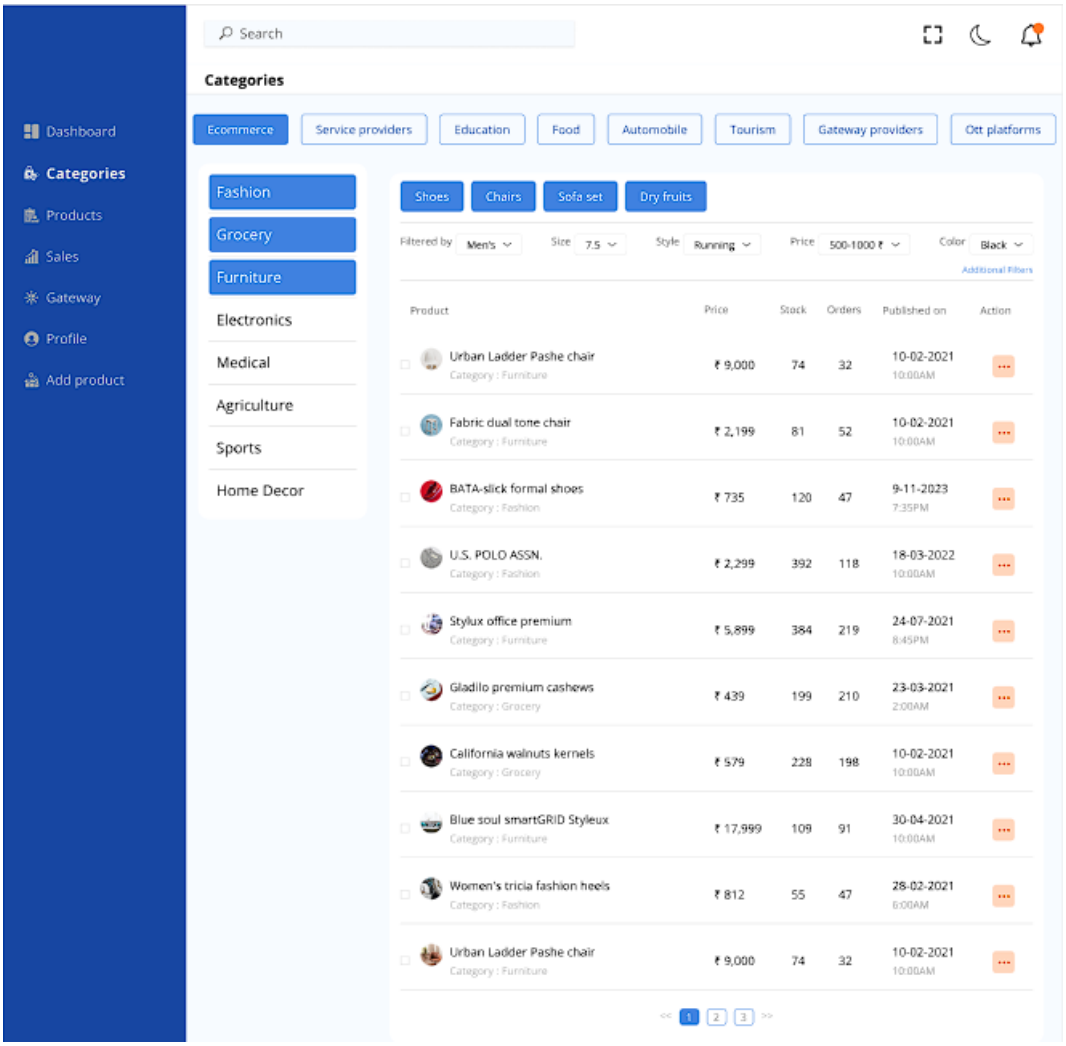


*Figure 29. Items classification*

### 5.16. *View order from customers*

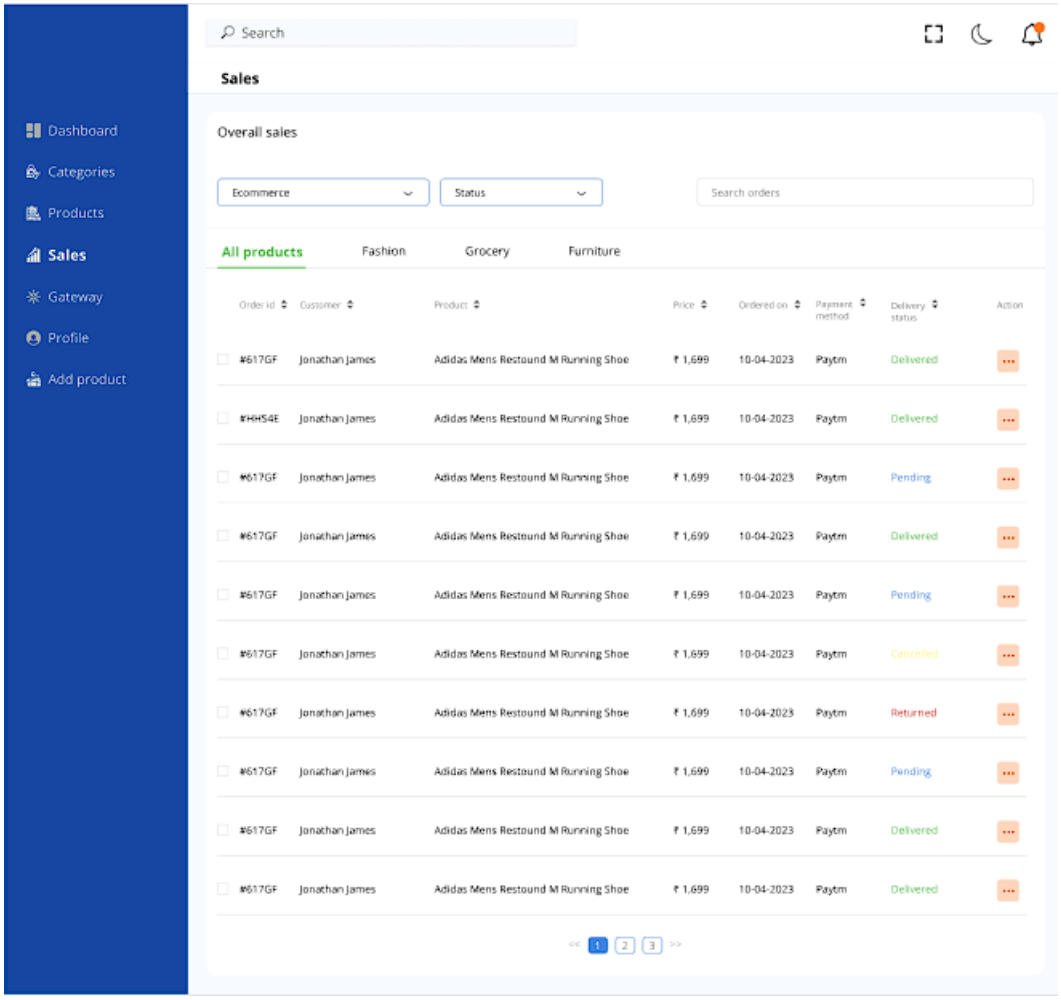


*Figure 30. View order from customers*

# 6. Database Diagram

## 6.1. *EER Diagram*

The schema of classes stored in the system's database will be represented as shown below:

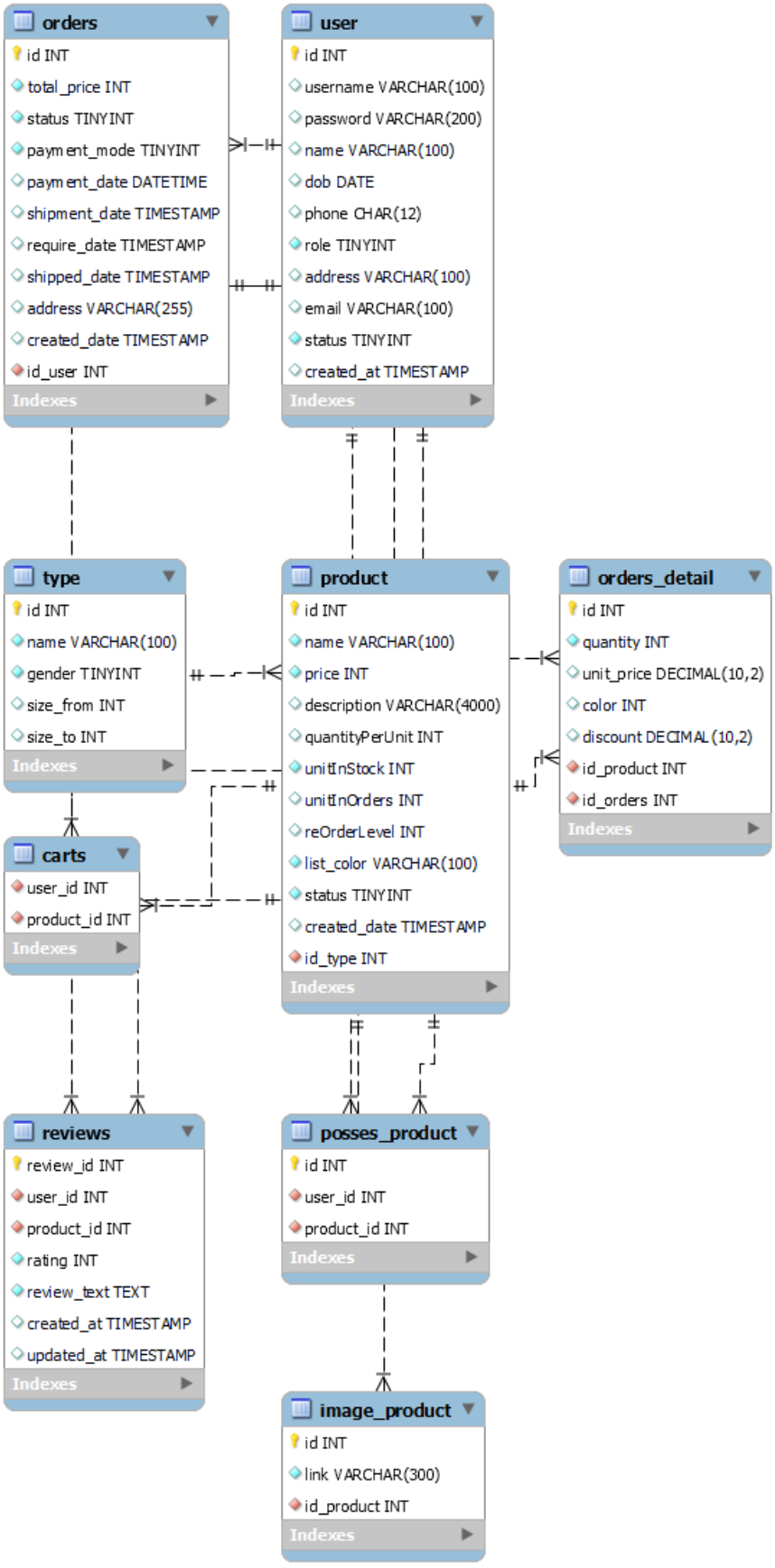


*Figure 31. EER Diagram of the system*

## 6.2. *Details of the tables in EER Diagram*

### 6.2.1. *User table*

*Table 3. EER Diagram - User table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifier for each user. |
| **username** | VARCHAR(100) | User's chosen username for logging in. |
| **password** | VARCHAR(200) | Securely stored password for user authentication. |
| **name** | VARCHAR(100) | Full name of the user. |
| **dob** | DATE | Date of birth of the user. |
| **phone** | CHAR(12) | Contact phone number of the user. |
| **role** | TINY INT | User's role or permissions within the system (e.g., customer, admin). |
| **address** | VARCHAR(100) | User's physical address for shipping or contact purposes. |
| **email** | VARCHAR(100) | User's email address for communication and notifications. |
| **status** | TINY INT | Current status of the user account (e.g., active, inactive). |
| **created_at** | TIMESTAMP | Timestamp indicating when the user account was created. |

### 6.2.2. *Product table*

*Table 4. EER Diagram - Product table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifier for each product. |
| **name** | VARCHAR(100) | Name or title of the product. |
| **price** | INT | Price of the product. |
| **description** | VARCHAR(4000) | Brief description or details about the product. |
| **quantityPerUnit** | INT | Quantity per unit (e.g., per box, per pack) for the product. |
| **unitInStock** | INT | Number of units currently in stock. |
| **unitInOrders** | INT | Number of units currently in orders. |
| **reOrderLevel** | INT | Threshold level for reordering the product. |
| **list_color** | VARCHAR(100) | Color variations available for the product. |
| **status** | TINY INT | Current status of the product (e.g., active, inactive). |

| | | |
|---|---|---|
| **create_time** | TIMESTAMP | Timestamp indicating when the product was added to the system. |
| **id_type** | INT | Identifier for the type or category of the product. |

### *6.2.3. Orders table*

*Table 5. EER Diagram - Orders table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifier for each order. |
| **total_price** | INT | Total price of the order. |
| **status** | TINY INT | Current status of the order (e.g., processing, shipped, delivered). |
| **payment_mode** | TINY INT | Mode of payment used for the order (e.g., credit card, PayPal). |
| **payment_date** | DATETIME | Date when payment for the order was made. |
| **shipment_date** | TIMESTAMP | The estimated date on which the retailer will hand off the shipment to the carrier. |
| **require_date** | TIMESTAMP | Date by which the customer requires the order. |
| **shipped_date** | TIMESTAMP | Date when the order was shipped. |
| **address** | VARCHAR(255) | Shipping address for the order. |
| **created_date** | TIMESTAMP | Timestamp indicating when the order was created. |
| **id_user** | INT | Identifier linking the order to the user who placed it. |

### *6.2.4. Orders_detail table*

*Table 6. EER- Diagram - Order_detail table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifier for each order detail entry. |
| **quantity** | INT | Quantity of the product ordered. |
| **unit_price** | DECIMAL(10, 2) | Unit price of the product at the time of the order. |
| **color** | INT | Color variation selected for the product. |
| **discount** | DECIMAL(10, 2) | Any discount applied to the product in this order detail. |
| **id_product** | INT | Identifier linking the order detail to the product. |
| **id_orders** | INT | Identifier linking the order detail to the order it belongs to. |

### 6.2.5. *Type table*

*Table 7. EER Diagram - Type table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifier for each product type. |
| **name** | VARCHAR(100) | Name or category of the product type. |
| **gender** | TINYINT | Gender category for which the product type is intended (e.g., men, women, unisex). |
| **size_from** | INT | Starting size range for the product type (e.g., smallest available size). |
| **size_to** | INT | Ending size range for the product type (e.g., largest available size). |

### 6.2.6. *Carts table*

*Table 8. EER Diagram - Carts table*

| Field Name | Data Type | Description |
|---|---|---|
| **user_id** | INT | Unique identifier for each product in customer's cart. |
| **product_id** | INT | Identifier for each customer in the database. |

### 6.2.7. *Reviews table*

*Table 9. EER- Diagram - Reviews table*

| Field Name | Data Type | Description |
|---|---|---|
| **review_id** | INT | Unique identifier for each review. |
| **user_id** | VARCHAR(100) | Identifier linking the review to the user who submitted it. |
| **product_id** | TINYINT | Identifier linking the review to the product being reviewed. |
| **rating** | INT | Numeric rating given by the user for the product. (i.e. star given) |
| **review_text** | INT | Textual content of the review provided by the user. |
| **created_at** | TIMESTAMP | Timestamp indicating when the review was created. |
| **updated_at** | TIMESTAMP | Timestamp indicating when the review was last updated. |

### 6.2.8. *Posses_product table*

*Table 10. EER Diagram - Possess_product table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifer for each possession. |
| **user_id** | INT | Unique identifier for each seller user. |
| **product_id** | INT | Identifier for each product sold by the seller user. |

#### *6.2.9. Image_product table*

*Table 11. EER Diagram - Image_product table*

| Field Name | Data Type | Description |
|---|---|---|
| **id** | INT | Unique identifer for each image product. |
| **link** | VARCHAR(300) | Link address to the corresponding source product images |
| **id_product** | INT | Identifier for each product sold by the seller user. |

# 7. Data Structures and Algorithms

## *7.1. Data Structures*

### *7.1.1. Arrays – Store products*

- *Description*

An array is used to store products that are taken from the database.In this approach, each product is represented as an object, and all product objects are stored in a single array. Each object contains key-value pairs representing different attributes of the product.

```
const products = [
    {
      id: 1,
      name: "Product 1",
      price: 10.99,
      description: "Description of Product 1",
      // Other attributes...
    },
    {
      id: 2,
      name: "Product 2",
      price: 19.99,
      description: "Description of Product 2",
      // Other attributes...
    },
    // More product objects...
  ];
```

*Figure 32. Products stored in Arrray, and Object is the abstraction of product*

- *Advantages*

  - **Efficient Data Management**: This approach allows for efficient management of product data, as all products are stored in a single data structure, simplifying operations such as sorting, filtering, and iterating over the product list.
  - **Modularity**: Each product object encapsulates its attributes, promoting modularity and encapsulation in the codebase. This makes it easier to add, update, or remove attributes for individual products without affecting other parts of the code.
  - **Scalability**: While arrays of objects may not be the most efficient solution for extremely large datasets, they are scalable and suitable for moderate-sized product catalogs typically found in small to medium-scale e-commerce websites.
  - **Ease of Serialization**: Array of objects can be easily serialized into JSON format for storage in databases or transmission over the network, facilitating data persistence and communication between client and server components.
  - Highly effective when wanting to implement sorting algorithms to meet user needs.

- *Disadvantages*

  - **Memory Consumption:** Storing all products in memory as objects within an array may consume a significant amount of memory, especially for large product catalogs, potentially affecting application performance and scalability.

### *7.1.2. Object – Represent each product*

- *Description*

Objects are collections of key-value pairs, where each key (also known as a property) maps to a value. The value can be of any data type, including other objects, functions, or primitive values (like strings, numbers, or booleans).

- *Syntax*

- **Object literals:** create an object using curly braces {} and specifying its properties and values directly.
- **Constructor functions:** define a custom constructor function and create objects using the new keyword.

- *Advantages*
  - **Flexibility and Dynamic Properties:**
    - Objects allow adding, modifying, or deleting properties dynamically during runtime.
    - It allow us to create complex data structures by nesting objects within objects.
  - **Key-Value Pair Organization:**
    - Objects provide an intuitive way to organize related data.
    - Properties act as keys, making it easy to access values using descriptive names.
  - **Object-Oriented Programming (OOP):**
    - Objects are fundamental to OOP in JavaScript.
    - Define methods (functions) as properties, enabling behavior encapsulation.
  - **Data Hiding and Encapsulation:**
    - By using closures or constructor functions, we can create private variables (closures) or private methods (constructor functions) within an object.
    - This allows for data hiding and encapsulation.
- *Disadvantages*
  - **Memory Overhead:**
    - Objects consume more memory compared to simple data types (like strings or numbers).
    - Each object has additional overhead for property names and internal bookkeeping.
  - **Complexity and Inefficiency:**
    - Objects can become complex when deeply nested or when they contain circular references.
    - Iterating over object properties (e.g., using for...in loops) can be less efficient than iterating over arrays.
  - **Lack of Order:**
    - Object properties do not have a guaranteed order (unless using ES6 Maps).
    - If order matters, consider using arrays or other data structures.
  - **Mutable State:**

- Objects are mutable, meaning their properties can be changed after creation.
- This mutability can lead to unintended side effects if not managed carefully.

## *7.2. Algorithms*

### *7.2.1. Heap Sort - Arrange items according to the user's filter*

- *Description:*

To cater to user preferences such as displaying items in ascending or descending order of price, date of posting, popularity, or any other criterion, implementing the Heap Sort algorithm can efficiently address this requirement.
Heap Sort is a comparison-based sorting algorithm renowned for its effectiveness in arranging data in either ascending or descending order. It accomplishes this task by constructing a heap data structure from the provided dataset and then iteratively extracting the maximum (for a max-heap) or minimum (for a min-heap) element from the heap. As each element is extracted, the heap is reorganized to maintain its structure, ensuring that the remaining elements are still sorted accordingly.
By leveraging Heap Sort, websites can seamlessly adapt to user preferences and dynamically arrange items based on their desired criteria. Whether users seek to view products by price, date, popularity, or any other parameter, Heap Sort efficiently organizes the displayed items to align with their preferences, thereby enhancing the overall user experience.
In summary, implementing Heap Sort empowers websites to cater to diverse user preferences by efficiently sorting items according to specified criteria, thereby optimizing user satisfaction and engagement.

- *Advantages*

  - **Efficiency:** Heap Sort has a time complexity of O(n log n), which ensures efficient sorting even for large datasets of products. This means that regardless of the number of products being sorted, Heap Sort maintains a consistent level of performance, making it suitable for real-time sorting of products on e-commerce websites.
  - **Stability:** Heap Sort is a stable sorting algorithm, meaning that products with equal sorting keys (e.g., products with the same price) will retain their relative order after sorting. This ensures consistency in the arrangement of products, which is important for user experience. Users can rely on the sorting order to remain consistent across multiple interactions with the website.
  - **Flexibility:** Heap Sort allows sorting based on various criteria chosen by the user, such as price, popularity, rating, or any other relevant attribute. This flexibility enables users to customize how products are displayed according to their preferences and specific needs, enhancing their browsing experience.
  - **Ease of Implementation:** Implementing Heap Sort for sorting products in an array is relatively straightforward and does not require complex modifications or additional code beyond the standard Heap Sort algorithm. This simplifies the development process and ensures that the sorting functionality can be seamlessly integrated into e-commerce websites without significant overhead.

- *Disadvantages*

- **Not Adaptive:** Heap Sort does not adapt its behavior based on the input data, meaning that it does not take advantage of pre-sorted or partially sorted data. This can lead to suboptimal performance in certain scenarios compared to adaptive sorting algorithms.
- **Lack of Cache Efficiency:** Heap Sort has poor cache locality due to its non-locality of reference. This can result in inefficient memory access patterns, especially for large datasets, leading to slower execution times compared to algorithms with better cache efficiency.

# 8. APIs and Dependencies

## *8.1. API*

(None)

## *8.2. Dependencies*

### *8.2.1. React*

React is a popular JavaScript library used to build user interfaces (UIs) for web applications. Developed by Facebook, it is an open-source project. React allows developers to create reusable and efficient UI components, which can then be combined to create complex web applications. It is a technology designed to build dynamic, highly interactive, and high-performance web applications.

### *8.2.2. React-dom*

React-dom is an additional package of React, used to interact with the Document Object Model (DOM) in web applications. It provides methods for rendering React components into the DOM and managing their lifecycle.

### *8.2.3. Vite*

Vite is a high-speed web development tool built by Evan You, the creator of Vue.js. It is designed to provide a powerful and fast development experience for web projects using frameworks such as Vue.js, React, or Svelte. Vite operates by utilizing ESM (ECMAScript Modules) and executing code at build-time to deliver fast response times during development. It leverages Rollup to generate optimized builds for production environments.

### *8.2.4. Tailwind*

Tailwind CSS is a utility CSS library that supports building user interfaces (UIs) in web projects.

### *8.2.5. Express*

Express is a powerful and popular Node.js web framework used to build web applications and APIs. It provides an easy way to handle HTTP requests, route requests and middleware, manage sessions and cookies, and much more**.**

### *8.2.6. MySql*

MySQL is a popular open-source relational database management system widely used in web applications and information systems. It provides a reliable and high-performance database, supporting features such as complex queries, transaction management, automatic backup and recovery, and many other features.

### *8.2.7. Bcrypt*

Bcrypt is an encryption library used to hash passwords in web applications and sensitive data storage systems. It employs a robust hashing algorithm to encrypt passwords before storing them in the database. Utilizing bcrypt enhances password security by encrypting them into irreversible and difficult-to-reverse character strings, safeguarding user data against hacker attacks.