BS (SE) FINAL PROJECT

SNAPPICK APP



Project Advisor

**Mr. Hafiz M.Irfan sab**

Presented by:

**Group ID: 03**

**Faculty of Computing**

**Department of Software Engineering**

**The Islamia University of Bahawalpur  
Bahawalnagar Campus**

Software Requirements Specification

Version 1.0

SNAPPICK APP

Team Members

|  |  |
| --- | --- |
| Member Name | Primary Responsibility |
| Ameer Hamza | Group Leader, Development  QA and Documentation |
| Ome Shifa | Designing, Diagrams and Development |
| Alisha Abbas | Testing, Diagrams and Development. |

Table of Contents

Table of Contents iii

Revision History iii

1. Introduction 1

1.1 Product 1

1.2 Scope 1

1.3 Business Goals 2

1.4 Document Conventions 3

1.5 References 4

2. Overall Description 5

2.1 Product Features 5

2.2 User Classes and Characteristics 7

2.3 Operating Environment 8

2.4 Design and Implementation Constraints 8

2.5 Assumptions and Dependencies 9

3. System Features 10

4. Functional Requirements 13

4.1 Use-Case 1 13

4.2 Use-Case 2 (and so on) 14

5. External Requirements 21

6. Nonfunctional Requirements 24

6.1 Performance Requirements 24

6.2 Safety Requirements 24

6.3 Security Requirements 25

6.4 Software Quality Attributes 25

7. Other Requirements 26

Appendix A: Glossary 33

Appendix B: Analysis Models 35

Appendix C: Design Models 42

Appendix D: Screenshots 44

Appendix E: Test Cases 48

Revision History

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Date** | **Reason For Changes** | **Version** |
|  |  |  |  |
|  |  |  |  |

# Introduction

## Product

**SnapPick Mobile App** is a custom-built e-commerce platform designed to provide users with an exceptional shopping experience. Unlike other e-commerce platforms, SnapPick aims to offer high-quality products at minimal profit margins, ensuring users receive the best value for their money. This focus on affordability, coupled with a seamless and user-friendly experience, differentiates SnapPick from competitors like Shopify. Additionally, SnapPick provides location-based deals tailored to users' registered postal codes, ensuring they receive relevant offers and promotions specific to their geographic area, further enhancing the shopping experience.Scope

## Scope

The SnapPick Mobile App is a custom-built e-commerce platform developed to provide users with an enhanced and affordable online shopping experience [(Zhang & Wang, 2022).](SRS%20E-Commerce%20App.docx#r5) The app will serve as a robust and user-friendly solution for browsing, searching, and purchasing products across Android and iOS platforms [(Khan & Raza, 2020).](SRS%20E-Commerce%20App.docx#r6) Its primary objective is to offer high-quality products at minimal profit margins, ensuring customers receive the best possible value [(Davis & Lee, 2021).](SRS%20E-Commerce%20App.docx#r1) This differentiates SnapPick from other e-commerce platforms, such as Shopify, by offering more competitive pricing while maintaining product quality [(Simmons, 2023).](SRS%20E-Commerce%20App.docx#r2)

The app will feature core functionalities such as user registration, personalized login, product search with advanced filtering options, a dynamic shopping cart, and a secure checkout process, including payment options, such as Cash on Delivery (COD). A unique feature of SnapPick will be the location-based deals, which will allow users to receive relevant promotions based on their registered postal code, providing a more personalized shopping experience [(Nguyen & Tran, 2020).](SRS%20E-Commerce%20App.docx#r3)

SnapPick’s development aligns with the overarching business strategy of providing users with a custom-built e-commerce solution tailored to specific needs, as opposed to relying on generic platforms like Shopify. By focusing on affordability, quality, and ease of use, the SnapPick app aims to create a loyal customer base, increase engagement, and establish a competitive edge in the e-commerce market. The app’s goals are directly linked to the company’s long-term vision of expanding its market presence by offering unique features and services that are not available on other platforms [(Anderson & White, 2020).](SRS%20E-Commerce%20App.docx#r4)

## Business Goals

The **SnapPick Mobile App** is designed to support and align with several key business objectives, each contributing to the overall success of the e-commerce platform. The business goals addressed by this software include:

1. **Provide an Affordable Shopping Experience**: By offering high-quality products at minimal profit margins, SnapPick aims to deliver exceptional value to customers, distinguishing itself from other e-commerce platforms such as Shopify.
2. **Increase Market Reach and Customer Engagement**: The app aims to expand its customer base by offering a user-friendly and accessible platform on both **Android** and **iOS** devices, reaching a broader demographic of online shoppers.
3. **Enhance Customer Satisfaction and Loyalty**: With personalized features such as **location-based deals** and a seamless shopping experience, the app seeks to foster customer loyalty, ensuring that users are consistently engaged and satisfied with their purchases.
4. **Achieve Competitive Advantage**: SnapPick intends to create a unique position in the market by offering a tailored, custom-built solution that provides flexibility, quality, and affordability that are not typically available on existing platforms.
5. **Facilitate Efficient Order Management**: The app's efficient order processing, tracking, and notification systems will streamline order management, ensuring that customers receive timely and accurate updates about their orders.
6. **Optimize Revenue Generation**: By incorporating features such as product recommendations, location-based deals, and promotions, the app aims to maximize sales, increase conversion rates, and optimize revenue through targeted marketing strategies.
7. **Scalable Platform for Future Growth**: The app’s infrastructure is designed to be scalable, allowing for easy integration of new features, expansion into new markets, and support for future business growth.
8. **Develop Brand Recognition**: SnapPick seeks to establish a strong brand presence in the e-commerce industry, building trust and recognition through consistent product quality, user experience, and customer service.

## Document Conventions

The entire document must be justified.

**Convention for Main Title:**

* Font Face: Times New Roman
* Font Style: Bold
* Font Size: 18

**Convention for Sub Title:**

* Font Face: Times New Roman
* Font Style: Bold
* Font Size: 14

**Convention for Heading:**

* Font Face: Times New Roman
* Font Style: Bold
* Font Size: 12

**Convention for Body:**

* Font Face: Times New Roman
* Font Style: None
* Font Size: 12

The document is prepared using **Microsoft Word 2019** and utilizes the font type **Times New Roman**. The fixed font size used throughout the document is **14pt** for headings with **1.0 line spacing**. The **bold** property is applied to the headings to distinguish them from the body text.

## References

* + 1. **Journal Reference**

1. Davis, A., & Lee, J. (2021). E-commerce personalization and customer retention. Journal of Digital Commerce, 10(2), 45–56.
2. Simmons, R. (2023). The impact of affordability on online retail platforms. Journal of Business Strategy, 8(1), 30–42.
3. Nguyen, T., & Tran, M. (2020). Geolocation technology in e-commerce: A review. International Journal of Computing, 15(3), 88–101.
4. Anderson, K., & White, H. (2020). Custom-built e-commerce solutions: A competitive edge. International Business Review, 22(4), 67–79.
5. Zhang, Y., & Wang, S. (2022). Optimizing mobile e-commerce app interfaces for better user experience. Journal of Mobile E-Commerce, 14(3), 180-194.
6. Khan, M. S., & Raza, M. S. (2020). Mobile commerce trends and their impact on e-commerce platforms. E-Commerce and Technology Review, 19(2), 122-136.
   * 1. **BOOKs:**
7. Ian Sommerville, Software Engineering 8th edition. Pearson education,2008
8. Trafficspecial – john resse,E\_commerce website.2003

# Overall Description

## Product Features

### ****User Registration and Login****

* **Sign Up / Login**: Allows users to create accounts and log in for a personalized shopping experience.
* **Password Recovery**: Enables users to reset their passwords if they forget them.

### ****Product Search and Browsing****

* **Search Bar**: Users can search for products using keywords, categories, or tags.
* **Filters and Sorting**: Filters such as price range, category, brand, ratings, etc., and sorting by price, popularity, or new arrivals.
* **Product Details Page**: Comprehensive details about the product including images, specifications, reviews, and related products.

### ****Product Catalog Management****

* **Inventory Management**: Keep track of product availability and stock levels.
* **Product Variants**: Support for different sizes, colors, or models of a product.
* **Product Recommendations**: Suggested products based on previous searches or purchases.

### ****Shopping Cart****

* **Add to Cart**: Users can easily add products to their cart.
* **View Cart**: Allows users to review and modify cart items before checkout.
* **Save for Later**: Option to save items in the cart for future purchase.

### ****Checkout Process****

* **Guest Checkout**: Allows users to checkout without creating an account.
* **Address and Shipping Details**: Collects customer’s delivery address and preferred shipping method.
* **Payment Options**: Supports various payment methods, such as credit/debit cards, PayPal, COD (Cash on Delivery), or mobile payments.
* **Order Summary**: Users see a final review of their order, including products, shipping details, and total cost before confirming.

### ****Order Management****

* **Order Confirmation**: Confirmation email or notification sent to the customer.
* **Order History**: Users can view their past orders and reorder easily.

### ****User Reviews and Ratings****

* **Product Reviews**: Customers can leave reviews for products they’ve purchased, helping future customers make informed decisions.
* **Ratings**: Star ratings (1-5) for each product to give customers a quick overview.

### ****Wishlist****

* **Save for Later**: Customers can add products to their wishlist for future purchase.
* **Notifications**: Notify customers when their wishlist products are on sale or back in stock.

### ****Users Review and Ratings****

* **Product Reviews:** Customers can leave reviews for products they’ve purchased, helping future customers make informed decisions.
* **Ratings:** Star ratings (1-5) for each product to give customers a quick overview.

#### ****2.1.10 Location-Based Deals****

* The system shall allow users to view location-specific deals based on their registered postal code.
* The system shall enable administrators to create and assign deals to specific locations.

#### ****2.1.11 Logout Functionality****

* The system shall provide a secure logout option for users with responding efficiently.

## User Classes and Characteristics

**2.2.1 User** **classes**

* App Administration
* Customer
* customers service

**2.2.2 Characteristics of User Classes**

* Administration is responsible for managing the entire Application.
* The customer is responsible for purchasing products from the App.
* Customer service is responsible for solving problems encountered by customers and answering their questions.

## Operating Environment

### Hardware Interfaces

The SnapPick Mobile App will function on smartphones and tablets with the following minimum hardware specifications:

* **RAM:** Minimum of 2GB
* **Operating System:** Android 10+ or iOS 14+
* **Screen Size:** Devices with screen sizes ranging from 4.5 inches to 6.7 inches.
* **Processor:** Multi-core processors (e.g., Snapdragon 600 series or equivalent for Android and A12 Bionic or equivalent for iOS).

### Software Interfaces

The SnapPick Mobile App will interact with various software components to ensure a seamless user experience:

* **Backend API:** The app will integrate with a PHP-based backend API using RESTful services for data management and transaction processing.
* **Database Management:** The app will communicate with a MySQL database for storing and retrieving product, user, and order data.
* **Payment Gateway:** Integration with a secure payment gateway for Cash on Delivery (COD).
* **Geolocation Services:** The app will use geolocation APIs (e.g., Geolocator in Flutter) to capture the user’s location for location-based deals.

## Design and Implementation Constraints

The development of the SnapPick Mobile App is subject to the following constraints:

* **Regulatory Compliance:** The app must comply with local e-commerce laws and data privacy regulations, ensuring secure handling of user information.
* **Hardware Limitations:** The app must operate efficiently on devices with at least 2GB RAM and meet the requirements of Android 10+ and iOS 14+ operating systems.
* **Technology Stack:** The backend must be implemented using PHP, with MySQL as the database. The frontend will use Flutter to support cross-platform development for Android and iOS.
* **Integration Requirements:** The app must integrate seamlessly with third-party APIs such as geolocation services, Firebase for notifications, and payment gateways for COD and future payment options.
* **Security Considerations:** Data transmission must be secured using HTTPS, and sensitive data like user credentials must be encrypted.
* **Performance Standards:** The app must maintain optimal performance with a high number of concurrent users and handle large datasets without significant latency.
* **Maintenance and Scalability:** The design must support easy updates and scalability to accommodate future features and increased user demand.

## Assumptions and Dependencies

**2.5.1 The Assumptions are:-**

* The coding should be error free.
* The system should be user friendly so that it is easy to use for the users.
* The system should have more capacity and provide fast access to the database.
* The system should provide search facility and support quick transactions.
* Users may access from any moble device that has internet browsing capabilities and an internet connection.

**2.5.2 The dependencies are:-**

* The specific hardware and software due to which the product will be run.
* On the basis of listing requirements and specification the project will be develop and run.
* The end users (admin) should have proper understanding to the product.
* The information of all users must be stored in a database
* The system should have the general report store.

# System Features

## Customer Register

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Customer Register | User name, Phone Number, Email, Address, Gender, Postal Code. | Database record created. Success message displayed. | Validate details, check for duplicates, and store information in the database. |

## Customer Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Customer Login | Email and Password. | Access granted or error message displayed. | Validate input, authenticate credentials, and grant access or display an error. |

## Product Browsing and Searching

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Product Browsing and Searching | Search term (e.g., product name or keyword). | List of matching products displayed. | Query database using search term and return sorted product results. |

## Location-Based Deals

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Location-Based Deals | User’s registered postal code. | Location-specific deals displayed. | Retrieve location-specific deals from the database and display them on the homepage or section. |

## Add to Cart

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Add to Cart | Product, Quantity, User session or account. | Product added to cart. Cart updated with product details. | Validate product availability, update cart with product details, and calculate total cart value. |

## Checkout

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Checkout | Cart items, Delivery address, Contact details, Payment method. | Order summary, Payment confirmation, Success message. | Validate cart items, apply discounts or shipping fees, process payment (COD), and generate order for user. |

## Review and Rating

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Review and rating | Click on the "Logout" button in the profile menu. | The system retrieves and displays all reviews and the average rating for the product. | The review is stored in the database and the rating is updated for the respective product. |

## Order Placement

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Order Placement (COD) | Cart items, delivery address, and contact details. | Order confirmation message displayed. Order saved as 'Pending.' | Validate cart items and address, calculate total cost, save order to database, and notify user. |

## Notifications

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Notifications | The user submits the review and rating. | Push notification sent to user. | Detect status changes in the database and trigger notifications to user devices. |

## Logout

|  |  |  |  |
| --- | --- | --- | --- |
| **Feature Title** | **Input** | **Output** | **Processing** |
| Log out | Click on the "Logout" button in the profile menu. | The user with a confirmation dialog to confirm logout. | The system terminates the user session, clears cached data, and redirects the user to the login screen. |

# Functional Requirements

## User Registration and Login

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0001 | |
| **Purpose** | | To allow users to create accounts and log in to access personalized shopping features. | |
| **Priority** | | High | |
| **Pre-conditions** | | 1. The user must have a valid email address. 2. The app must be connected to the internet. | |
| **Post-conditions** | | 1. For new users, an account is created and saved in the database. 2. For existing users, access is granted to their dashboard | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user selects "Sign Up" or "Login." | | The system displays the corresponding action. |
| **2** | The user enters required credentials. | | The system validates the credentials. |
| **3** | The user submits the details. | | The system grants access or shows an error message. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user enters an invalid email or password. | | The system displays an error message. |
| **2** | The user clicks "Forgot Password." | | The system sends a password recovery email. |

Table 1: Registration and Login

## Product Browsing and Searching

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0002 | |
| **Purpose** | | To allow users to search and browse products efficiently by keywords, categories, or filters. | |
| **Priority** | | High | |
| **Pre-conditions** | | 1.The app must be connected to the internet. 2. Product data must be available in the database. | |
| **Post-conditions** | | 1. The user views a list of relevant products. 2.The system saves search filters for user convenience (if implemented). | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user enters a keyword in the search bar. | | The system displays matching products. |
| **2** | The user selects a category or filter. | | The system applies the selected filter and updates the product list. |
| **3** | The user clicks on a product. | | The system displays detailed product information. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user enters a search term that yields no results. | | | The system displays a "No Products Found" message and suggests similar products. |
| **2** | The user resets the filters. | | The system reloads the complete product list. |

Table: Product Searching

## Shopping Cart

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0003 | |
| **Purpose** | | To allow users to add, view, and manage products in their shopping cart. | |
| **Priority** | | High | |
| **Pre-conditions** | | 1.The user must be logged in to save the cart. 2. Product data must be available in the database. 3. The app must be connected to the internet. | |
| **Post-conditions** | | 1. Products are added to the cart.  2. The user can review, update, or remove items from the cart. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user selects a product to add to the cart. | | The system validates stock availability and adds the product to the cart. |
| **2** | The user views the cart. | | The system displays all items in the cart with prices and quantities. |
| **3** | The user updates the quantity of a product. | | The system updates the total price and adjusts the stock accordingly. |
| **4** | The user removes an item from the cart | | The system removes the item and updates the cart total. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user tries to add an out-of-stock product to the cart. | | | The system displays an "Out of Stock" message. |
| **2** | The user clears the cart. | | The system removes all items and displays an empty cart message. |

Table3: Shopping Cart

## Checkout Process

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0004 | |
| **Purpose** | | To allow users to review their cart, provide delivery details, and confirm their orders with the selected payment method. | |
| **Priority** | | High | |
| **Pre-conditions** | | 1.The user must have items in the shopping cart. 2.The app must be connected to the internet. 3. Delivery addresses must be valid and complete. | |
| **Post-conditions** | | 1.The order is placed and saved in the database. 2.The user receives an order confirmation message or email. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user proceeds to checkout from the cart. | | The system displays the checkout page with cart details and total cost. |
| **2** | The user enters or selects a delivery address. | | The system validates the address and displays available shipping options. |
| **3** | The user confirms the payment method (e.g., COD). | | The system verifies the payment option and updates the order details. |
| **4** | The user confirms the order. | | The system saves the order in the database and displays a confirmation message or email. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user enters incomplete delivery details. | | | The system prompts the user to complete the missing fields. |
| **2** | The user cancels the checkout process. | | The system retains the cart contents for later checkout. |

Table 4: Checkout Process

## User Reviews and Ratings

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0005 | |
| **Purpose** | | To allow users to provide feedback on purchased products and view reviews and ratings to assist in making informed decisions. | |
| **Priority** | | Medium | |
| **Pre-conditions** | | 1.The user must be logged into their account. 2.The user must have purchased the product being reviewed. 3.The product must exist in the database. | |
| **Post-conditions** | | 1.The review is stored in the database. 2.The rating is updated for the respective product. 3. Other users can view the review and rating. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user navigates to the purchased product's page. | | The system displays the product details and review section. |
| **2** | The user clicks on "Write a Review" or "Rate Product." | | The system displays a form for entering review text and selecting a rating (1-5 stars). |
| **3** | The user submits the review and rating. | | The system validates the input, saves the review and rating, and displays a confirmation message. |
| **4** | The user views reviews for the product. | | The system retrieves and displays all reviews and the average rating for the product. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user submits an incomplete review or invalid rating. | | | The system prompts the user to provide complete and valid input. |
| **2** | The user attempts to review a product they haven’t purchased. | | The system displays an error message and prevents review submission. |

Table 5: Review & Rating

## Wishlist

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0006 | |
| **Purpose** | | To allow users to save products to a wishlist for future reference and receive notifications about changes related to those products. | |
| **Priority** | | Medium | |
| **Pre-conditions** | | 1. The user must be logged into their account. 2. The product must exist in the database. | |
| **Post-conditions** | | 1.The product is successfully added to the wishlist. 2.The user can view and manage their wishlist. 3. Notifications are sent for relevant changes  (e.g., price drop, back in stock). | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user navigates to the product page. | | The system displays the product details. |
| **2** | The user clicks on "Add to Wishlist." | | The system adds the product to the user’s wishlist and displays a success message. |
| **3** | The user views their wishlist. | | The system retrieves and displays the list of saved products. |
| **4** | The user removes an item from the wishlist. | | The system removes the selected item and updates the wishlist. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user tries to add a product that is already in the wishlist. | | | The system informs the user that the product is already in the wishlist. |
| **2** | The user accesses the wishlist but it is empty. | | The system displays a message encouraging the user to browse and add products. |

Table 6: Wishlist

## Location-Based Deals

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0007 | |
| **Purpose** | | To display deals and offers the user’s location based deals on their registered postal code. | |
| **Priority** | | Medium | |
| **Pre-conditions** | | 1.The user must have a registered account. 2.The user’s postal code must be provided and stored in their profile. 3. Relevant deals must exist in the system's database for the user's location. | |
| **Post-conditions** | | 1.The user is presented with deals specific to their registered location. 2. Deals are updated dynamically if the user’s postal code changes. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user logs into the app. | | The system retrieves the user’s postal code and fetches location-based deals. |
| **2** | The user navigates to the “Deals” section. | | The system displays deals relevant to the user’s location. |
| **3** | The user clicks on a deal to view details. | | The system shows detailed information about the selected deal, including terms and conditions. |
| **4** | The user updates their postal code in the profile. | | The system updates the location and fetches deals for the new postal code. |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user’s postal code does not match any active deals. | | | The system informs the user that no deals are available for their location and suggests browsing other categories. |
| **2** | The user tries to access deals without a registered account. | | The system prompts the user to log in or sign up to access location-based deals. |

Table7: Location Based Deals

## Logout Functionality

|  |  |  |  |
| --- | --- | --- | --- |
| **Identifier** | | UC-0008 | |
| **Purpose** | | To allow users to securely log out of the application and terminate their session. | |
| **Priority** | | Medium | |
| **Pre-conditions** | | 1.The user must be logged into their account. 2.The application must be connected to the backend server. | |
| **Post-conditions** | | 1.The user session is terminated securely. 2. Any cached session data is cleared from the app. | |
| **Typical Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | The user clicks the "Logout" button in the profile menu. | | The system prompts the user with a confirmation dialog to confirm logout. |
| **2** | The user confirms the logout action. | | The system terminates the user session, clears cached data, and redirects the user to the login screen. |
| **3** | The user closes the app after logging out. | | The system ensures no session data is retained |
| **Alternate Course of Action** | | | |
| **S#** | **Actor Action** | | **System Response** |
| **1** | |  | | --- | | The user tries to log out while offline. | | | The system notifies the user that logout will be processed when reconnected to the server and clears local session data temporarily. |
| **2** | The user cancels the logout action after the confirmation dialog. | | The system closes the dialog and keeps the user logged in. |

Table8: Logout

# External Interface Requirements

## User Interfaces

The SnapPick Mobile App will feature a user-friendly interface optimized for mobile devices, ensuring a seamless shopping experience. Key UI characteristics include:

* **Layout**: The app will have a modern design with a bottom navigation bar for quick access to Home, Search, Cart, Profile, and Orders. The layout will be responsive to accommodate various screen sizes and orientations (portrait and landscape).
* **Buttons and Functions**: Standard buttons such as "Add to Cart," "Checkout," and "Log In" will be strategically placed for easy access. A "Help" button will appear on all screens for user assistance.
* **Input Fields**: Forms for registration, checkout, and login will include labeled input fields, placeholders, and validation messages for clear user guidance.
* **Error Messages**: Clear, concise error messages will be displayed near invalid input fields to help users correct mistakes. Generic error messages will guide users to contact support when necessary.
* **Navigation**: The app will feature a consistent navigation bar with accessible icons and labels for easy movement between sections. Users can also navigate using swipe gestures.
* **Feedback**: Visual feedback, such as loading spinners and confirmation messages, will appear during processing actions, ensuring users are informed of progress.
* **Design Standards**: The UI will adhere to Material Design principles for Android and Human Interface Guidelines for iOS, maintaining consistency across platforms.

## Hardware Interfaces

The SnapPick Mobile App is designed to run on smartphones with the following minimum hardware specifications:

* **Supported Device Types**: Smartphones with Android 10+ and iOS 14+.
* **Physical Components**: The app will interact with mobile device components such as the touchscreen, camera (for barcode scanning), GPS (for location-based deals), and sensors (e.g., accelerometer for gesture-based navigation).
* **Data and Control Interactions**: The app will communicate with the device’s hardware for tasks such as capturing user location via GPS for location-based deals and enabling barcode scanning for product information.
* **Communication Protocols**: The app will use standard communication protocols like HTTP/HTTPS for API calls to the backend, and Bluetooth/Wi-Fi when necessary for device pairing or data syncing.

## Software Interfaces

The SnapPick Mobile App interfaces with several key software components to ensure smooth functionality:

* **Backend API**: The mobile app communicates with a PHP-based backend API for handling product data, user profiles, orders, and payments.
* **Database**: The system uses a MySQL database for storing user data, product details, order history, and reviews. It supports CRUD (Create, Read, Update, Delete) operations between the mobile app and the database.
* **Geolocation APIs**: The app integrates with Geolocator (Flutter package) for capturing the user’s location to provide location-based deals.
* **Communication Protocols**: The mobile app uses HTTPS for secure communication between the frontend and backend. RESTful API calls are made to retrieve and send data.
* **External Libraries**: The app uses Flutter SDK for frontend development and Flutter plugins for accessing native device features such as camera, GPS, and payment systems.

## Communications Interfaces

The SnapPick Mobile App requires robust communication capabilities to support user interactions and backend synchronization. The communication interfaces include:

* **Network Protocols**: The app uses HTTPS for secure communication between the frontend (mobile app) and the backend (PHP-based API). All data transferred over the network will be encrypted using **SSL/TLS** to ensure secure transactions.
* **Email Notifications**: The system will send email notifications to users for order confirmations, shipment updates, and promotional offers. Email delivery will be facilitated through third-party email services (e.g., SendGrid or Amazon SES).
* **Payment Gateway Communication**: Communication with external payment gateways (e.g., PayPal, Stripe) is conducted via their RESTful APIs to securely process transactions. Data related to user payments will be encrypted and transferred securely.
* **Web Services**: The app will communicate with the backend through RESTful API calls using JSON as the data format. All requests and responses between the mobile app and server will be in JSON format for easy parsing and data exchange.
* **Data Transfer Rates**: Communication between the app and backend should ensure **minimal latency**, with a target of 2-3 seconds for typical API response times. For a smooth user experience, the app should handle intermittent network issues gracefully.
* **Security and Encryption**: All communication, including user credentials, payment information, and personal data, will be encrypted using AES and SSL/TLS protocols to ensure data security during transmission.

# Nonfunctional Requirements

## Performance Requirements

* **Response Time:** Any interaction between the user and the website should not exceed 2 seconds to ensure a fast and seamless browsing experience for users, enhancing user satisfaction.
* **Availability:** The website should be available for use 24 hours a day, 365 days per year, ensuring that users can access the platform at any time without interruptions.
* **Concurrency:** The website must support up to 500 simultaneous users at any given time, ensuring that high traffic and multiple users do not affect the website's performance or functionality.
* **Data Synchronization:** The website should download new status parameters within 5 minutes of a change, ensuring that users have the most up-to-date information and enhancing the real-time experience.

## Safety Requirements

* **Data Backup and Recovery:** Regular backups of user data, product information, and transaction history must be maintained to ensure recovery in case of system failure or data corruption.
* **User Data Protection:** Personal user information, including login credentials and payment details, must be encrypted both during transmission and while stored in the database to prevent unauthorized access.
* **Secure Payment Processing:** All payment transactions must be processed through secure and verified payment gateways, with encryption protocols like HTTPS to safeguard sensitive financial data.
* **Access Control:** Different types of users (admin, customer, guest) should have clearly defined access levels, ensuring that only authorized personnel can access sensitive data and perform critical operations.
* **Session Timeout and Auto-Logout:** The system must implement automatic session timeouts after a period of inactivity to prevent unauthorized access to the app, ensuring users' data is protected when not in use.

## Security Requirements

* **Secured Database:** The system must employ a secured database to ensure data protection and prevent unauthorized access to sensitive information.
* **User Permissions:** Users can only view information but cannot modify it, except for their personal details, ensuring data integrity and privacy.
* **Role-Based Access Control:** The system should differentiate user roles and impose specific restrictions to safeguard sensitive data and operations.
* **Secure Communication:** All communications must be conducted over HTTPS rather than HTTP-4 to ensure encryption and secure data transfer.

## Software Quality Attributes

* **Usability**: The app will have an intuitive interface ensuring ease of use for new and returning users.
* **Reliability**: The app will operate consistently without crashes or failures during common tasks.
* **Performance:** The app will provide fast response times, with pages loading in under 3 seconds.
* **Maintainability**: The app will feature modular code and thorough documentation to support easy updates and bug fixes.
* **Security**: The app will ensure secure storage and transmission of sensitive user data.
* **Scalability**: The app will be able to handle increasing numbers of users and product listings without performance degradation.
* **Interoperability**: The app will integrate seamlessly with external systems such as payment gateways and APIs.
* **Portability**: The app will be compatible with both Android and iOS platforms.
* **Testability**  
  The app will be designed for thorough testing, with high levels of test coverage.
* **Adaptability**  
  The app will be able to adapt to future updates, user preferences, and market trends.

# Other Requirements

## Database Tables

### User Table:

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| user\_id | INT (PK) | Unique user identifier |
| email | VARCHAR(255) | Email address (unique) |
| password | VARCHAR(255) | Hashed password |
| Phone number | VARCHAR(255) | User's phone number |
| name | VARCHAR(255) | Full name of the user |
| Create\_at | TIMESTAMP | Account creation timestamp |
| Updated\_at | TIMESTAMP | Last profile update timestamp |
| Date\_of\_birth | date | Date of birth |

**Attributes:** user\_id, Email,password,phone number**,** name , create\_at , update\_at , Date\_of \_ birth.

**Description:** The User table holds essential information about each user in the system, uniquely identified by user\_id. It includes personal details like name, email, phone number, and date\_of\_birth, as well as security details such as password. The table also tracks account activity with created\_at and updated\_at timestamps, which record when the account was created and last updated.

**7.1.2 Product table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| product\_id | (INT, Primary Key) | Unique identifier for each product. |
| name | (VARCHAR) | Product name |
| description | (TEXT) | Detailed product description. |
| price | (DECIMAL) | Product price. |
| stock\_quantity | (INT) | Quantity in stock. |
| category\_id | (INT, Foreign Key | Links to the Categories table. |
| created\_at | (DATETIME) | Date added to the catalog. |

**Attributes:** product\_id (INT, Primary Key) , name , description (TEXT), price (DECIMAL) , stock\_quantity (INT), category\_id (INT, Foreign Key) , created\_at (DATETIME).

**Description:**

The Products table stores information about each product available for sale. Each product entry has a unique product\_id, along with details such as name, description, price, and stock\_quantity. Additional fields may include category, brand, and SKU for organization and inventory tracking. This table is essential for managing the product catalog, allowing for efficient updates to pricing, stock levels, and product information.

**7.1.3 Categories Table :**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| category\_id | (INT, Primary Key) | Unique identifier for each category. |
| description | Text | Description of the category |
| parent\_id | (INT, Foreign Key) | Parent category for nested categories (if needed)**.** |
| name (VARCHAR) | (VARCHAR) | Category name |

**Attributes:** category\_id , description , parent\_id , name (VARCHAR).

**Description:**

The Categories table organizes products into distinct groups, enhancing product navigation and search functionality. Each category entry has a unique `category\_id` and includes fields such as `name` and `description` to specify the category's purpose. Parent-child relationships within categories may also be represented, allowing for nested categories and subcategories. This structure enables efficient product classification and improves the user experience when browsing or filtering products.

**7.1.4 Orders Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| user\_id | (INT, Foreign Key) | ID of the user placing the order |
| order\_date | (DATETIME) | Date and time of order. |
| status | (VARCHAR) | Order status (Pending, Shipped, Delivered, etc.). |
| order\_id | (INT, Primary Key) | Unique identifier for each order. |
| total\_amount | (DECIMAL | Total cost for the order. |
| shipping\_id | (INT, Foreign Key) | ID from the Shipping table. |

**Attributes:**

order\_id (INT, Primary Key) , user\_id (INT, Foreign Key) , order\_date (DATETIME) , status(VARCHAR) , total\_amount (DECIMAL) , shipping\_id (INT, Foreign Key) .

**Description:**

The Orders table captures all essential details of customer orders. Each order entry is uniquely identified by an `order\_id` and includes a link to the `user\_id` of the customer who placed the order. Key fields include `order\_date`, `total\_amount`, and `status` (e.g., Pending, Shipped, Delivered, Cancelled) to track the order’s lifecycle. The table may also reference `shipping\_id` and `payment\_id` for associated shipping details and payment transactions, facilitating comprehensive order management from placement through fulfillment.

**7.1.5 Order Items Table**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Order\_item\_id | (INT, primary key) | Unique identifier for each order item. |
| Order\_id | (INT, Foreign Key) | Links to the Orders table. |
| Product\_id | (INT, Foreign Key) | ID of the product. |
| quantity | (INT) | Order id |
| price | (DECIMAL) | Select the price |

**Attributes:**

order\_item\_id (INT, Primary Key) , order\_id (INT, Foreign Key) , product\_id (INT, Foreign Key) ,quantity (INT) , price (DECIMAL)

**Description:**

The Order\_Items table records specific details about each product included in an order. Each entry is identified by a unique `order\_item\_id` and links to both an `order\_id` (from the Orders table) and `product\_id` (from the Products table) to specify the relationship between orders and individual products.

**7.1.6 Payments Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data type | Description |
| order\_id | (INT, Foreign Key): | Associated order ID. |
| payment\_date | (DATETIME) | Payment date. |
| amount | (DECIMAL) | Payment amount |
| payment\_method | (VARCHAR): | Method (Credit Card, PayPal, etc.). |
| Payment\_id | (Int,primary key) | Unique identifier for each payment |

**Attributes:**

payment\_id (INT, Primary Key) , order\_id (INT, Foreign Key) ,payment\_date (DATETIME), amount (DECIMAL) , payment\_method (VARCHAR) .

**Description:**

The Payments table stores information related to customer payments for orders. Each entry has a unique `payment\_id` and is linked to an `order\_id` to associate the payment with a specific order. Key fields include `payment\_date`, `amount`, and `payment\_method` (e.g., Credit Card, PayPal, Bank Transfer) to detail the transaction. The `status` field (e.g., Completed, Pending, Failed) tracks the payment’s current state. This table ensures accurate recording of payment information and helps manage transaction statuses across the order lifecycle.

**Shipping Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Shipping\_id | INT, Primary Key | Unique identifier for each shipping entry. |
| Order\_id | INT, Foreign Key | Associated order. |
| Shipping\_date | DATETIME | Date when the order was shipped. |
| Delivery\_date | DATETIME | Expected delivery date. |
| Status | VARCHAR | e.g., In Transit, Delivered |
| Shipping\_address | TEXT | Delivery address for the order. |

**Attributes:** Shipping ID, Order ID, Shipping Date, Delivery Date, Status, Shipping Address.

**Description:** The **Shipping** object represents the shipping details for an order. Each shipping entry has a unique ID and is linked to a specific order. It includes the shipping date, expected delivery date, the current status (e.g., In Transit, Delivered), and the delivery address.

**Reviews Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Review\_Id | INT, Primary Key | Unique identifier for each review**.** |
| User\_Id | INT, Foreign Key | ID of the user reviewing. |
| Product\_Id | INT, Foreign Key | ID of the product. |
| Rating | INT | Rating out of 5. |
| Comment | TEXT | Review text. |
| Created\_At | DATETIME | Review creation date. |

**Attributes:** Review ID, User ID, Product ID, Rating, Comment, Created At.

**Description**: Represents customer reviews for products, including a unique identifier for each review, the user who wrote the review, the product being reviewed, the rating given, the review text, and the date the review was created.

**Promotions Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Promo\_Id | INT, Primary Key | Unique identifier for each promotion. |
| Code | VARCHAR | Promotion or discount code. |
| Discount\_Percent | DECIMAL | Discount percentage. |
| Start\_Date | DATETIME | Start date of the promotion. |
| End\_Date | DATETIME | End date of the promotion. |
| Description | TEXT | Brief description of the promotion. |

**Attributes:** Promo ID, Code, Discount Percent, Start Date, End Date, Description.

**Description**: Represents promotional offers, including a unique identifier for each promotion, the promotion or discount code, the discount percentage, the start and end dates of the promotion, and a brief description of the promotion.

**Product\_Images Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Image\_Id | INT, Primary Key | Unique identifier for each image. |
| Product\_Id | INT, Foreign Key | Associated product ID. |
| Image\_Url | TEXT | URL to the image file. |
| Alt\_Text | VARCHAR | Alt text for accessibility. |

**Attributes:** Image ID, Product ID, Image URL, Alt Text.

**Description**: Represents images associated with products, including a unique identifier for each image, the related product, the URL to the image file, and alt text for accessibility.

**Wishlist Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Wishlist\_Id | INT, Primary Key | Unique identifier for each wishlist. |
| User\_Id | INT, Foreign Key | ID of the user who owns the wishlist. |
| Product\_Id | INT, Foreign Key | ID of the product in the wishlist. |
| Added\_Date | DATETIME | Date the product was added. |

**Attributes:** Wishlist ID, User ID, Product ID, Added Date.

**Description**: Represents a user's wishlist, including a unique identifier for each wishlist, the user who owns the wishlist, the product added to the wishlist, and the date the product was added.

**Cart Table:**

|  |  |  |
| --- | --- | --- |
| Attributes | Data Type | Description |
| Cart\_Id | INT, Primary Key | Unique identifier for each cart item. |
| User\_Id | INT, Foreign Key | ID of the user. |
| Product\_Id | INT, Foreign Key | Product added to the cart. |
| Quantity | INT | Quantity of the product in the cart. |
| Added\_Date | DATETIME | Date the item was added to the cart. |

Appendix A: Glossary

* **Use Case**  
  Describes a scenario that shows the interaction between the system and users (actors) to accomplish a specific goal, outlining system behavior in response to user actions.
* **Misuse Case**  
  A variant of use cases that models potential incorrect or malicious actions by users, helping to identify vulnerabilities in the system.
* **Class Diagram**  
  A type of structural diagram in UML that represents the system’s classes, their attributes, methods, and relationships between objects.
* **Data Flow Diagram (DFD)**  
  A graphical tool that models the flow of data through a system, showing processes, data stores, and data interactions.
* **Entity-Relationship (ER) Diagram**  
  A diagram used to model the data structure of the system, showing entities, their attributes, and the relationships between them.
* **Component Diagram**  
  A UML diagram that models the system’s physical components (e.g., software modules, databases), showing their interactions and dependencies.
* **State Diagram**  
  Represents the states an object can be in, and the transitions between these states triggered by events or actions, useful for modeling dynamic behavior.
* **Sequence Diagram**  
  A UML diagram that illustrates how objects interact in a specific sequence, showing the order of messages exchanged between them.
* **Activity Diagram**  
  A behavioral diagram that models the flow of activities or actions within a process or a system, often used to represent business workflows.
* **Functional Requirements**  
  Defines specific functions or features that the system must support, such as processing transactions, user authentication, or generating reports.
* **Non-functional Requirements**  
  Describes the system’s operational attributes, such as performance, security, scalability, and usability, focusing on how the system performs rather than what it does.
* **User Interface (UI) Design**  
  The layout and interaction design of the user-facing part of the system, focusing on how users will navigate and interact with the application.
* **System Architecture**  
  The structural framework of a system, detailing how various components interact and function to meet the system's overall goals.
* **Security Requirements**  
  Specifies the security measures necessary to protect the system from unauthorized access, data breaches, and other security threats, including encryption and authentication.
* **Performance Requirements**  
  Defines the performance criteria for the system, including response time, throughput, scalability, and reliability under varying load conditions.
* **System Flow**  
  Describes the sequence of actions or processes within the system, focusing on how inputs are transformed into outputs throughout the system.
* **Testing Requirements**  
  Specifies the criteria and types of tests required to ensure the system meets functional, non-functional, and security requirements, including unit, integration, and acceptance tests.
* **Stakeholders**  
  Individuals or groups who have an interest in the system’s development, including end-users, project managers, developers, and clients.
* **Authentication**  
  The process by which a system verifies the identity of a user, often involving the use of usernames, passwords, or biometric data.
* **Authorization**  
  Defines the permissions granted to a user after authentication, determining which resources or functions they can access.
* **Scalability**  
  The ability of a system to handle an increasing amount of work or to be enlarged to accommodate that growth.
* **Availability**  
  The percentage of time that a system is operational and accessible, ensuring minimal downtime and ensuring continuity of service.
* **Usability**  
  Refers to the ease with which users can learn and operate the system, ensuring a positive user experience with clear and intuitive interfaces.
* **Reliability**  
  Describes the system's ability to perform consistently without failure over a period, ensuring high uptime and accuracy of operations.
* **Maintainability**  
  The ease with which a system can be modified to fix defects, improve performance, or adapt to new environments.
* **Portability**  
  The system’s ability to operate on different platforms with minimal changes, ensuring compatibility across various devices and operating systems.
* **Interoperability**  
  The ability of the system to work with other systems or technologies to exchange data or perform tasks across platforms.
* **Reuse**  
  The ability to use existing software components or modules in new applications or systems to save time and reduce development effort.
* **Modularity**  
  The design principle of breaking down a system into smaller, self-contained components or modules, allowing for easier development and maintenance.
* **Backup and Restore**  
  Procedures to create copies of critical data and systems for recovery purposes in the event of a failure or loss.
* **API (Application Programming Interface)**  
  A set of protocols and tools that allow different software applications to communicate and interact with each other, enabling integration and functionality sharing.
* **Session Management**  
  The process of keeping track of a user's interactions with the system over time, typically using session identifiers to maintain user states during browsing or transactions.

Appendix B: Analysis Models

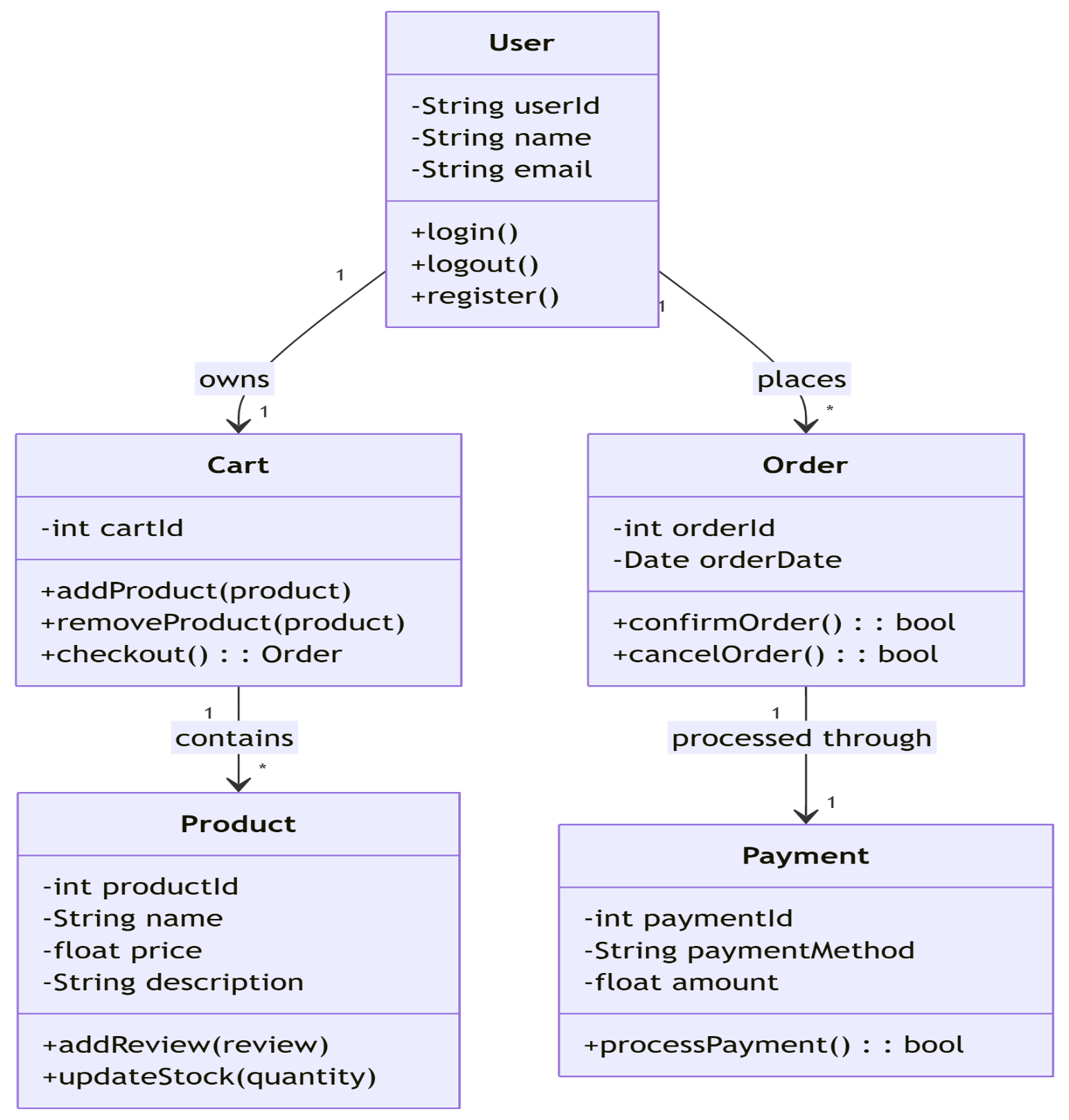


Figure 1: Class Diagram.

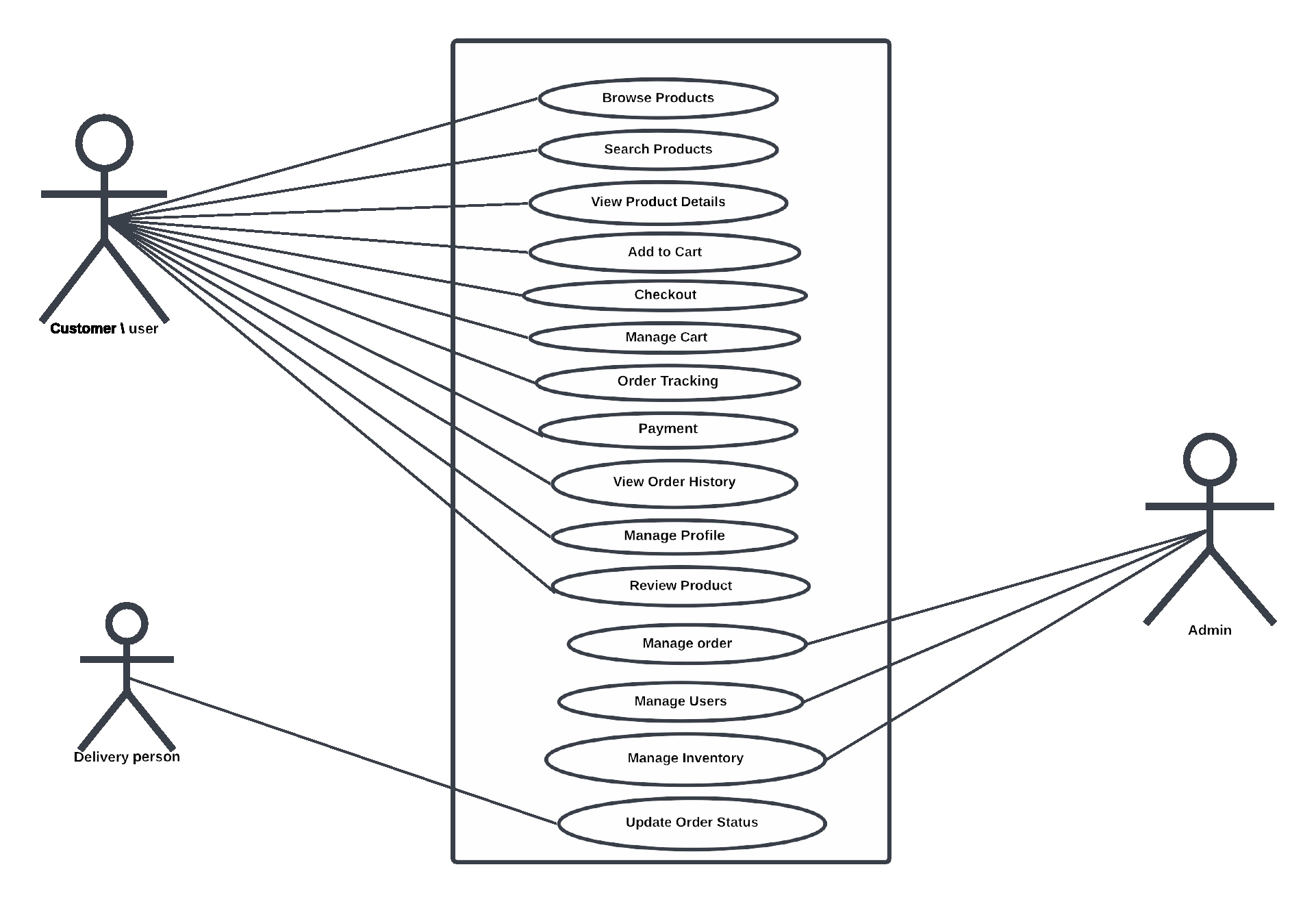


Figure 2: Use Case Diagram.

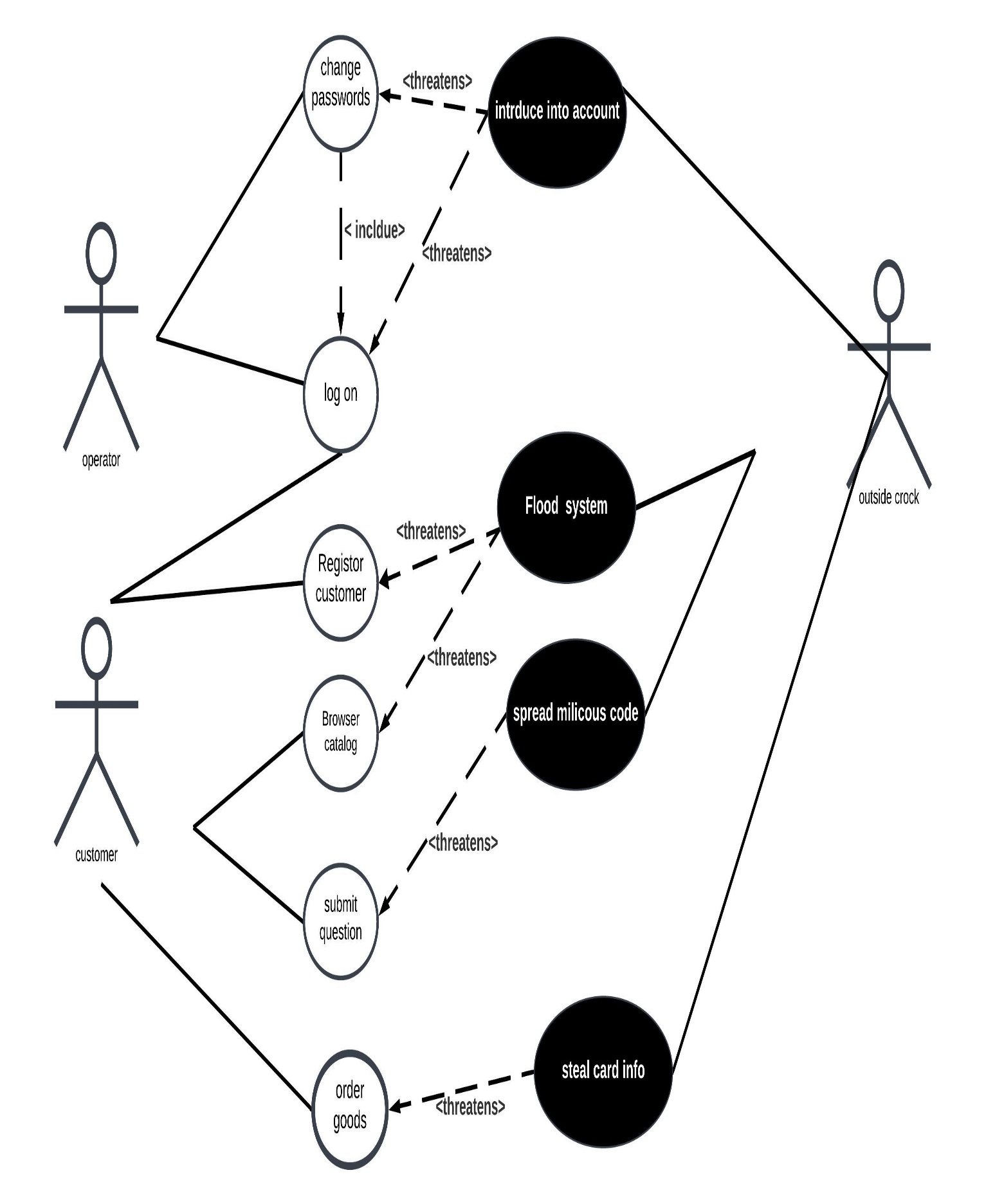


Figure 3: Miss Use-Case Diagram.

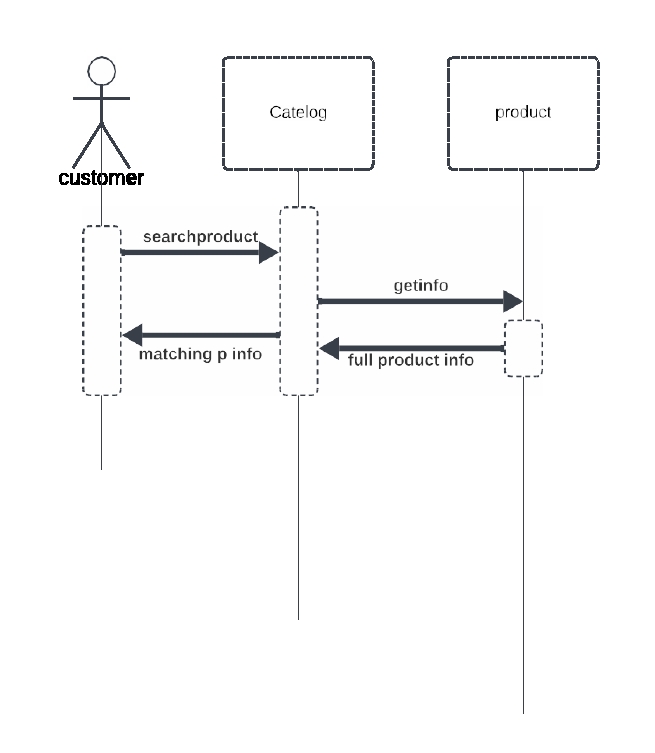


Figure 4: Sequence Diagram.

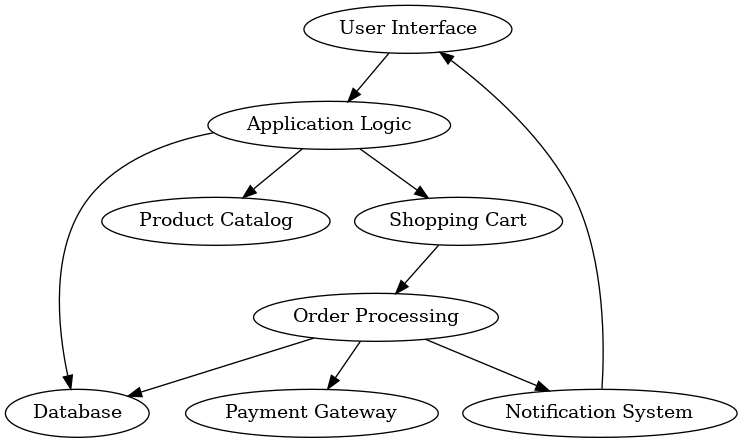


Figure 5: Static Diagram.

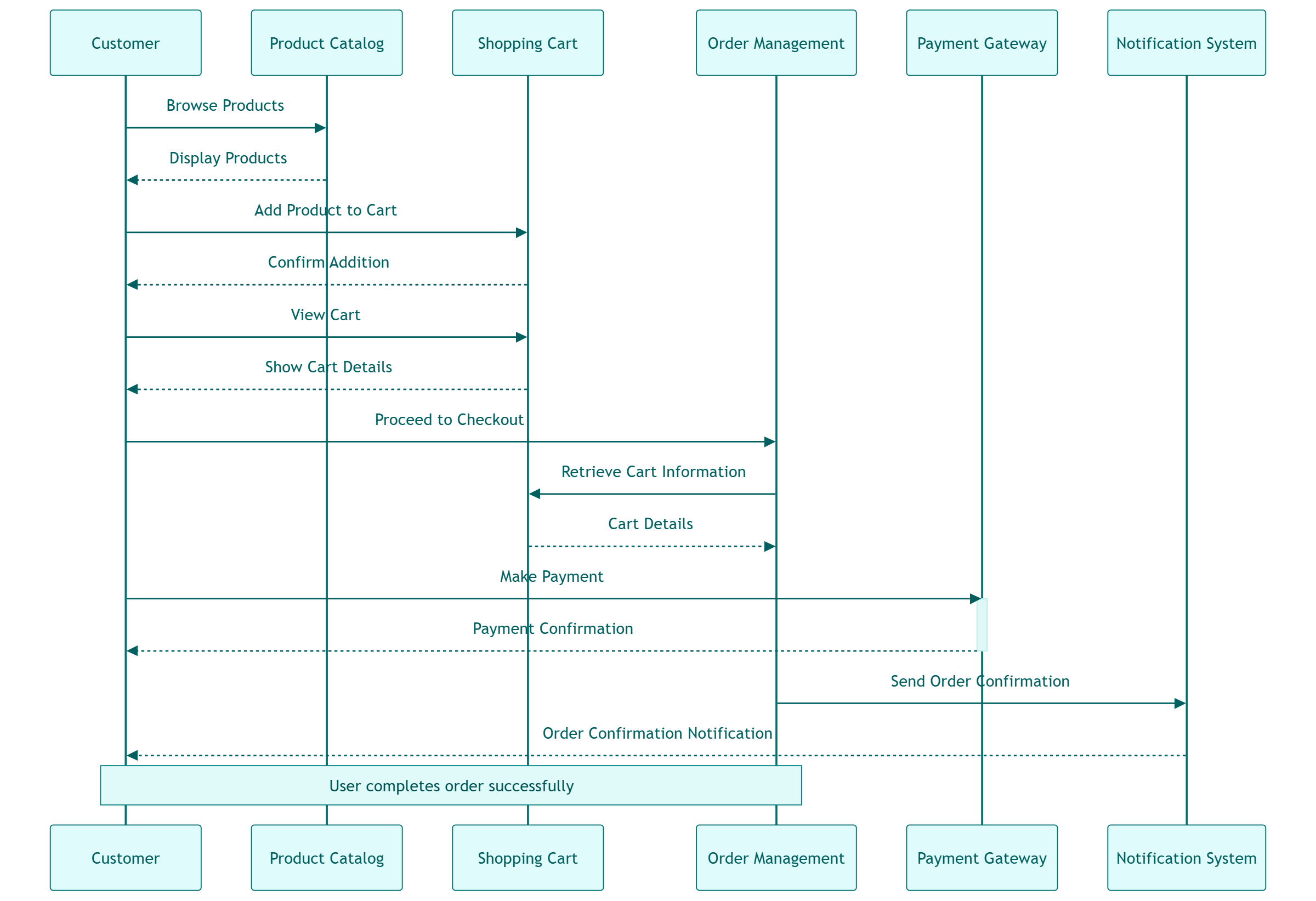


Figure 6: Collaboration Diagram.

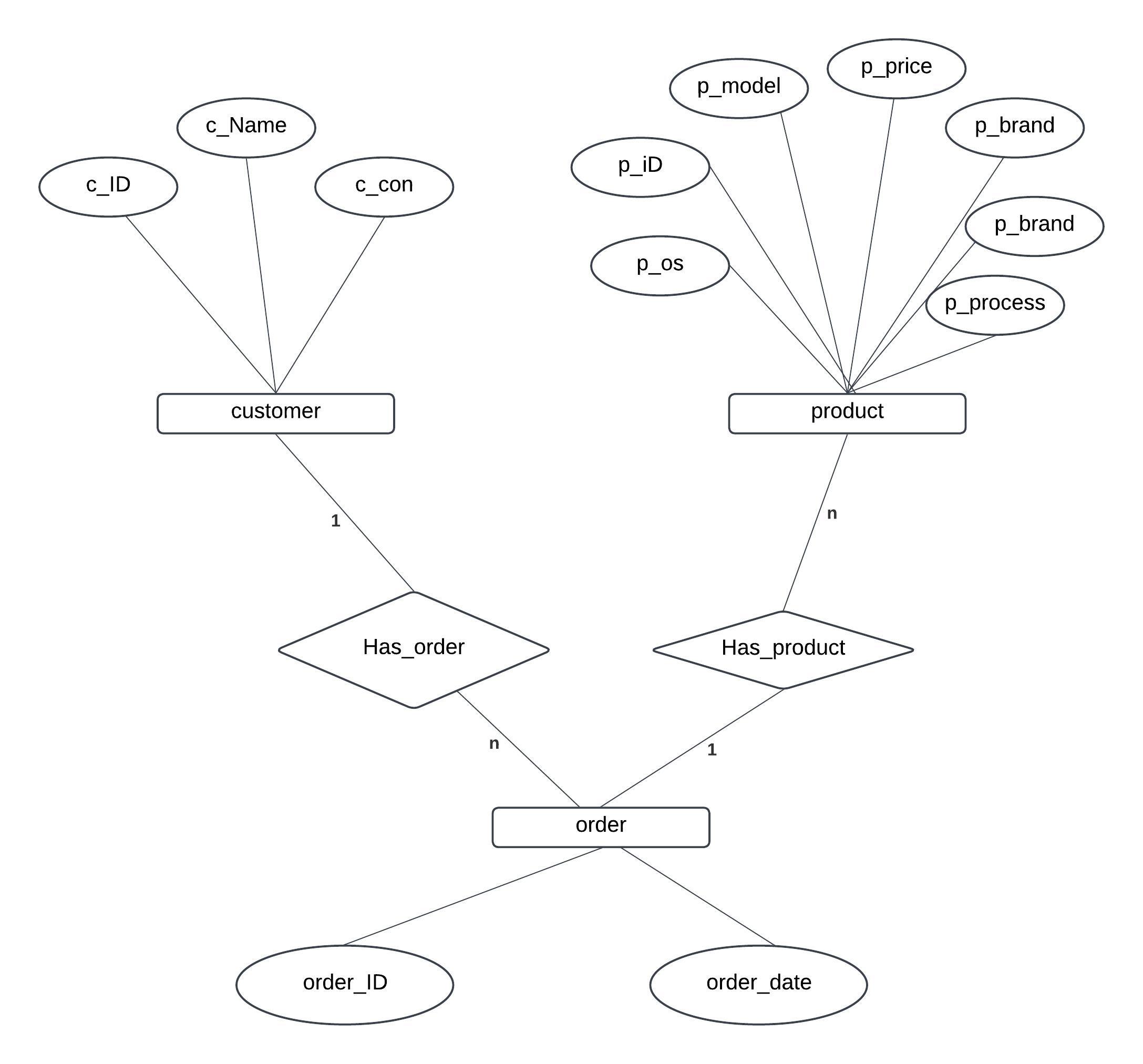


Figure 7: ER- Diagram.

Appendix C: Design Models

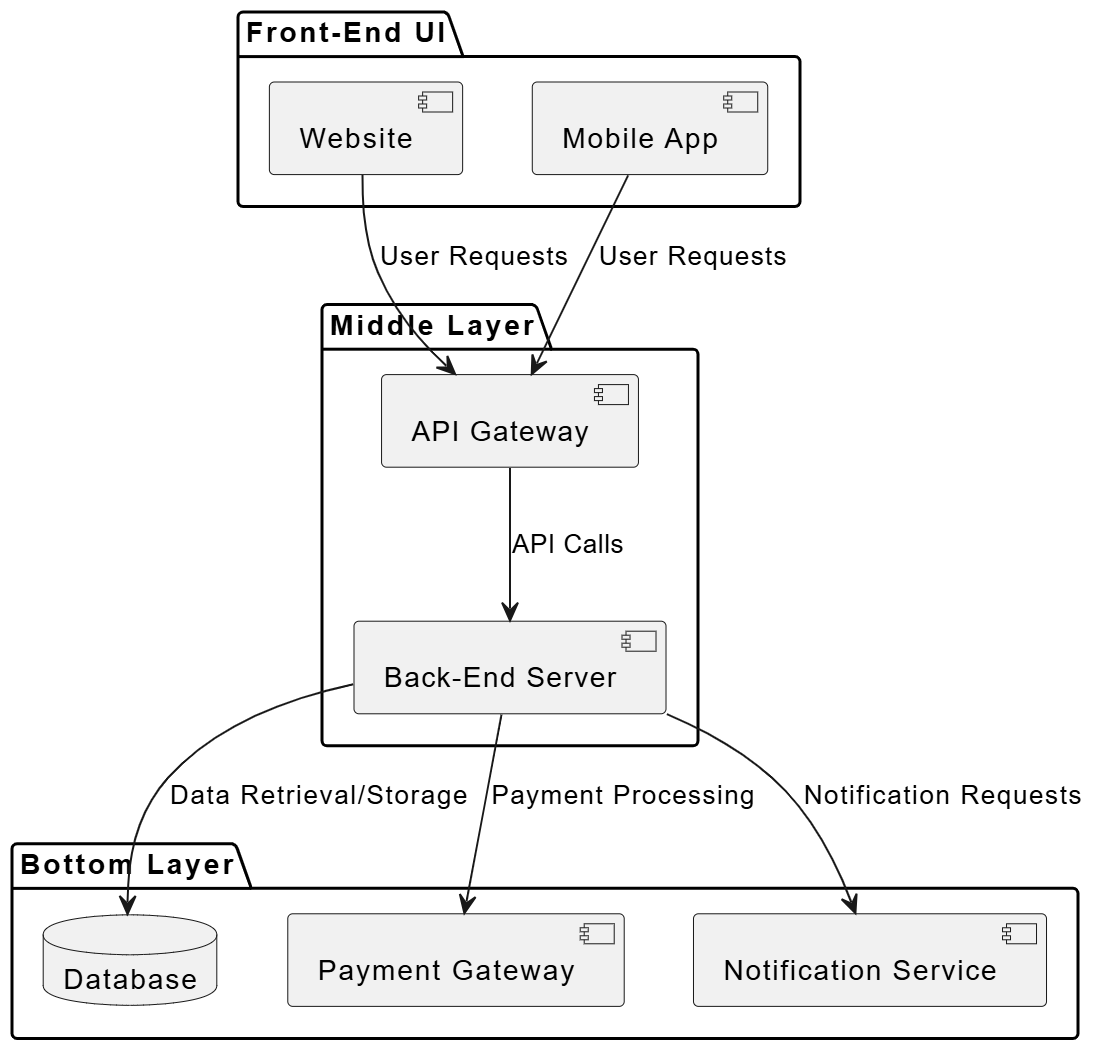


Figure 8: Component Diagram.

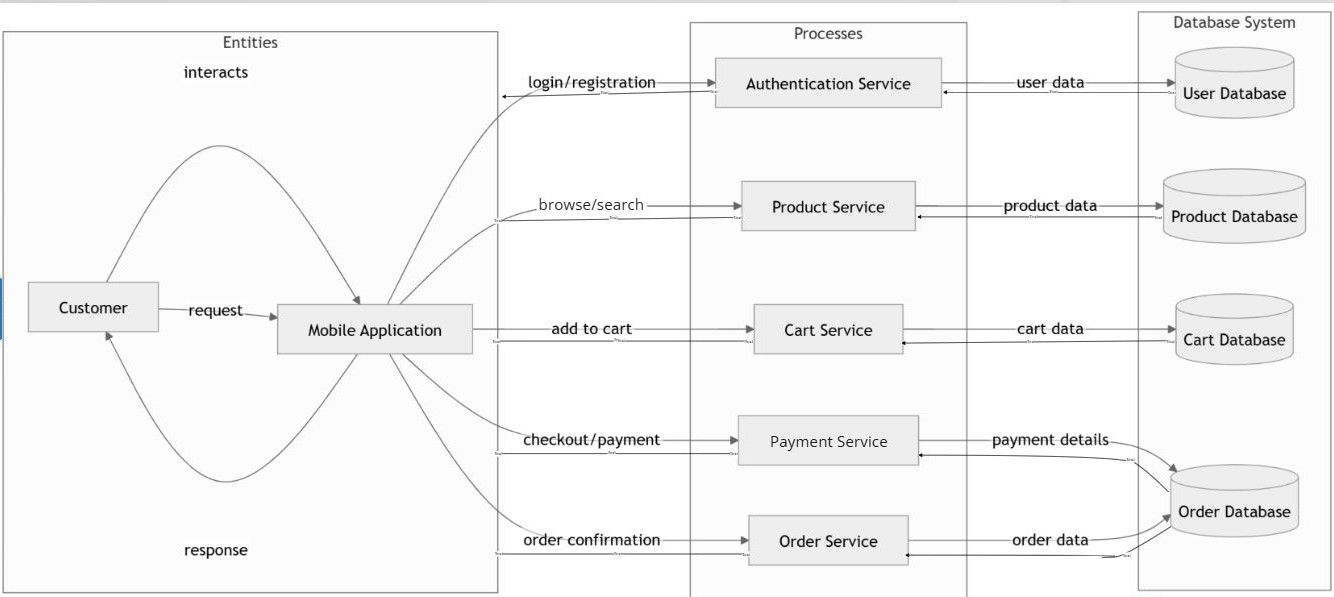


Figure 9: Data Flow Diagram.

Appendix D: Screenshots

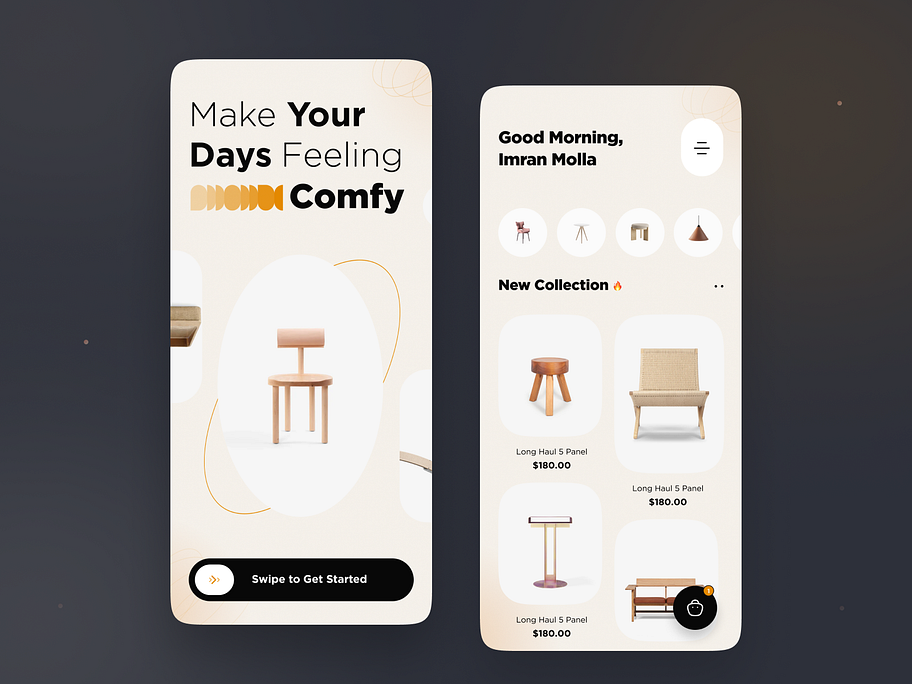


Image 1: Splash Screen

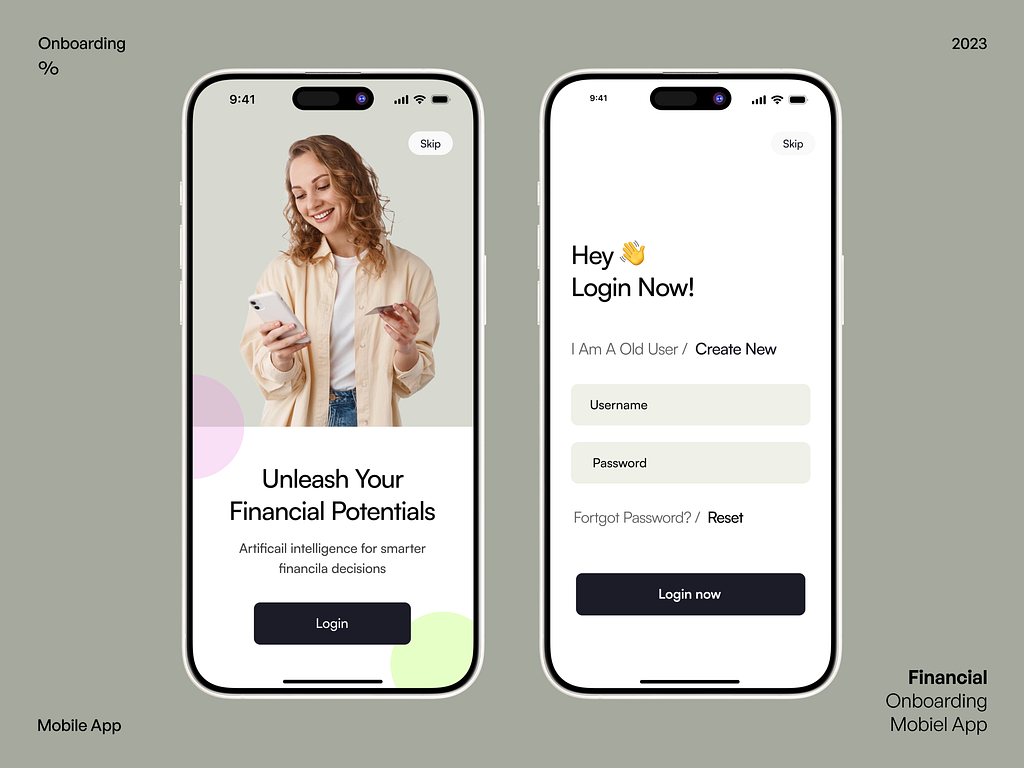


Image 2: Login Screen

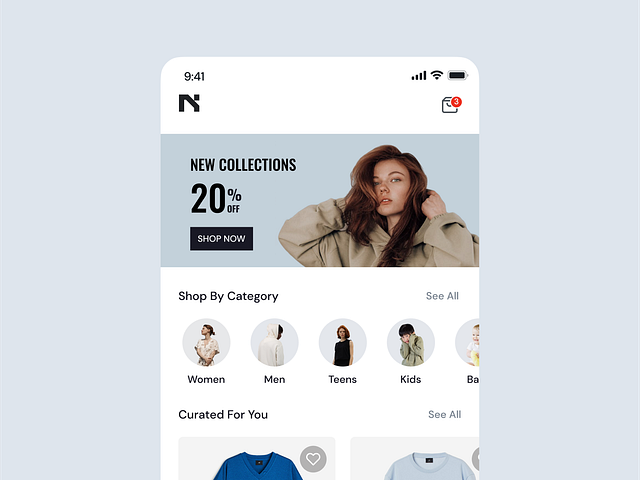


Image 3: Home Screen

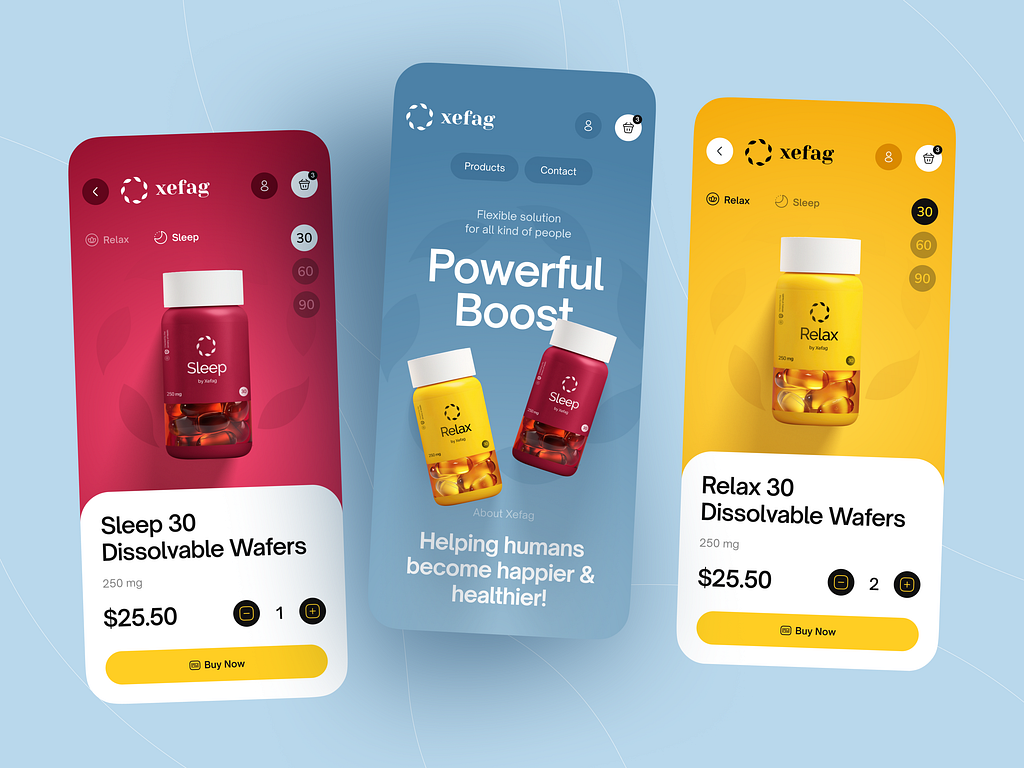


Image 4: Products Screen

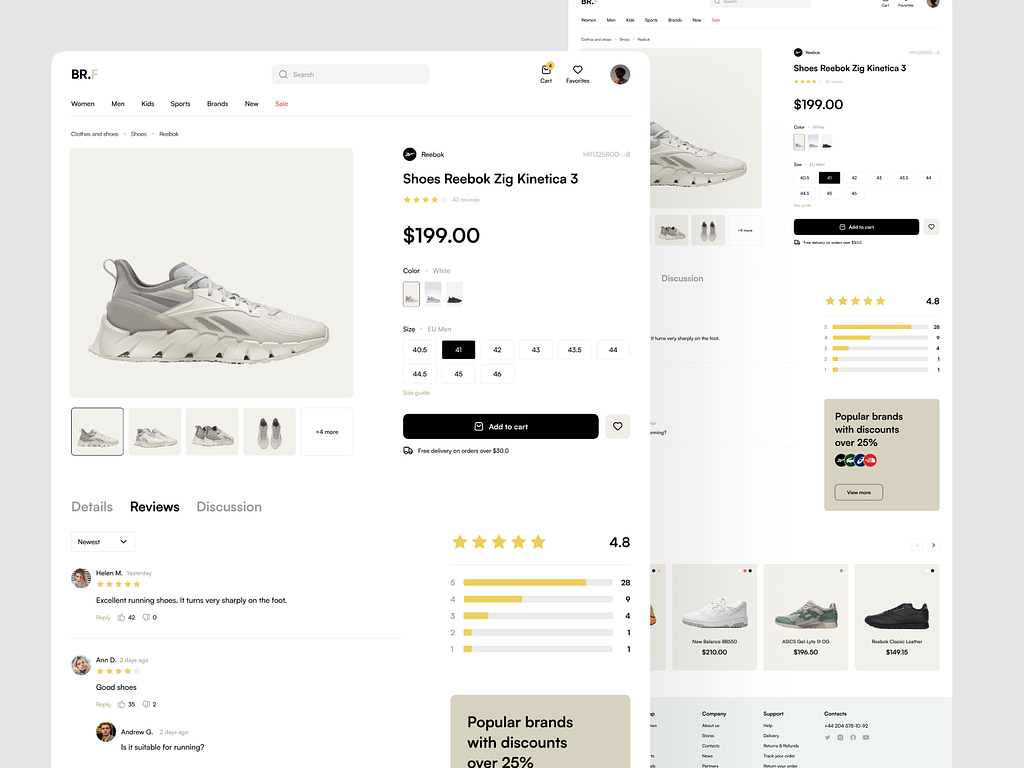


Image 5: Product Screen

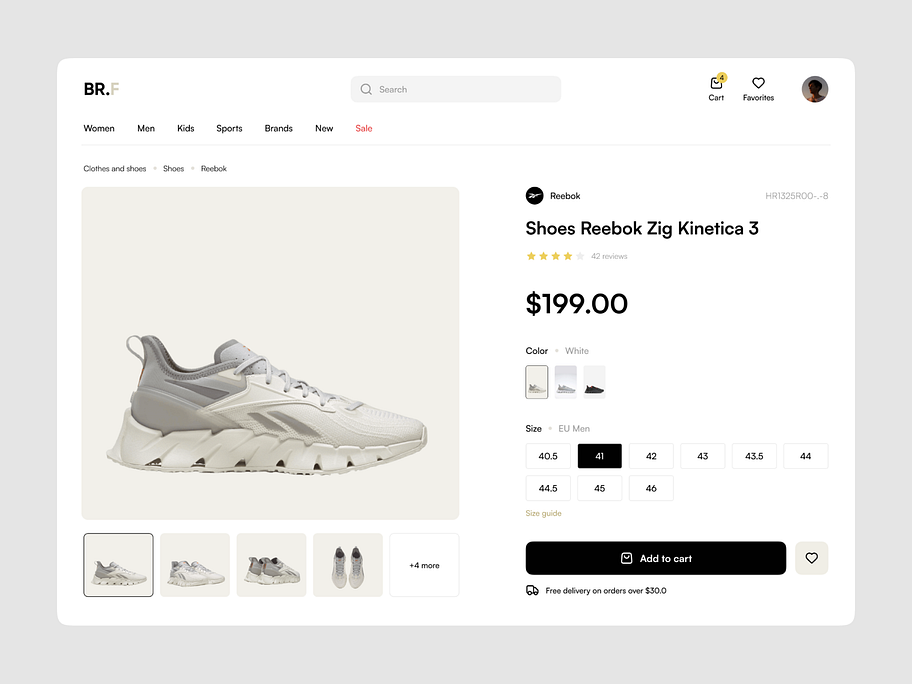


Image 6: Product Info Screen

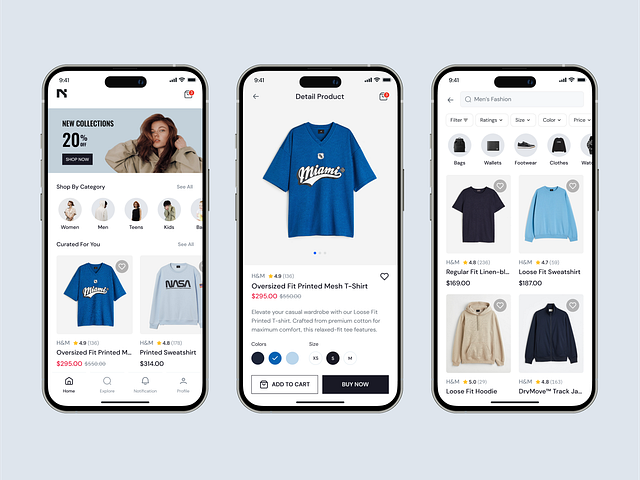


Image 7: Add to Cart Screen

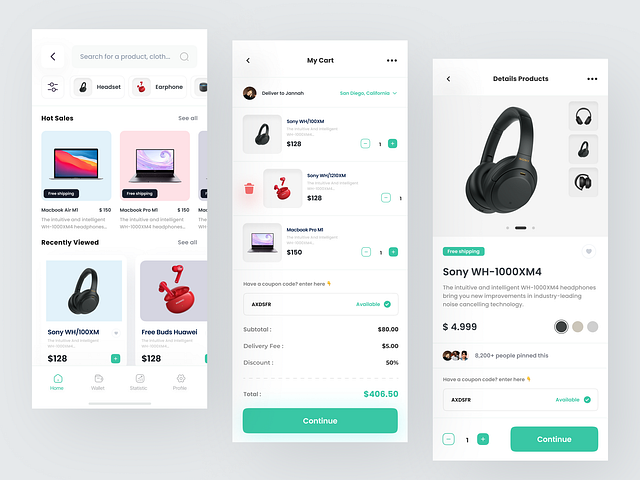


Image 8: Checkout Screen

Appendix E: Test Cases

|  |  |
| --- | --- |
| **Identifier** | TC-01 |
| **Priority** | High |
| **Related requirements(s)** | UC-0001 |
| **Short description** | Test the user registration process with valid data. |
| **Pre-condition(s)** | -The app is installed and running.  -The "Sign Up" page is accessible. |
| **Input data** | - User Name: Ameer Hamza  - Email: hamza@example.com  - Password: Pass@123  - Address: 456 Example Road  - Gender: male  - Postal Code: 67890 |
| **Detailed steps** | 1. Launch the app.  2. Open the "Sign Up" page.  3. Fill in the registration form.  4. Click the "Register" button. |
| **Expected result(s)** | - A success message is displayed: "Registration Successful."  - User data is stored in the database.  - User is redirected to the login page or dashboard. |
| **Post-condition(s)** | - User is registered.  - User can log in with the entered credentials. |

Table 2: Registeration

|  |  |
| --- | --- |
| **Identifier** | TC-02 |
| **Priority** | High |
| **Related requirements(s)** | UC-0001 |
| **Short description** | Test the login process with valid credentials. |
| **Pre-condition(s)** | - The app is installed and running.  - The user has already registered with valid credentials. |
| **Input data** | - Email: hamza@example.com  - Password: Pass@123 |
| **Detailed steps** | 1. Launch the app. 2. Open the "Login" page. 3. Enter the email and password. 4. Click the "Login" button. |
| **Expected result(s)** | - User is successfully logged in and redirected to the dashboard. |
| **Post-condition(s)** | - User is logged in.  - User can access the features available to them. |

Table 2: Login

|  |  |
| --- | --- |
| **Identifier** | TC-03 |
| **Priority** | High |
| **Related requirements(s)** | UC-0002 |
| **Short description** | Test the product search functionality with a valid keyword. |
| **Pre-condition(s)** | - The app is installed and running.  - The product database contains items for searching. |
| **Input data** | - Search Term: "Smartphone" |
| **Detailed steps** | 1. Launch the app.  2. Open the "Search" feature.  3. Enter the search term in the search bar.  4. Click the "Search" button. |
| **Expected result(s)** | - A list of products matching the search term is displayed. |
| **Post-condition(s)** | - Products are displayed correctly based on the search term. |

Table 2: Product Searching

|  |  |
| --- | --- |
| **Identifier** | TC-04 |
| **Priority** | High |
| **Related requirements(s)** | UC-0003 |
| **Short description** | Test the add-to-cart functionality with valid products. |
| **Pre-condition(s)** | - The app is installed and running.  - User is logged in.  - The product is available in the catalog. |
| **Input data** | - Product: "Smartphone" |
| **Detailed steps** | 1. Launch the app.  2. Browse products.  3. Click on a product.  4. Click the "Add to Cart" button. |
| **Expected result(s)** | - Product is added to the cart.  - Cart icon shows the updated number of items. |
| **Post-condition(s)** | - The item is successfully added to the cart. |

Table 2: Shopping Cart

|  |  |
| --- | --- |
| **Identifier** | TC-05 |
| **Priority** | High |
| **Related requirements(s)** | UC-0004 |
| **Short description** | Test the checkout process with valid payment and shipping information. |
| **Pre-condition(s)** | - The app is installed and running.  - User is logged in.  - User has items in the cart. |
| **Input data** | - Shipping Address: 123 Main Street  - Payment Method: Credit Card |
| **Detailed steps** | 1. Launch the app.  2. Open the cart.  3. Proceed to checkout.  4. Fill in shipping and payment details.  5. Confirm order. |
| **Expected result(s)** | - The user is shown an order summary with all details.  - Order is confirmed and the payment is processed. |
| **Post-condition(s)** | - Order is placed successfully.  - User is shown a confirmation page. |

Table 2: Checkout Process

|  |  |
| --- | --- |
| **Identifier** | TC-06 |
| **Priority** | Medium |
| **Related requirements(s)** | UC-0005 |
| **Short description** | Test the user review and rating functionality for a product. |
| **Pre-condition(s)** | - The app is installed and running.  - User is logged in.  - Product is available for review. |
| **Input data** | - Product: "Smartphone"  - Rating: 4 stars  - Review: "Great phone, excellent features!" |
| **Detailed steps** | 1. Launch the app. 2. Browse and select a product. 3. Scroll down to the reviews section.  4. Click the "Write a Review" button.  5. Enter the review and rating.  6. Submit the review. |
| **Expected result(s)** | - The review is posted successfully.  - Rating is updated on the product page. |
| **Post-condition(s)** | - The product has an updated review and rating. |

Table 2: Review and Ratings

|  |  |
| --- | --- |
| **Identifier** | TC-07 |
| **Priority** | Medium |
| **Related requirements(s)** | UC-0006 |
| **Short description** | Test the functionality of adding, viewing, and managing items in the wishlist. |
| **Pre-condition(s)** | The user must be logged in to the system. |
| **Input data** | Product ID, User ID. |
| **Detailed steps** | 1. Navigate to the product page. 2. Click the "Add to Wishlist" button. 3. Navigate to the Wishlist section from the user profile. 4. Verify the added product appears in the wishlist. 5. Remove an item from the wishlist. 6. Verify the wishlist updates correctly. |
| **Expected result(s)** | 1. Product is successfully added to the wishlist. 2. Wishlist displays the correct items. 3. Removed items no longer appear in the wishlist. |
| **Post-condition(s)** | The updated wishlist is saved to the database and is accessible for future sessions. |

Table 2: Wishlist

|  |  |
| --- | --- |
| **Identifier** | TC-08 |
| **Priority** | Medium, |
| **Related requirements(s)** | UC-0007 |
| **Short description** | Test the location-based deals feature with valid postal code. |
| **Pre-condition(s)** | - The app is installed and running.  - User is logged in.  - The user’s postal code is registered. |
| **Input data** | - Postal Code: 62300 for BWN |
| **Detailed steps** | 1. Launch the app.  2. Open the homepage.  3. Wait for location-based deals to load.  4. View the deals available for the entered postal code. |
| **Expected result(s)** | - The user sees relevant deals based on their postal code. |
| **Post-condition(s)** | - The user is shown location-specific deals. |

Table 2: Location-Based Deals

|  |  |
| --- | --- |
| **Identifier** | TC-09 |
| **Priority** | High |
| **Related requirements(s)** | UC-0008 |
| **Short description** | Test the logout functionality to ensure the user is logged out correctly. |
| **Pre-condition(s)** | - The app is installed and running.  - User is logged in. |
| **Input data** | - None (Just the action of logging out) |
| **Detailed steps** | 1. Launch the app.  2. Open the user profile or settings page.  3. Click the "Logout" button. |
| **Expected result(s)** | - The user is logged out successfully.  - User is redirected to the login page. |
| **Post-condition(s)** | - User is logged out.  - User cannot access logged-in features without logging in again. |

Table 2: Logout

References

**Journal Reference**

1. Davis, A., & Lee, J. (2021). E-commerce personalization and customer retention. Journal of Digital Commerce, 10(2), 45–56.
2. Simmons, R. (2023). The impact of affordability on online retail platforms. Journal of Business Strategy, 8(1), 30–42.
3. Nguyen, T., & Tran, M. (2020). Geolocation technology in e-commerce: A review. International Journal of Computing, 15(3), 88–101.
4. Anderson, K., & White, H. (2020). Custom-built e-commerce solutions: A competitive edge. International Business Review, 22(4), 67–79.
5. Zhang, Y., & Wang, S. (2022). Optimizing mobile e-commerce app interfaces for better user experience. Journal of Mobile E-Commerce, 14(3), 180-194.
6. Khan, M. S., & Raza, M. S. (2020). Mobile commerce trends and their impact on e-commerce platforms. E-Commerce and Technology Review, 19(2), 122-136.

**Books:**

1. Ian Sommerville, Software Engineering 8th edition. Pearson education,2008
2. Trafficspecial – john resse,E\_commerce website.2003.