COMP 354 Intro to Software Development
Fall 2023

Assignment #2
Software Requirements Specification Document
Shopping Cart App

Weilun Zhang 40190549
Jingchao Song 40159533
ZhaoYang Liu 40157299
Haoran Sun 40149349
Qianrui Tao 40162182
Xu Zhang 40169981

Due date: Oct 27th, 2023

Introduction:

The purpose of this document is to design a higher quality e-commerce program and apply it to a shopping website. This website allows users to purchase books, learning tools or other goods. The e-commerce program is designed to provide users with a better shopping experience. Experience can recommend relevant products based on the items users usually search and browse, thereby greatly improving consumers' shopping experience and consumption desire. Currently, shopping websites lack personalized product recommendations and insufficient promotion of stranded products. This e-commerce program the application is intended to achieve a benign consumption cycle relationship between consumers and products. The details of how the software meets these requirements are elaborated in use cases and supplementary specifications to influence organizational strategy.

1.1 Scope

The goal behind creating the Shopping cart web is to enable users to effortlessly discover books that align with their interests and requirements. Additionally, it offers users enhanced and more tailored tools for managing their reading lists. These three new functionalities will empower users to refine their search and locate a book that suits them more effectively. Furthermore, these added features will streamline the search process, ultimately saving users time and making their search efforts more efficient.

1.2 Definition and Acronyms

Trems	Definitions
MUI	Material User Interface
SQL	Structured Query language
GPL	General public license
UI	User Interface
API	Application programming Interface
ISBN	International Standard Book Number

1.3 References

Reference ID	Title	Reference
1.	Designing systems and software applications by using models [IBM]	https://www.ibm.com/docs/en/rational-soft-arch/9.5?topic=modeling-designing-systems-software-applications-by-using-models
2.	System: Online Shopping Cart (E-commerce website)	https://github.com/shashirajraja/shopping-cart

2.Positioning

2.1 Problem Statement

The problem of	What is the problem? At present, the shopping system only allows users to display targeted products, and the practicality and personalization of functions are very lacking.
Affects	Who is getting affected and how ? When consumers use this website to shop, they will be pushed to products that are frequently viewed and related.
The impact of which is	What is the impact of the current system? The current system will greatly reduce consumers' shopping experience and desire to shop.
A successful solution would be	What is the successful solution/system- to be? We hope that the system can push relevant products by analyzing the products that users have previously browsed or retrieved, and push discounted products based on the personal information provided by users to improve consumers' shopping experience and promote website revenue.

2.2 Product Position Statement

For	The user of system People who need to purchase goods from this website
Who	What is their need? They need to develop an e-commerce program to facilitate consumers to browse related products of interest more quickly when shopping.
System(name of the System	Website/webapp/ apk/etc
That	How does the system help the user? Push relevant products and recommend discounted products by analyzing users' browsing and purchasing preferences.
Unlike	What is lacking in the existing system? The existing system has the problem of single product display
Our product	What is provided in our product and how is it beneficial? Allow consumers to view the items and quantities in the shopping cart and change the desired quantity at will, as well as related products they like will be pushed by the system

3. Stakeholder Description

3.1 Stakeholder Summary

By understanding the roles and interests of each stakeholder group, the Concordia Cart team can ensure that the platform meets the needs of all parties involved and contributes to its overall success.

Name	**Description**	**Responsibilities**
Customer	These are the primary users of the Concordia Cart application.	Provide genuine feedback for improvements，

	They will use the application to browse, search, and purchase books, learning tools, and other goods.	Report bugs or malfunctions in the system,and ensure that they use the platform ethically and responsibly.
Product Owner	The individuals or groups who have a vested interest in the financial success of the Concordia Cart. They typically decide on features and prioritize them based on business needs.	Ensure the app meets business objectives, prioritize features and fixes and allocate resources for the development and maintenance of the application.
Developer	This group is responsible for building, testing, and maintaining the Concordia Cart application.	Develop and deploy new features,fix bugs, improve performance and ensure the security and stability of the application.
Tester	Tests the scenario according to customer requirements and suggests required changes to the developer.	Test the application against its functional requirements,ensure that features like product search, adding items to the cart, checkout process, etc., work as intended and report defects or discrepancies between the actual behavior and expected behavior.

3.2 User Summary

The detailed interaction and collaboration between these users, as well as their specific requirements, can be further detailed in use cases, user stories, and supplementary specifications.

Name	**Description**	**Responsibilities**
End Consumer	Individuals who visit the Concordia Cart website to search, browse, and purchase books, learning tools, or other goods.	Search and browse products,add desired products to the shopping cart,place orders and complete transactions,provide feedback and ratings on products,maintain their user profiles with accurate information.
Administrator	Individuals responsible for managing and overseeing the operations of the Concordia Cart website.	Handle user inquiries, complaints, and feedback.Monitor and manage transactions. Update website content and banners.Oversee promotional campaigns and strategies.

System Developer and Tester	Technical personnel responsible for the design, development, maintenance, and support of the e-commerce program and website.	Develop, test, and deploy updates and features for the website. Ensure the security and stability of the platform. Address technical issues and bugs reported by users or administrators. Backup and restore data as necessary.

3.3 User Environment

To ensure that the users of Concordia Cart have an optimal experience, it's crucial to define the environment in which the software will run efficiently. The following are the software, server, hardware, and network requirements necessary for the smooth operation of the Concordia Cart e-commerce application.

1.Customers will need a smartphone, laptop or desktop computer with a stable internet connection and browser support in order to connect and access the Concordia Cart.
2.The Concordia Cart for users is designed to be responsive and accessible across a variety of devices including: Desktops, Laptops, Tablets, and Smartphones.
3.System Administrator, Tester and System Developer need to be connected to the server environment using their laptop or desktop with an updated browser to perform their task.

4. Product Overview

4.1 Product Perspective

The Concordia Cart system is a stand-alone application, but it's placed within the larger ecosystem of e-commerce. When users interact with the system, they are not just browsing products; they are interacting with a smart platform that learns from their interactions, searches, and past purchases.

Contextual Representation of Concordia Cart:
Users: These are individuals who come to the Concordia Cart website or app with the intent of browsing or purchasing goods. They can search products, add to cart, browse recommendations, and make purchases.
Admin: Responsible for managing product listings, handling disputes, and ensuring smooth operation of the platform.
Sellers: Individuals or businesses who list their products on the platform. They can monitor their sales, get insights on popular products, and benefit from the advanced recommendation system which highlights their products to potential buyers.
Recommendation Engine: An underlying AI system that tracks user behavior, analyzes patterns, and suggests products based on user preferences, searches, and browsing history.
Checkout System: Handles user transactions, ensuring smooth and secure payment processes.

Interactions with the System:
Search & Browse: Users can search for specific items or browse through categories, and the system will display relevant products.
Personalized Recommendations: As users interact more with the platform, Concordia Cart's recommendation engine will suggest products tailored to their interests, improving their shopping experience.
Cart & Checkout: Users can add products to their cart and proceed to a secure checkout, completing their purchase.
Feedback & Reviews: Post-purchase, users can leave feedback or review products, influencing future buyers and assisting in refining the recommendation engine.
Admin & Seller Dashboards: Advanced interfaces for sellers to manage their listings and for admins to oversee platform operations.

4.2 Assumptions and Dependencies
This table outlines the basic assumptions made about the software and its corresponding dependencies. It's essential for all stakeholders to understand these points, ensuring alignment on expectations and clear communication on dependencies.

Users have access to stable internet connections.	Stable internet service providers.
Users are familiar with basic e-commerce functions like searching and checking out.	Proper user documentation and onboarding tutorials.
All products in the store have correct and updated descriptions and images.	A reliable content management system (CMS) for product information.
The recommendation system operates with minimum latency for real-time suggestions.	Efficient algorithms and server infrastructure.
The platform adheres to data protection laws and ensures users' privacy.	Legal counsel to ensure compliance with GDPR, CCPA, etc.
The website can integrate with major payment gateways for smooth transaction processes.	Partnerships or APIs from major payment providers like Stripe, PayPal.
The website is expected to function seamlessly on various devices (desktop, mobile, tablets).	Responsive web design and regular testing on different devices.

5. Features

ID	Features	Description
1.	Register /Create an account	Record user information and use a combination of email address and password to use website service.
2.	Login	Users have to login with their credentials (email address and password) and username to access web-apps.
3.	View available items in stock	Webpage displays all available items that are able to be purchased by users. It has all product details for users who might be interested in them.
4.	Search and filter item available items	Searching tools to help users find their wanted electronic products. Filters help users quickly find corresponding products, such as product prices, and so on
5.	Display most and least selling products	Users can see all the information about which product is most and least popular.
6.	Discount and promotion for users	Users could see which product is on sale. And Details of products on sale.
7.	Secondhand product	Users could check which product is used with lower cost compared with the new product. This is only available for students.
8.	Add/Remove items to shopping cart	Users may remove or add as many as products they wish to purchase and put them into the shopping cart to prepare for the payment process.
9.	Update items in Cart	Every time users add or remove a new product into their shopping cart. The numbers, price , name will automatically update and refresh at the same time.

10.	Payment method register	Users need to provide all required payment information to place their orders such as name, living address , credit card number, expiry date etc. Shopping app will retain the user's payment information for direct use next time.
11.	Place orders	Users confirm the products they wish to buy. Then, the system receives orders and is ready to ship the products to users.
12.	Check order history	Users can use it to find which products they bought in chronological order.
13.	Receipt email	Users can obtain a receipt of their orders. It contains all necessary information to their orders.
14.	Tracking shipment status	This allows users to track whether their product has been shipped or not. And estimation of time they are expected to receive it.
15.	User profile	Users can edit their username, email, zip code, address, and personal photo through it.
16.	Add/Remove product into stock	Admin can use product update form to add new or remove existing products available in online website such as price, product description, name etc. It will update the website products after adding or removing.
17.	Discount and sale	Website could generate a report of sale information. So, the admin could implement a discount strategy to promote the products based on most and least selling data.
18.	Reminder email to users	Admin could send emails to remind users that the out-of-stock product is available again on this website.
19.	Restocking alert	Admin will automatically receive a reminder about the product which is in low stocking (less than 3). It reminds the admin to restock as soon as possible.
20.	Comment/ Feedback for admin	Users can write down feedback after using this website. It allows users to contact the admin to provide their opinions about this using experience or the technical issue that they encountered when visiting. the website.

6. Quality Requirements

Requirement	Explanation
Compatibility	This shopping app can run on various web browsers such as Safari, Firefox, Chrome. It should allow users access this website through their installed browsers.
Performance	This shopping app should run at average speed at least. It is supposed to give back a response within 1 to 2 seconds when users search to implement all features in this website.
Security	Users will store their private information on this website such as personal information, banking information. It is necessary for them to prevent data breach and protect their accounts with a strong password.
User Interface	A good UI will help users use shopping app easily. They can be clearly navigated to to respect pages by using simple searching tools.
Efficiency	The shopping app should give me accurate and exact products when users and admin try to find them.
Hardware Requirement	This shopping app should be able to run on pad, phones, laptop, and desktop.
Reliability	This shopping app is supposed to be reliable when users are using it. It cannot be crashed, and it should accept a large number of users shopping at the same time. There are not supposed to have many bugs when users or admin try to implement all the features in this website.

7. Use-Cases

User:
Id: UC-01
Use Case: Login to customer account
Description:
This use case occurs when the customer wants to log in to their account.
Level: User Goal
Primary Actor: Customer
Supporting Actors: System
Stakeholders and Interests:
Customer: Users want to access their account for shopping.
System Administrator: Admin checks users' information and grants access to the account.

Developer: Develops login operation on branch employees' scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:
Customers must connect to the internet and provide valid information.
Post-Conditions:
Success end condition: Customer logs into their account successfully and views account information.
Failure end condition: Customer fails to log into their account if the username or password is not correct.
Main Success Scenario:

1. Users click on the "Login" button.
2. Users enter their email and password.
3. If the information is correct, the dashboard of the user will be displayed.
4. View information (user profile and previous orders)
5. logout

Extensions

Update user profile

1. Customer goes to the "Profile" tab.
2. System displays a window with the user's profile information (email, address, phone number, and password).
3. Customer changes the information.
4. Customer presses the "confirm" button.
5. System validates information and notifies the customer with a success message.

Customer is new on the website

1. Customer clicks on the "Register" button.
2. Customer provides required details in the registration form
3. Customer chooses their interests and preferences (products and brands).
4. System validates the provided information.
5. Customer account is created, and the customer receives the confirmation email.

Special Requirements:

1. Ease of Use
2. Responsiveness

Id: UC-02
Use Case: Search products from the website
Description:
This use case occurs when the customer wants to search items available on the shopping website.
Level: User Goal
Primary Actor: Customer
Supporting Actors: System
Stakeholders and Interests:
Customer: User wants to find available items.

Branch Employee: manages and maintains the available items.
Developer: Develops searching operation on branch employees scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.

Pre-Conditions:

1. The product's status must be correctly maintained by the website manager.
2. Customer must connect to internet

Post-Conditions:

1. Success end condition: If the product status is precisely reflected on the website, then the customer is successfully able to check availability of an item.
2. Failure end condition: If the product status is not precisely reflected on the website, then the customer is not able to check availability of an item.

Main Success Scenario:

1. Customer enters the item information in the search bar.
2. System searches the item in inventory and displays the results.
3. Customer views the details by clicking on items
4. Customer adds the item into the shopping cart.

Extensions
Customer searches an item using filter

1. Customer selects a category from the navigation bar.
2. The website displays search results with the available filters (category, price, and brand).
3. Customer applies the filter to narrow down the search result
4. Customer selects the item and adds it into the shopping cart.

Special Requirements:

1. Ease of Use
2. Responsiveness
3. Availability

Id: UC-03
Use Case: View recommended products
Description:
This use case occurs when the customer wants to view recommended items (most selling and least selling items, promotions, and used items) available on the shopping website.
Level: User Goal
Primary Actor: Customer
Supporting Actors: System
Stakeholders and Interests:
Customer: User wants to find items recommended based on their interests and preferences.
Branch Employee: manages and maintains the available items.
Developer: Develops display operation on branch employees' scenarios for the branch employees and performs unit testing.

Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Customer must have an account
2. The product's status must be correctly maintained by the website manager.
3. Browsing and purchase history or customer preferences are available in the website database.

Post-Conditions:

1. Success end condition: If the product status is precisely reflected on the website, and customer interests are available in the database, the customer is successfully able to view the recommended products.
2. Failure end condition: If the product status is not precisely reflected on the website or customer interests are not available in the database, the customer is unable to view the recommended products.

Main Success Scenario:

1. Customer logs into their account.
2. Customer goes to the website main page.
3. System displays the three recommended categories (most selling and least selling item, promotions, and used items).
4. Customer selects one of the categories and views a list of items.
5. Customer adds the items into the shopping cart.

Special Requirements:

1. Ease of Use
2. Responsiveness
3. Availability

Id: UC-04
Use Case: Fulfill and track orders
Description:
This use case occurs when the customer wants to complete and track the order.
Level: User Goal
Primary Actor: Customer
Supporting Actors: System
Stakeholders and Interests:
Customer: User wants to complete the order.
Branch Employee: manages and maintains the customer orders.
Developer: Develops order operation on branch employees' scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Customer must be logged into their account.
2. Customer must have items in the shopping cart
3. Customer must have valid payment information (credit card or gift card and address).

Post-Conditions:

1. Success end condition: If the payment information is valid, the order is completed successfully.
2. Failure end condition: If the payment information is invalid, the order cannot be completed.

Main Success Scenario:

1. Customer goes to the shopping cart and presses the “check out” button.
2. Customer enters the payment information and shipping address.
3. System transfer payment into payment gateway.
4. Payment gateway processes the transaction and provides a payment confirmation.
5. User receives the summation of the order.

Extensions:

Customer tracks their orders.

1. Customer logs into their account
2. Customer selects the order history and clicks the order they want to track
3. Website displays the order status.

Special Requirements:

1. Ease of Use
2. Responsiveness
3. Availability

Admin:
Id: UC-01
Use Case: Update website products
Description:
This use case occurs when the admin updates product information.
Level: User Goal
Primary Actor: Admin
Supporting Actors: System
Stakeholders and Interests:
Customer: Search product available on the website.
System Administrator: updates website products.
Branch Employee: manages and maintains the product information.
Developer: Develops update operation on branch employees’ scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Admin has an account and logged in.
2. Admin has accurate product information.

Post-Conditions:

1. Success end condition: Admin modifies the information successfully.
2. Failure end condition: Admin fails to modify the information.

Main Success Scenario:

1. Admin logs into the account and goes to the “update product” tab.
2. Admin adds new products or changes the information for the existing product.

3. Admin clicks the “save” button.
4. Website notifies the admin with a success message.

Special Requirements:

1. Responsive.
2. Ease of Use.
3. Accuracy.

Id: UC-02
Use Case: verify and complete customer orders
Description:
This use case occurs when the admin wants to verify and complete customer orders.
Level: User Goal
Primary Actor: Admin
Supporting Actors: System
Stakeholders and Interests:
Customer: Search product available on the website
System Administrator: verifies and completes customer orders
Branch Employee: manages and maintains the product information.
Developer: Develops complete orders operation on branch employees’ scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Admin has an account and logged in.
2. Customer placed an order successfully.

Post-Conditions:

1. Success end condition: Order is sent to the distribution center for shipping.
2. Failure end condition: Order is canceled if there is no stock or the payment information is invalid.

Main Success Scenario:

1. Admin logs into the account and goes to the “Orders” tab.
2. Admin clicks the “Ship now” button.
3. Website notifies the admin with a success message, and orders are moved to “Shipped” tab.
4. System sends an email confirmation to the customers.

Extension:

Admin modifies orders as requested by customers.

1. Admin clicks the “Modify” button
2. Admin edits the quantity.
3. Website notifies the admin with a success message.
4. System sends an email confirmation to the customers.

Special Requirements:

1. Responsive.
2. Ease of Use.
3. Accuracy.

Id: UC-03
Use Case: Manage stocks

Description:
This use case occurs when the admin manages the stocks.
Level: User Goal
Primary Actor: Admin
Supporting Actors: System
Stakeholders and Interests:
Customer: Search product available on the website
System Administrator: manages stock information
Branch Employee: manages and maintains the product information.
Developer: Develops manage stock operation on branch employees' scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Admin has an account and logged in.
2. Admin has accurate stock information.

Post-Conditions:

1. Success end condition: Stock information on the website is accurate.
2. Failure end condition: Stock information on the website is not accurate.

Main Success Scenario:

1. Admin logs into the account and goes to the "Stock" tab.
2. Admin edits the quantity.
3. Website notifies the admin with a success message.

Extensions:

Admin is notified when an item with 3 or fewer quantity

1. Admin goes to the "Stock" tab.
2. Admin views the item with a warning sign.
3. Admin updates the item quantity.
4. Website notifies the admin with a success message.

Special Requirements:

1. Responsive.
2. Ease of Use.
3. Accuracy.
4. Availability

Id: UC-04
Use Case: Generate sales data reports
Description:
This use case occurs when the admin generates the sales reports to identify most-selling and least-selling products within specific categories or across the entire inventory
Level: User Goal
Primary Actor: Admin
Supporting Actors: System
Stakeholders and Interests:
System Administrator: manages sales report
Branch Employee: manages and maintains the product information.

Developer: Develops report operation on branch employees' scenarios for the branch employees and performs unit testing.
Tester: Tests the scenario according to user requirements and suggests required changes to the developer.
Pre-Conditions:

1. Admin has an account and logged in.
2. There are completed orders in the database.

Post-Conditions:

1. Success end condition: Reports are generated successfully.
2. Failure end condition: Reports are not generated.

Main Success Scenario:

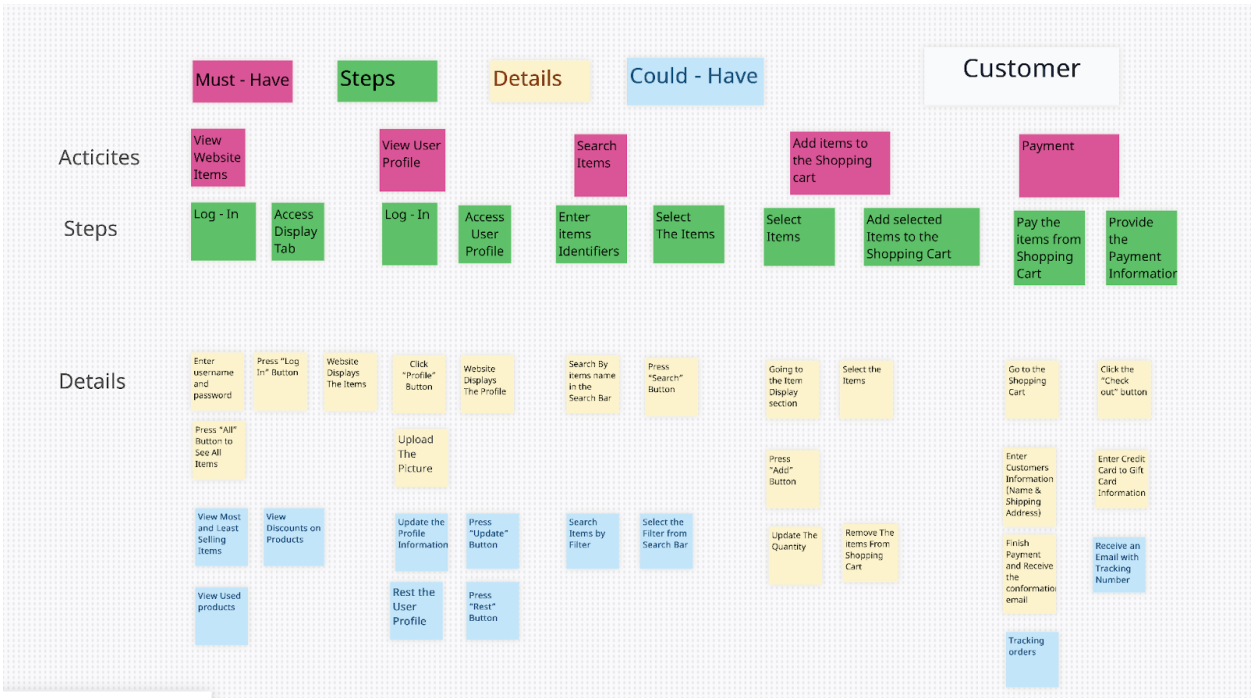
1. Admin logs into the account and goes to the "Report" tab.
2. Admin selects reporting period and category.
3. Website generates a report with most selling and least selling products highlighted.

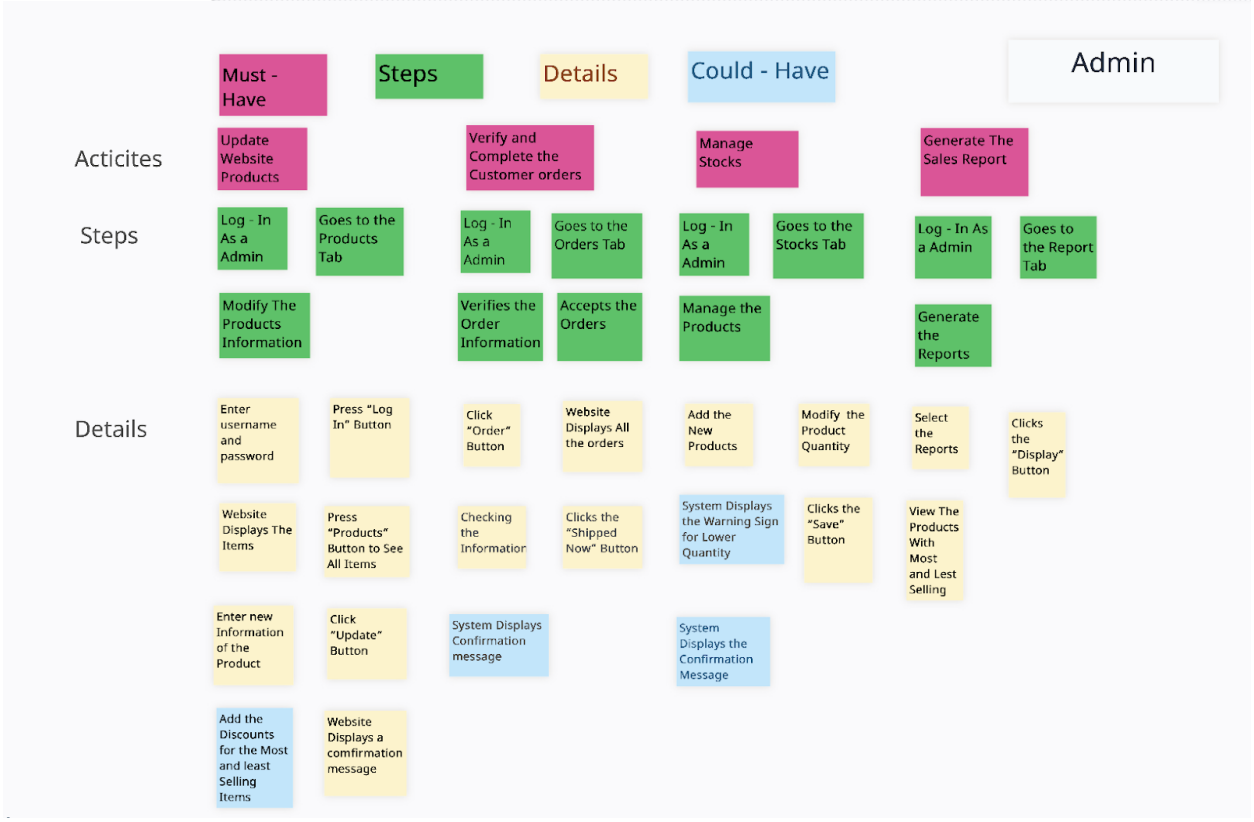
Extensions:

Admin is notified an item with most and least selling

1. Admin checks the sales report
2. Admin views the items with most and least selling
3. Admin adds the discounts for these items
4. Website notifies the admin with a success message.

8. User-Stories





9. UML Diagrams

1. Use Case Diagram

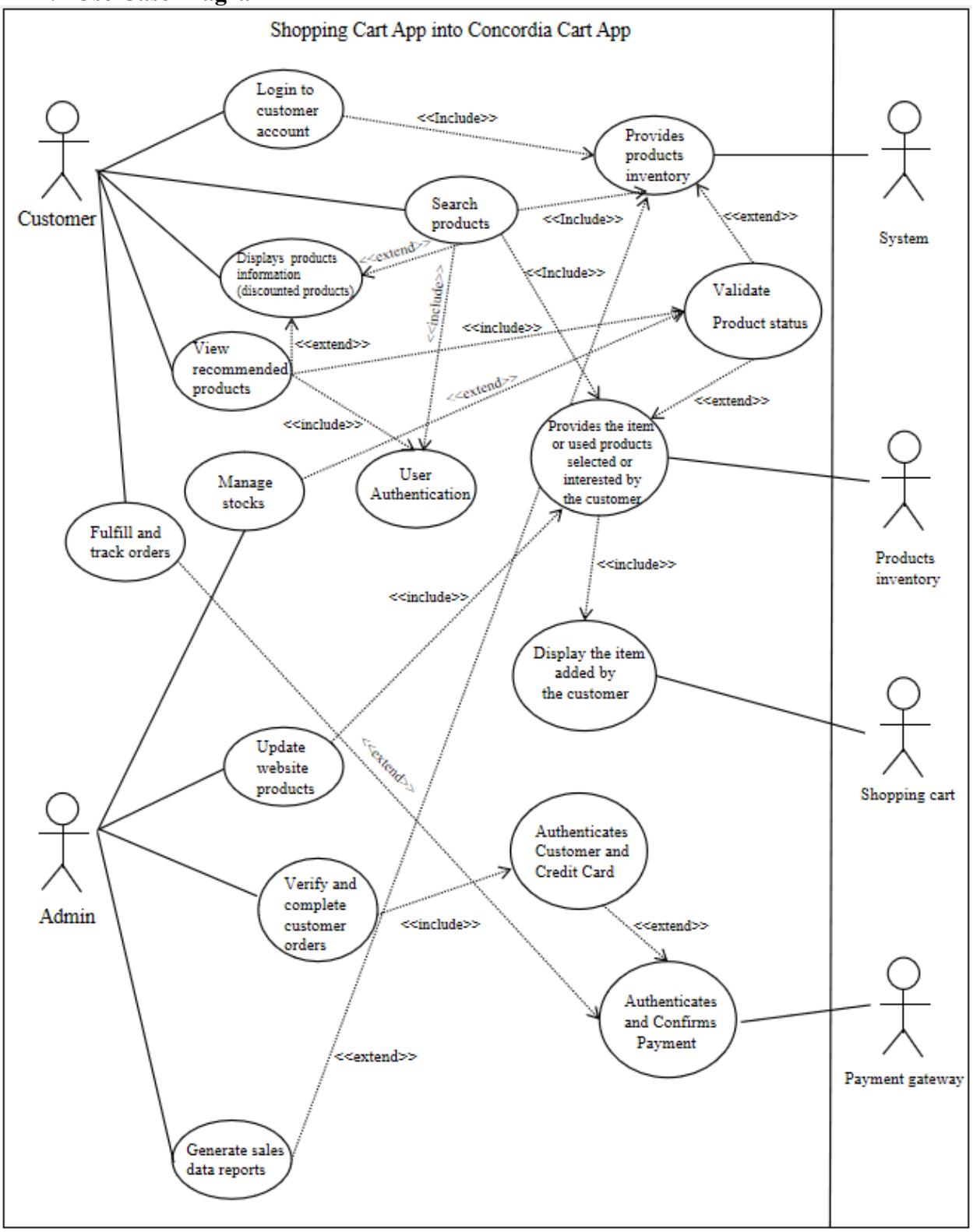


Figure 1. Shopping Cart App System Use Case Diagram

2. Sequence Diagram

2.1 Making order

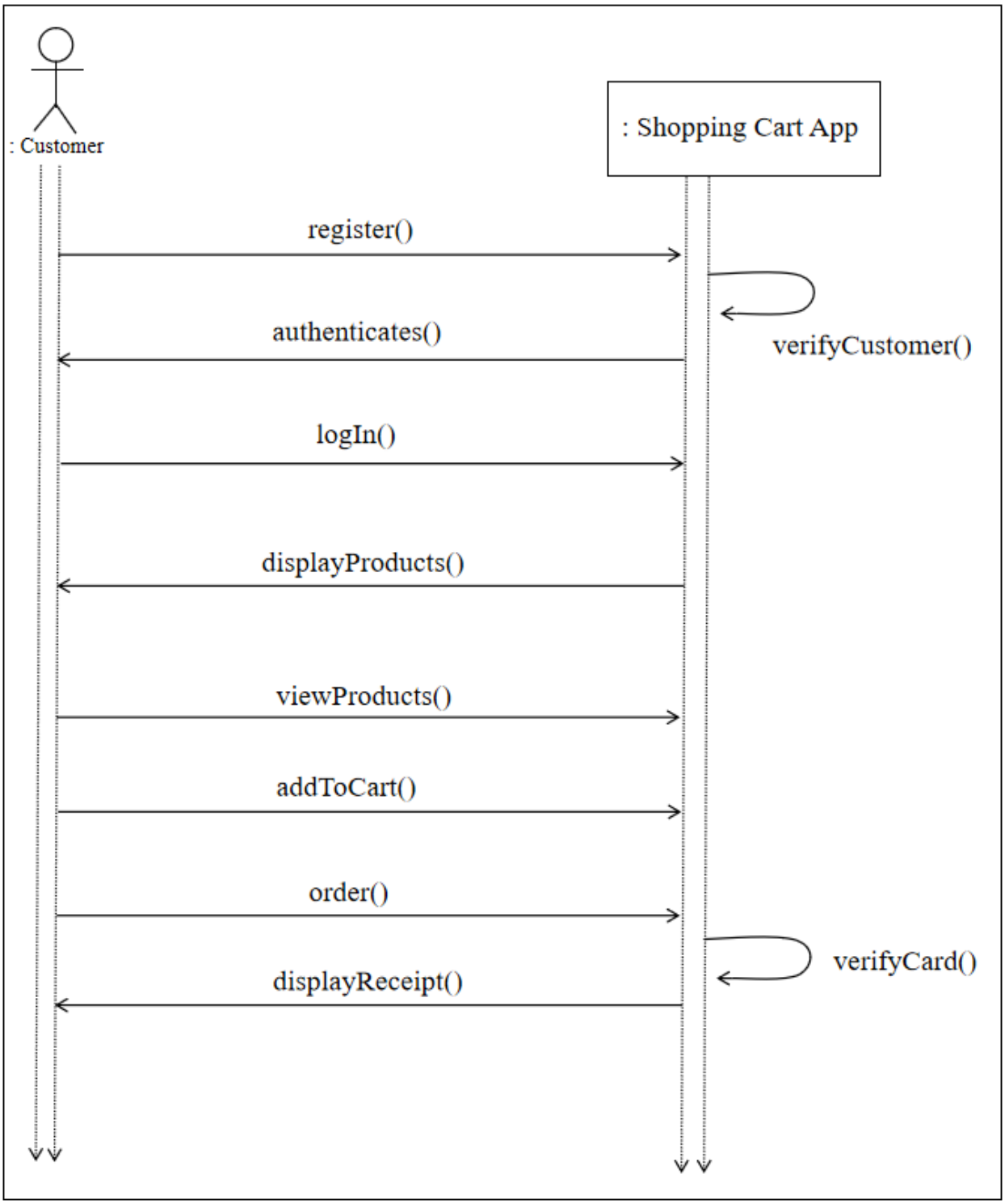


Figure 2. Sequence Diagram of Making Order

2.2. Check Shopping Cart

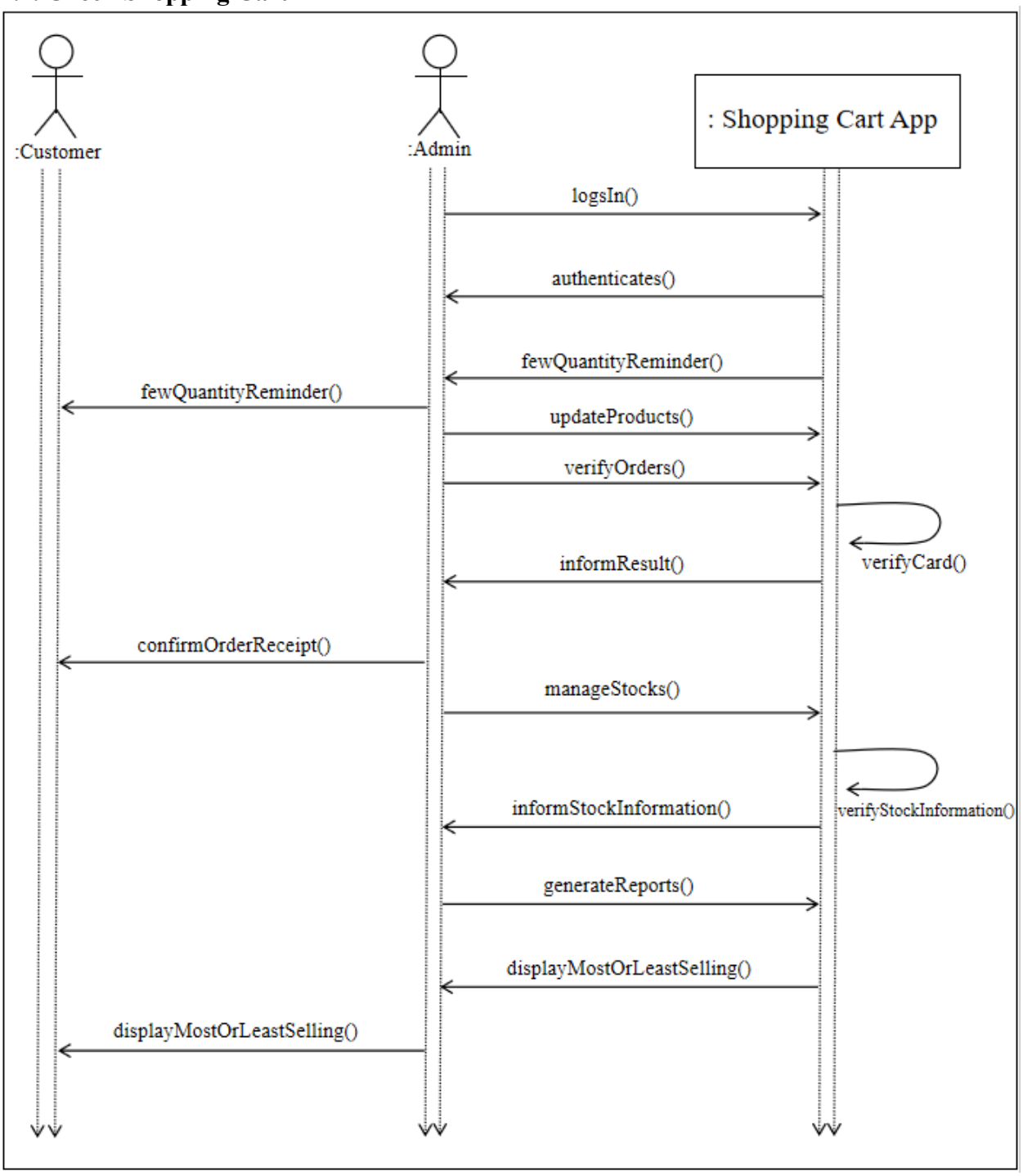


Figure 3. Sequence Diagram of Checking Shopping Cart

3. Activity-diagram

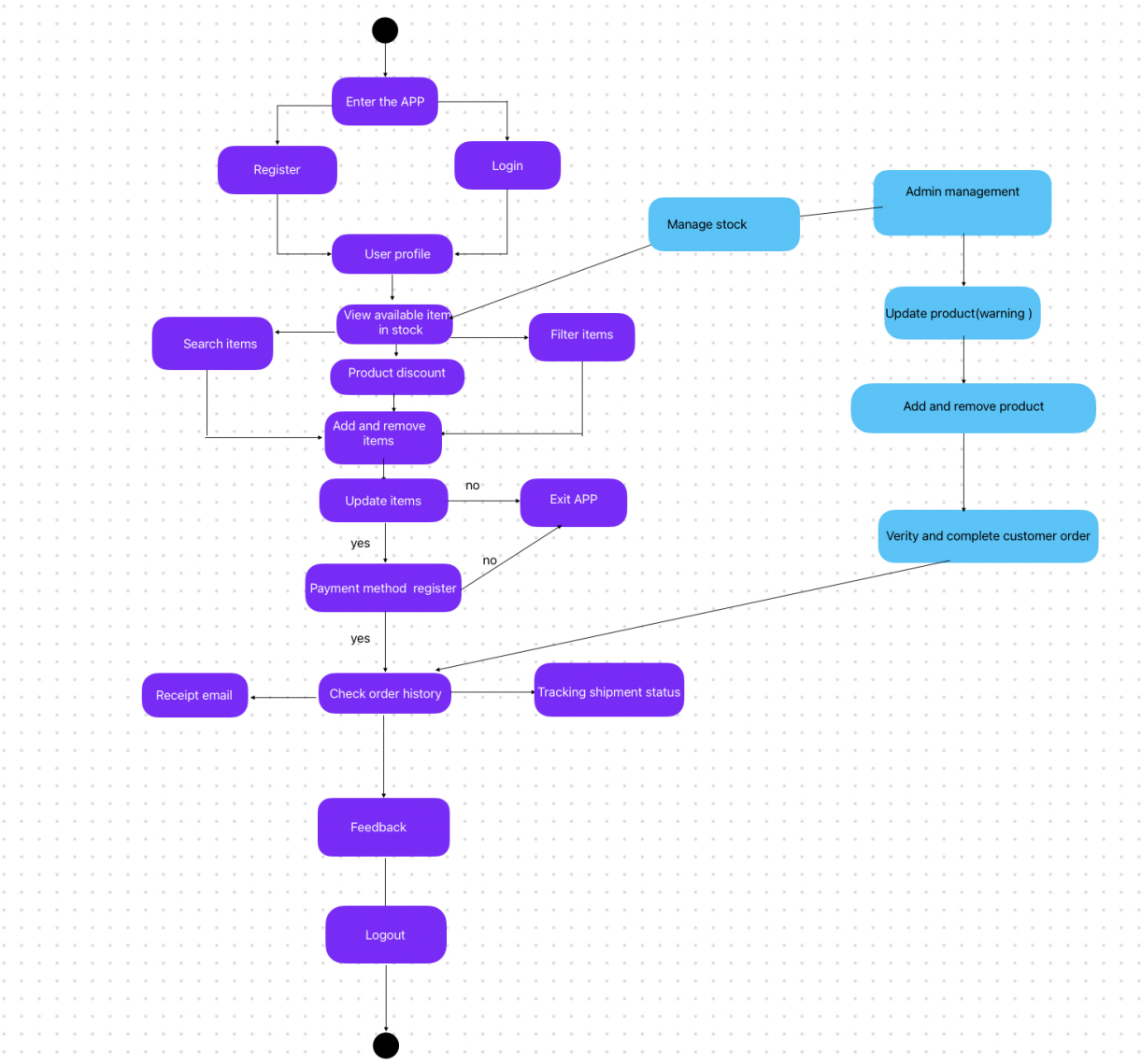
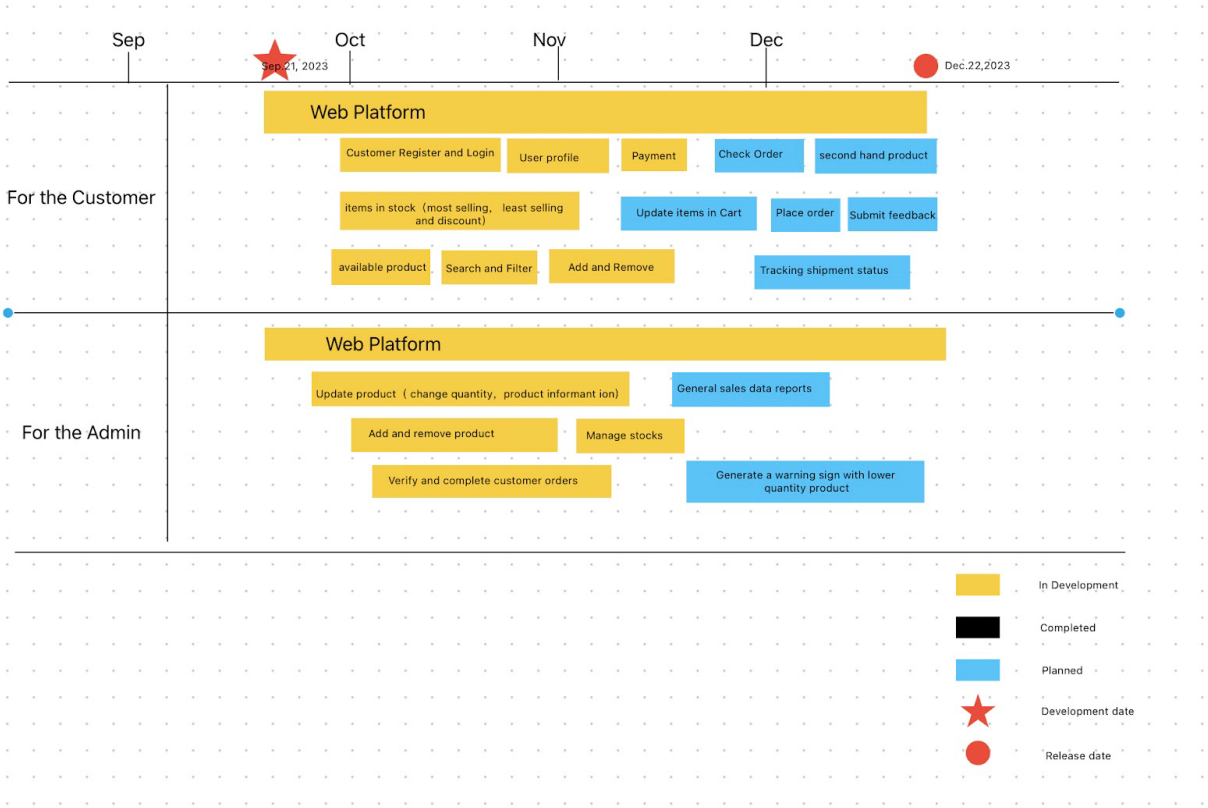


Figure 4. Shopping Cart App Activity Diagram

10. Roadmap



11. Software Design Strategies

In software development, software design strategies refer to the methods and principles chosen to achieve the requirements and goals. Design strategies ensure that the software has the abilities of reliability, scalability, and efficiency. In the development process of “Concordia Cart” application, design strategies provide a pivotal role which presents the high cohesion (where the elements with components are related to each other closely) and low coupling (lowest dependencies between components) between the software components.
Software design strategies are important to protect the software system’s performance. A clear design strategy is able to avoid the inconsistencies during the development process. Our main design strategies are the relation of components, a component level design with required internal interfaces, and a simple and user-friendly interface, during the design of “Concordia Cart” application.

Software Design Strategies:

Our software design strategies will follow these principles:

High Cohesion and Low Coupling:
We will guarantee that the component has high cohesion, we need to design the component has a distinct responsibility and function of each component should not contain unnecessary tasks. In our design strategies, each component is designed to be focused on completing a specific task

without any unrelated functions. This clear definition is helpful to let team members understand the role of components. When we modify or extend the system, it cannot affect the other components. Moreover, we will minimize the coupling between components. It means that we can increase our software flexibility by reducing the interdependencies between each component. The lower coupling means the modification of any components will not cause change in other components. As a result, each component can be developed and tested easily with any influence.

Learnability and Usability:
We will focus on the user experience in the user interface design. This means using a user-friendly layout with clear labels and a simple workflow to enable new customers to get started quickly. For example, the common actions and features will be prominently displayed, the navigation should be clear and concise, and the consistent color typography should be used to avoid visual fatigue.

Modular Design:
We are also considering the use of a modular design approach. This approach proves helpful when we divide the system into independent modules, allowing each module to focus on specific functions. We can enhance the flexibility of the system.

Iterative Development:
The iterative development method is used during our software development. we are able to modify the system based on the feedback. It can make sure that we can satisfy the customers' requirements and quickly solve the issues during the development time.

By following the design strategies above, we are able to develop and create a stable, user-friendly, and easy maintain Concordia Cart application.

12.Architectural Level Design

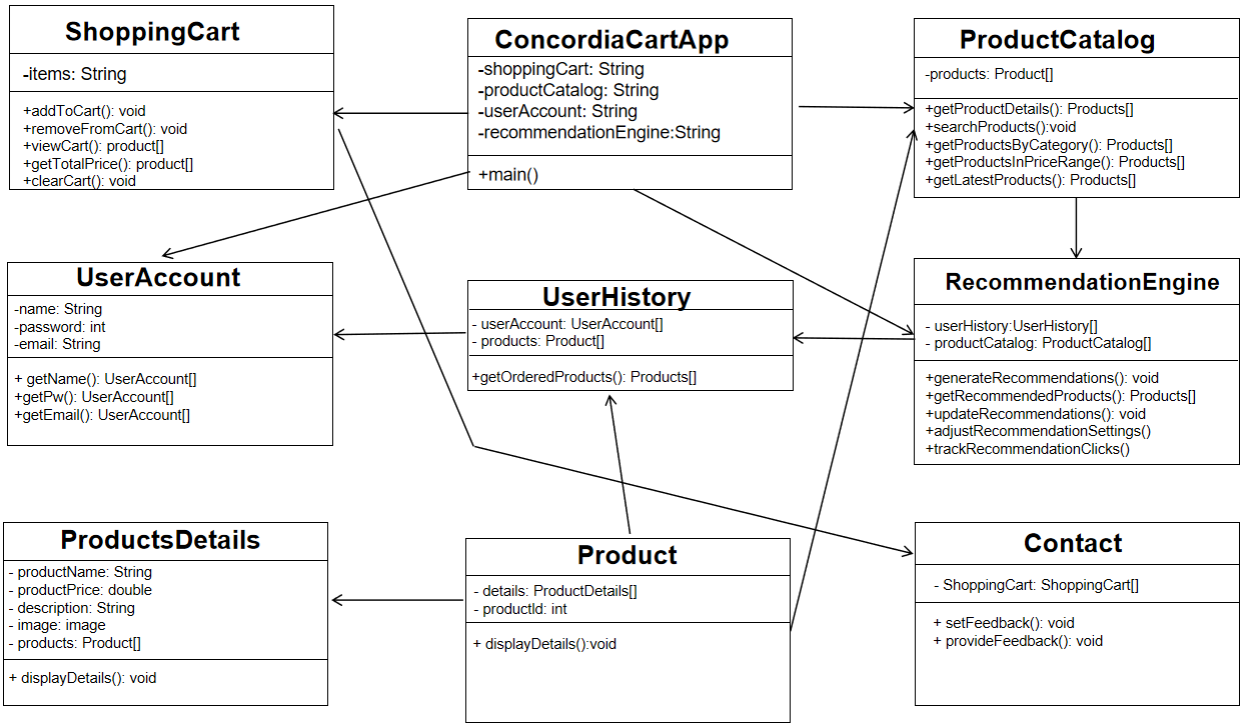
To create an architectural design for the Concordia Cart App, we need to define the primary components and relationships in the system. Here is a possible class diagram for the application:

Overview of the diagram classes:

ConcordiaCartApp	This is the main class that manages the entire application.
ShoppingCart	This class represents the shopping cart of a user.

ProductCatalog	This class represents the catalog of products available on the website.
RecommendationEngine	This class is responsible for providing personalized product recommendations to users.
UserAccount	This class represents a user's account.
UserHistory	This class represents the history of products that a user has viewed or purchased.
ProductDetails	This class represents the details of a product, including its name, price, description, and image.
Product	This class represents a product. It has a product ID and product details.
Contact	This class represents feedback or review products, influencing future buyers and assisting in refining the recommendation engine.

Clickable	Click() method enables clicking operation
Selectable	Select() method enables selection



13.Component Level Design

ConcordiaCartApp:

High Cohesion:

- This class efficiently manages the entire application by orchestrating different functionalities and components. The function of main() in the program is mainly to run the program.
- Ensure that it maintains a unified and coherent purpose for managing various aspects of the application.

Low Coupling:

- ConcordiaCartApp serves as an application toolbar, which is required to enable users to navigate different functions of the website, such as ShoppingCart, ProductCatalog, UserAccount, and RecommendationEngine.
- Allowing the ConcordiaCartApp to oversee various functionalities independently.

Definition:

- Serves as the central class that governs the entire application.
- Contains instances of ShoppingCart, ProductCatalog, UserAccount, and RecommendationEngine to oversee different aspects of the application.

Responsibilities:

- Manage the coordination and interaction of different application functionalities.
- Control the flow of data and operations between the components.
- Ensure smooth communication and cooperation among ShoppingCart, ProductCatalog, UserAccount, and RecommendationEngine.

Composition:

- Main():It is the core of the program.Does not contain submodules but interacts with instances of ShoppingCart, ProductCatalog, UserAccount, and RecommendationEngine.

Interaction:

- It has a toolbar which consists of items and their subitems. These are the, “ShoppingCart, ProductCatalog”, “UserAccount” and “RecommendationsEngine” tabs .

Interface:

- Define clear interfaces and methods for user interactions and application functionalities.
- Specify input parameters and expected output for each interface method, ensuring a cohesive user experience.

ShoppingCart:

High Cohesion:

- The methods addToCart(), removeFromCart(), viewCart(), getTotalPrice(), reviewCart(), and clearCart() are grouped together because they perform related functions in the program, including adding/removing, viewing the shopping cart, calculating the total price, and removing selected products. Grouping these functions together makes it highly cohesive.
- Concentrate on managing the user's shopping cart, focusing on cart-related functionalities.Ensure that the class is dedicated to handling the user's intended product purchases and cart management.

Low Coupling:

- Minimize dependencies with other classes, enabling the ShoppingCart to operate independently and reducing inter-class dependencies.

Definition:

- Represents the shopping cart of a user, containing products the user intends to purchase.

Responsibilities:

- Manage the addition, removal, and viewing of products in the user's shopping cart.
- Maintain the list of products in the shopping cart.
- Provide functionality to calculate the total price of the items in the cart.
- Support operations for reviewing and confirming the cart contents before the final purchase.

Composition:

- addToCart(product, quantity): Adds a specified quantity of a product to the cart.
- removeFromCart(product, quantity): Removes a specified quantity of a product from the cart.
- viewCart(): Displays the current contents of the shopping cart, categorized by product types.
- getTotalPrice(): Calculates the total price of all items in the cart.
- clearCart(): Empties the shopping cart, removing all items.

Interaction:

- A dropdown menu to select the group from a list of groups.
- Interact with UserAccount to associate the shopping cart with a user's account and store the cart contents.
- Collaborate with ProductCatalog to retrieve product details to display in the cart, including information about product categories.

Interface:

- Provide functionality to add/remove, preview the total price, and view all items in the shopping cart.

ProductCatalog:

High Cohesion:

- Concentrate on managing the product catalog and its related operations, ensuring a clear focus on catalog management.Ensure that the class is primarily responsible for handling the product catalog and its associated functionalities.
- The methods getProductDetails(), searchProducts(), getProductsByCategory(), getProductsInPriceRange(), and getLatestProducts() are grouped together because they perform related functions in the program, allowing users to search for products under different criteria, search for products by category, search within price ranges, and view the latest products. Grouping these functions together makes it highly cohesive.

Low Coupling:

- Minimize dependencies with other classes, enabling the ProductCatalog to function independently and reducing inter-class dependencies.

Definition:

- Represents the catalog of products available on the website, providing users with access to a wide range of products.
- Maintains a comprehensive list of available products and their details.

Responsibilities:

- Manage the product catalog, including product creation, modification, and retrieval of product details.
- Provide methods for users to search for products based on various criteria, such as name, category, or price range.
- Offer functionality for users to obtain detailed information about specific products within the catalog.
- Ensure the catalog remains up-to-date with the latest product information and availability.

Composition:

- getProductDetails(productId): Retrieves detailed information about a specific product.
- searchProducts(criteria): Allows users to search for products based on specified search criteria.
- getProductsByCategory(category): Retrieves a list of products within a given category(used products, most/least selling, discounted products).
- getProductsInPriceRange(minPrice, maxPrice): Retrieves products falling within a specified price range.
- getLatestProducts(count): Retrieves the most recent products added to the catalog.

Interaction:

- Collaborate with the ConcordiaCartApp to provide users with access to the product catalog.
- Interact with UserAccount to track and store user viewing history and purchase activity, facilitating personalized recommendations.

Interface:

- Provide different search methods to make it more convenient and efficient for users to find the right products.

Recommendation Engine:

High Cohesion:

- Concentrate on generating personalized product recommendations for users. Maintain a clear and cohesive focus on recommendation generation based on user history and the product catalog.
- The methods generateRecommendations(), getRecommendedProducts(), updateRecommendations(), adjustRecommendationSettings(), and trackRecommendationClicks() are grouped together because they collectively perform related functions in the program, which involve providing personalized product recommendations based on purchase history, search history, products within the shopping cart, user preferences, and feedback to recommend other relevant products. Grouping these functions together makes it highly cohesive.

Low Coupling:

- Avoid tight coupling with other components, allowing for flexibility and adaptability in the recommendation process.

Definition:

- Represents a class responsible for generating personalized product recommendations for users.
- Utilizes user history records and the product catalog to formulate tailored product suggestions.

Responsibilities:

- Implement recommendation algorithms that consider user preferences, viewing history, and purchase history to generate product recommendations.
- Regularly update and refine recommendation strategies to enhance the accuracy and relevance of suggestions.
- Collaborate with the ProductCatalog component to access the latest product information for recommendation generation.

Composition:

- generateRecommendations(userHistory, productCatalog): Generates a list of recommended products based on the user's history and the current product catalog.
- getRecommendedProducts(): Retrieves the list of recommended products to display to the user.
- updateRecommendations(): Periodically updates recommendations to adapt to user preferences and changing product availability.
- adjustRecommendationSettings(settings): Allows customization of recommendation algorithms and parameters to tailor recommendations to user preferences.
- trackRecommendationClicks(userFeedback): Gathers user feedback on recommended products to further refine the recommendation algorithms.

Interaction:

- Interact with the UserAccount to access user history records and preferences.
- Collaborate with the ProductCatalog to retrieve detailed information about recommended products, ensuring up-to-date recommendations.

Interface:

- Providing personalized product recommendations using related functionalities helps users discover products that are suitable for them and increases the sales of products.

User Account

High cohesion:

·The methods getName(), getEmail() and getPw() are grouped together, because these methods perform relevant functions in the program, and T=these functions are: the system obtains the user name and the user's email address. Grouping these methods together makes it highly cohesive.

Low coupling:

·UserAccount is linked to the ConcordiaCartApp, because the user can only open the program to perform actions, such as setting a username, password, and email address.

Definition:

·The purpose is to let the system require users to set username, password and add email address.

Responsibilities:

·Requiring users to register and login their account.

Composition:

·getName(name): The system will ask users to create an account and require username.

·getEmail(email): The system will ask users to create an account and add their email address.

·getPw(password): The system will ask users to create an account and create a password.

Interaction:

“Create account” button to allow users to create an account, and “login” button allow users to login their account.

Interface:

Provides methods to enable users to create accounts or login.

User History:

High cohesion:

● The User History class is highly cohesive as it is focused on managing the history of products viewed or purchased by a user. All its methods and attributes are centered around this responsibility.

Low coupling:

●User History should have low coupling with other components. It interacts with the User Account and Product classes but does not depend on the details of these classes.

Definition:

● User History is a class that keeps track of the products that a user has viewed or purchased. It has a list of Product objects.

Responsibilities:

●Maintain a list of products viewed or purchased by the user.

●Provide methods to add a product to the history.

●Provide methods to retrieve the list of products in the history.

Composition:

● A list of Product objects.

Interaction:

● User History interacts with the following components:

● User Account: User Account accesses User History to retrieve or update the user's history.

● Product: User History stores a list of Product objects that the user has viewed or purchased.

Interface:

● Provides methods that enable the user to add a product to the history and returns a list of products from the history.

Product Details:

High cohesion:

● The Product Details class is highly cohesive as it solely focuses on maintaining and providing details about a single product. All its attributes and methods are centered around describing and showcasing a product's specific attributes like its name, price, description, and image.

Low coupling:

●The Product Details class has low coupling as it operates independently of other classes for its core functionality. While other classes might reference Product Details to obtain information about a product, this class does not need to interact heavily with other components of the system, ensuring ease of maintenance and modifications.

Definition:

● The Product Details class provides an encapsulated representation of all the pertinent details related to a product available on the Concordia Cart App.

Responsibilities:

●Store essential details about a product: name, price, description, and image.

●Provide access to these details through its interface to other classes or components that need this information.

Composition:

●Attributes:

productId: a unique identifier for each product.

name: the name or title of the product.

price: the monetary value of the product.

description: a textual description of the product, providing users with more information.

image: a graphical representation or image of the product.

Methods:

viewDetails(): allows external entities to retrieve the product's details.

Interaction:

● While the Product Details class itself is designed to be independent, it can be referenced or used by classes like Product, Shopping Cart, or the Recommendation Engine when there's a need to showcase or make decisions based on products' details.

Interface:

● The primary interface provided by the Product Details class is the viewDetails() method. This method can be invoked by other classes or components to obtain a product's details.

Product:

High Cohesion:

●The "Product" class is highly cohesive as it encapsulates all the attributes and behaviors that are specific to a product. This includes the product ID and product details.

Low coupling:

●The "Product" class is loosely coupled with other classes. It interacts with other classes through well-defined interfaces, but the internal implementation details of the "Product" class are independent of other classes. This makes it easier to maintain and update the "Product" class without affecting other parts of the system.

Definition:

●The "Product" class represents a product in the e-commerce platform. It contains the essential information and behaviors related to a product.

Responsibilities:

●Store product ID and product details.

●Provide access to product details.

Composition:

●productId: int

●details: ProductDetails

Interaction:

●The "Product" class interacts with the "ProductDetails" class to get the detailed information of the product. It can also interact with other classes such as "ShoppingCart" and "UserHistory" to support adding the product to a shopping cart or recording the product in a user's history.

Interface:

●The "Product" class provides a simple interface for other classes to interact with it. The primary method provided by the "Product" class is:

viewDetails(): ProductDetails

Contact:

High cohesion:

● The Contacts class is highly cohesive as it solely focuses on handling feedbacks and reviews about products. Every method and attribute in this class is directly related to managing, storing, or retrieving feedback and reviews.

Low coupling:

●The Contacts class should ideally have minimal dependencies on other classes, ensuring that changes in other classes won't severely affect Contacts. It might have a link to the Product class to associate feedback with a specific product and perhaps to UserAccount to identify who provided the feedback.

Definition:

● Contacts class represents product feedback and reviews that influence future buyers and assists in refining the recommendation engine.

Responsibilities:

●Store feedback and reviews.

●Link feedback to specific products.

●Link feedback to specific users.

●Provide metrics or summaries of feedback for use by other components, like the recommendation engine.

Composition:

● Concordia Cart App is composed of ShoppingCart, ProductCatalog, UserAccount, and RecommendationEngine.

● UserAccount is composed of UserHistory and ShoppingCart.

● ProductCatalog is composed of Product and ProductDetails.

● RecommendationEngine is composed of UserHistory and ProductCatalog.

Interaction:

● Concordia Cart App interacts with all other components.

● ShoppingCart interacts with ProductCatalog to add or remove products.

● RecommendationEngine interacts with UserHistory to generate recommendations.

● UserAccount interacts with ShoppingCart and UserHistory to manage the user's account.

● Contacts interacts with RecommendationEngine to refine the recommendations based on user feedback.

Interface:

● Concordia Cart App exposes interfaces for searching products, viewing product details, adding products to the cart, and checking out.

●ShoppingCart exposes interfaces for adding and removing items, clearing the cart, and getting the total price.

●ProductCatalog exposes interfaces for searching products and getting product details.

●RecommendationEngine exposes an interface for recommending products.

●UserAccount exposes interfaces for viewing history and updating history.

●UserHistory exposes interfaces for adding products and getting the list of products.

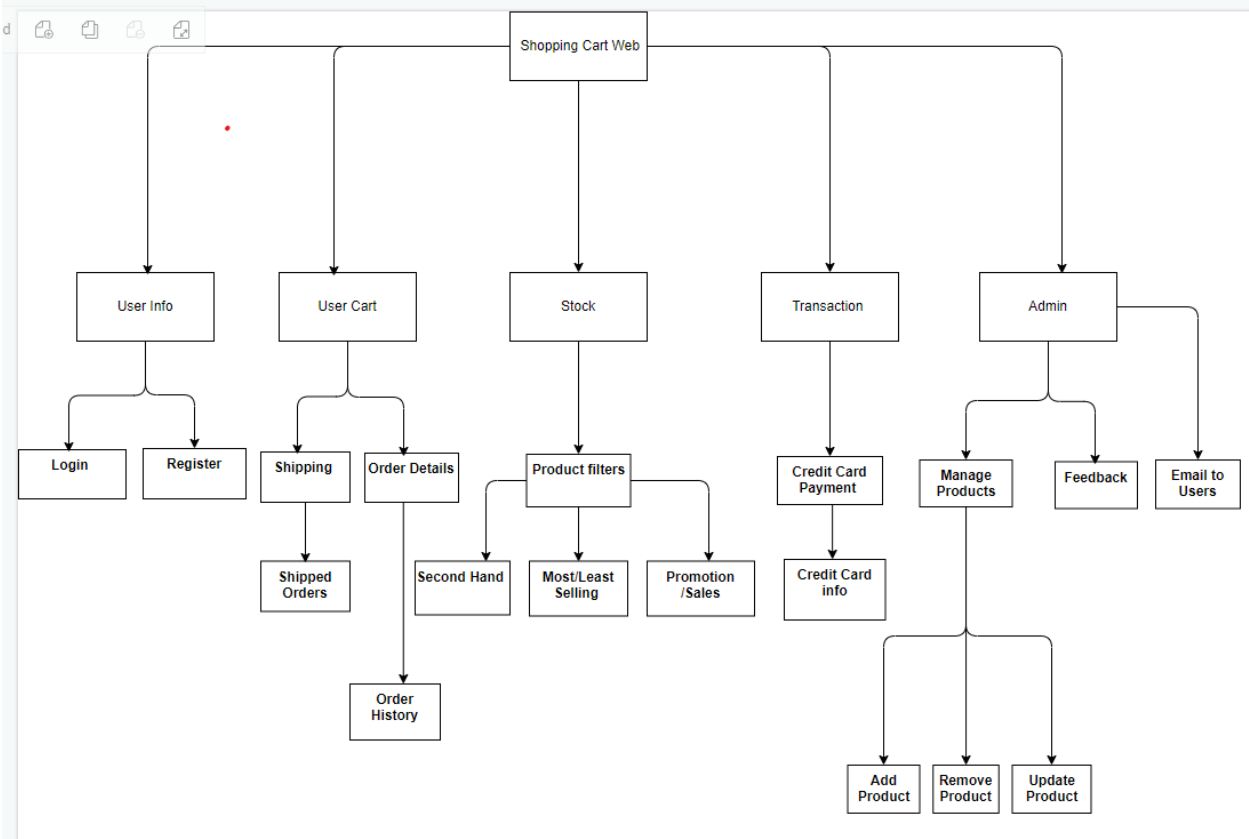
●ProductDetails exposes an interface for viewing product details.

●Contacts exposes interfaces for collecting feedback and reviewing products.

14. Interface Level Design

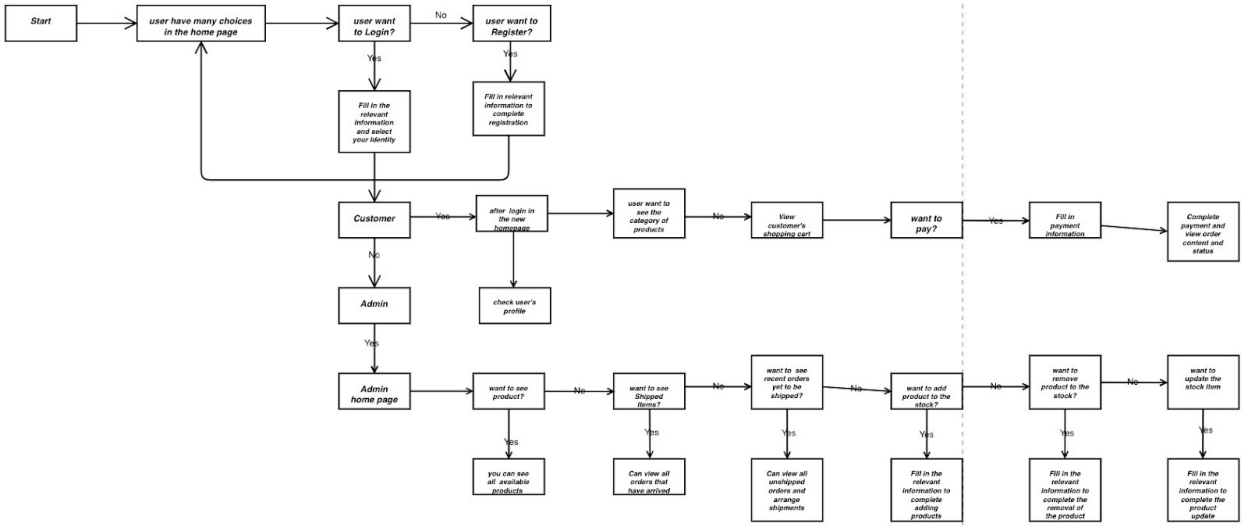
1. Information Desig :

The diagram shows the visual presentation of features and infrastructures that a shopping cart website contains.



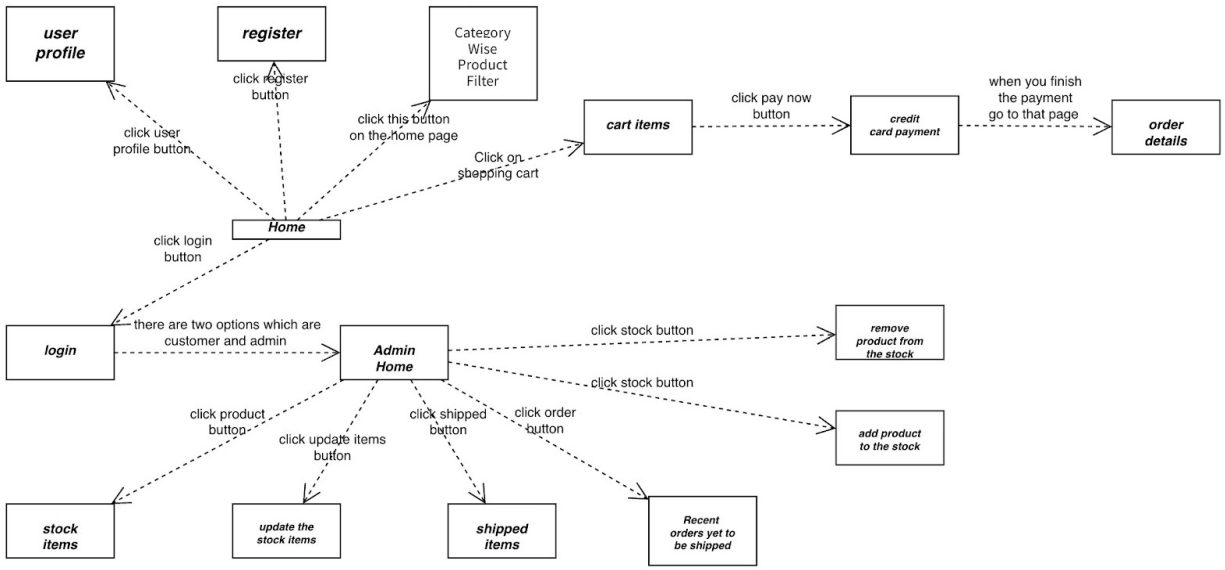
Interaction Design

2. This section details how users operate on each page, what options are available, and the next interface display after selecting.



Visual Design

3. This section introduces the use process of this website and the content contained in each part as well as extended content.

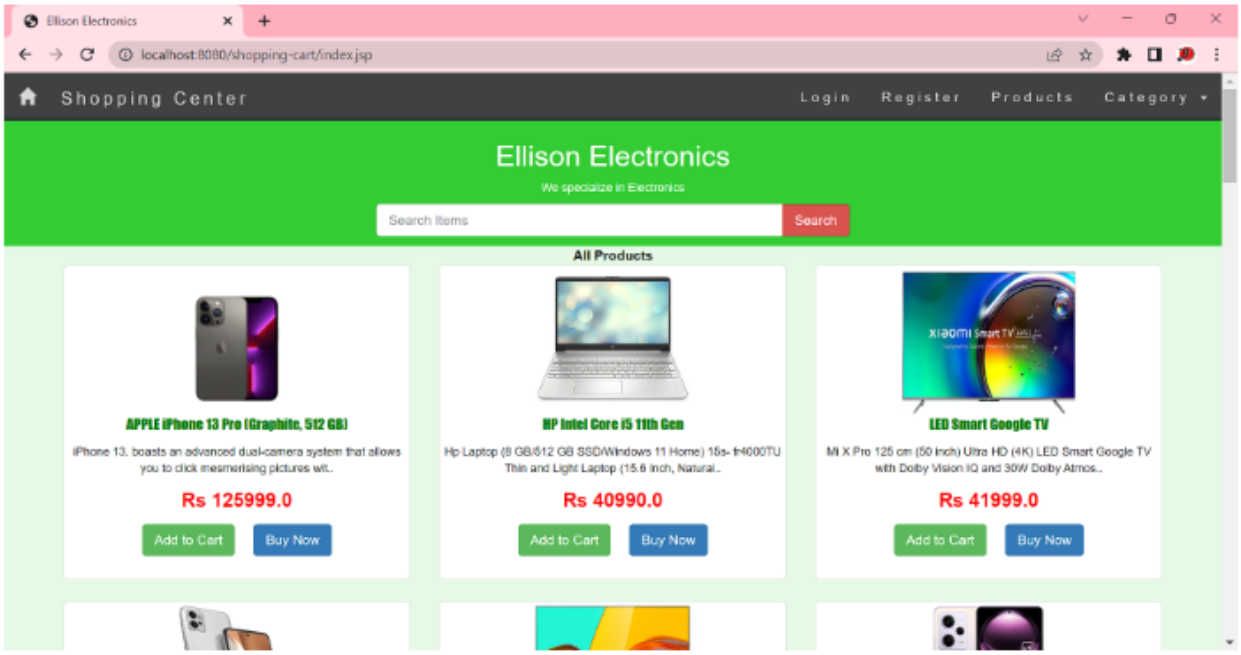


User Experience Design

4. Based on a previous visual design diagram, the shopping cart website allows both users and admin to access all the functions that the website has. Users can login or register their

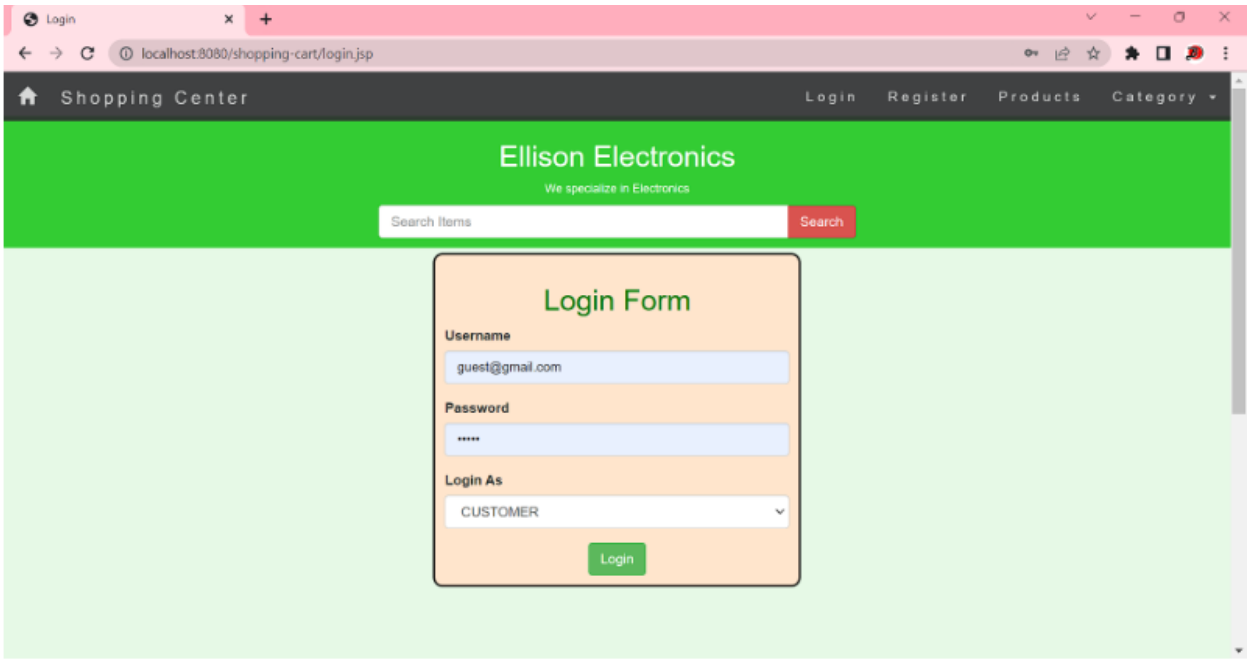
accounts to allow them to use shopping cart website service. Admin will also use the website to manage the stock products in this website. In the UI of the shopping cart website, users and admin will select responsive button to implement the functions of this shopping cart website.

The Home page includes general functions of this shopping cart website with respect to the top corner of the website.



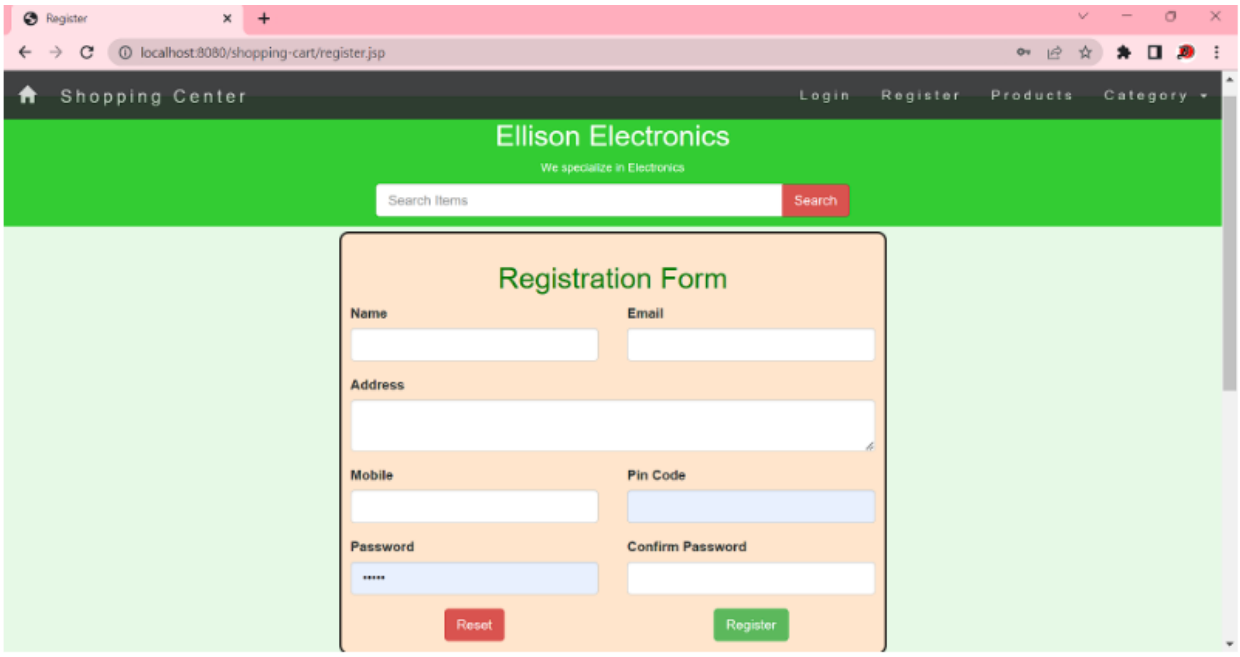
1. Existing users could login to their account to use website service by clicking Login button, it will direct to a login page to help users to login, or they could register a new account if they are new to this website:

Login for existing users:
Users will only need to enter their Username and password to login to their account

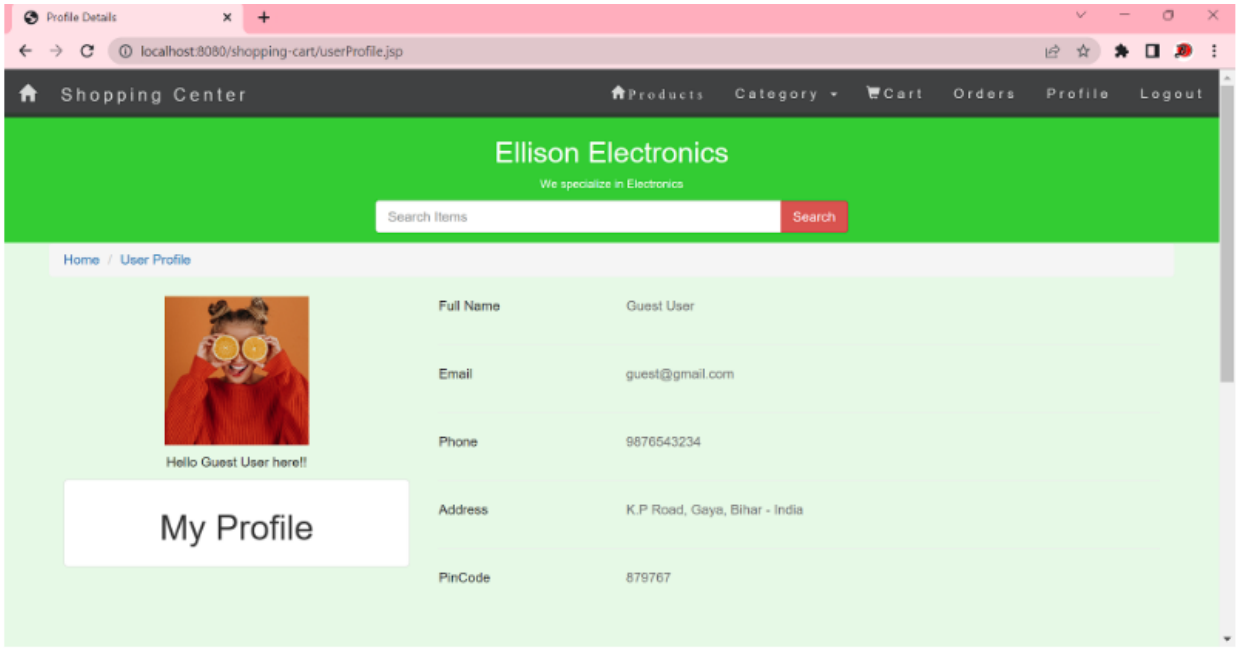


Register for new users:

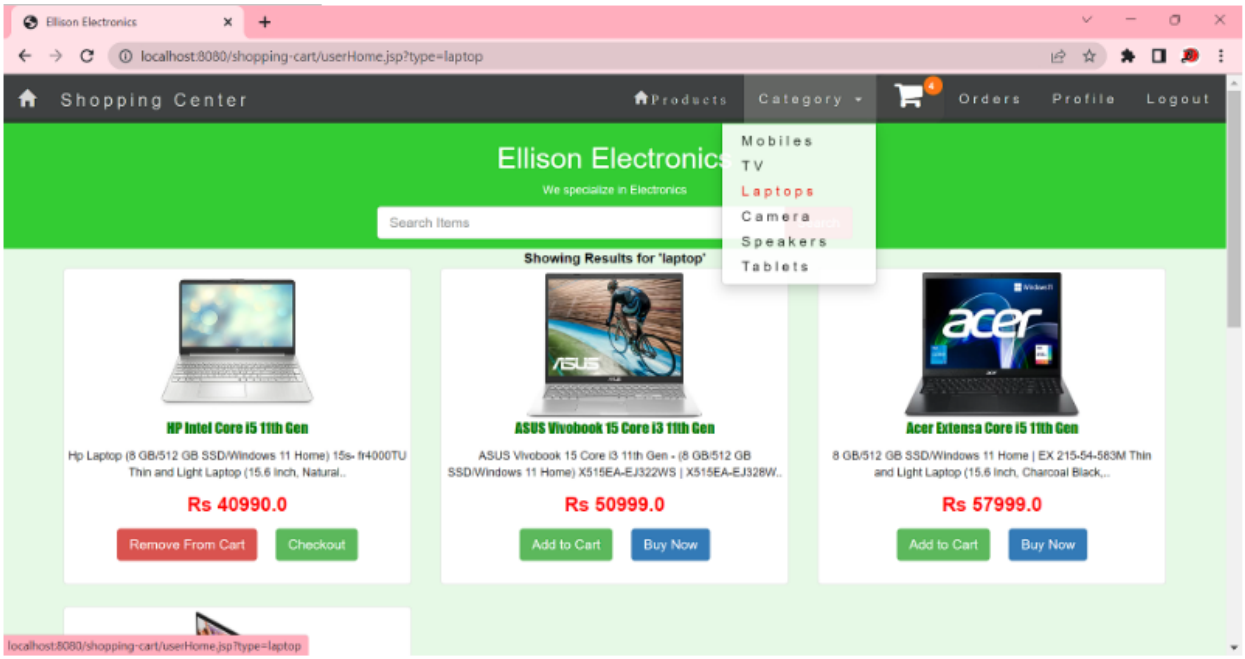
New users filling out the form to provide necessary information to create their own account



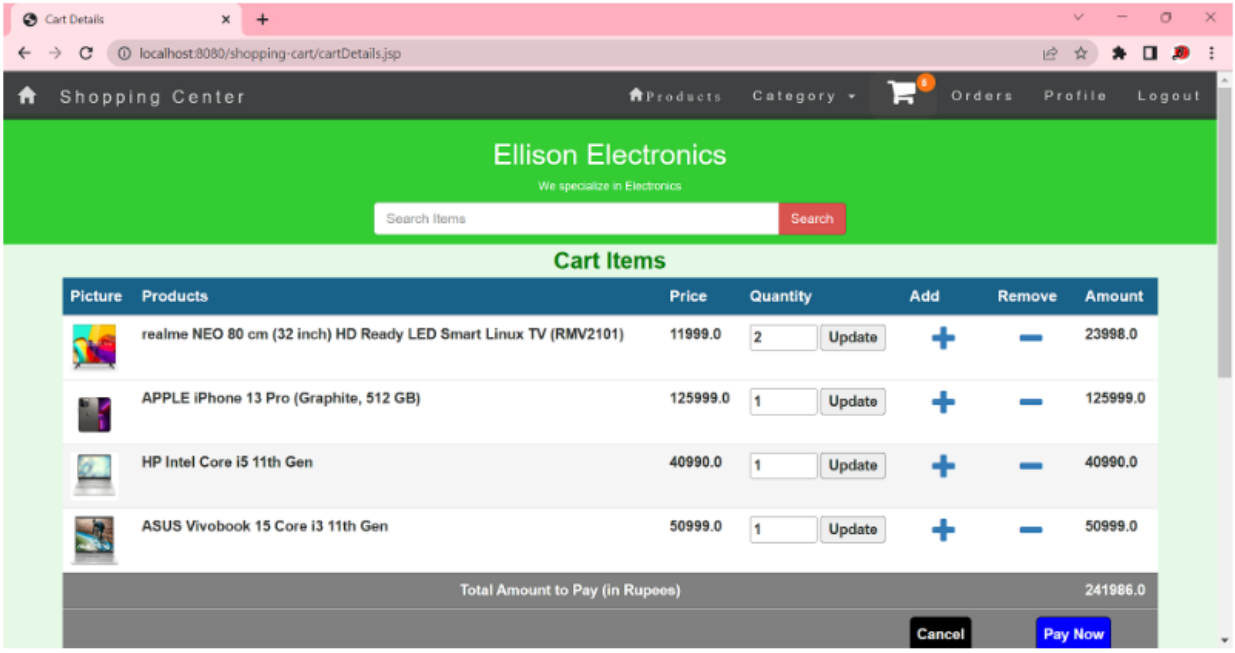
Checking user info:
After creating an account or login to the account, users or admin may check their personal information by accessing the profile at the top right of website.



Clicking the “Category” button, users can filter the products in the shopping cart website and choose to add to the cart to buy products .The most selling and least selling products, second hand products , and promotions will also be appearing on this website. It will be completing further in the project.

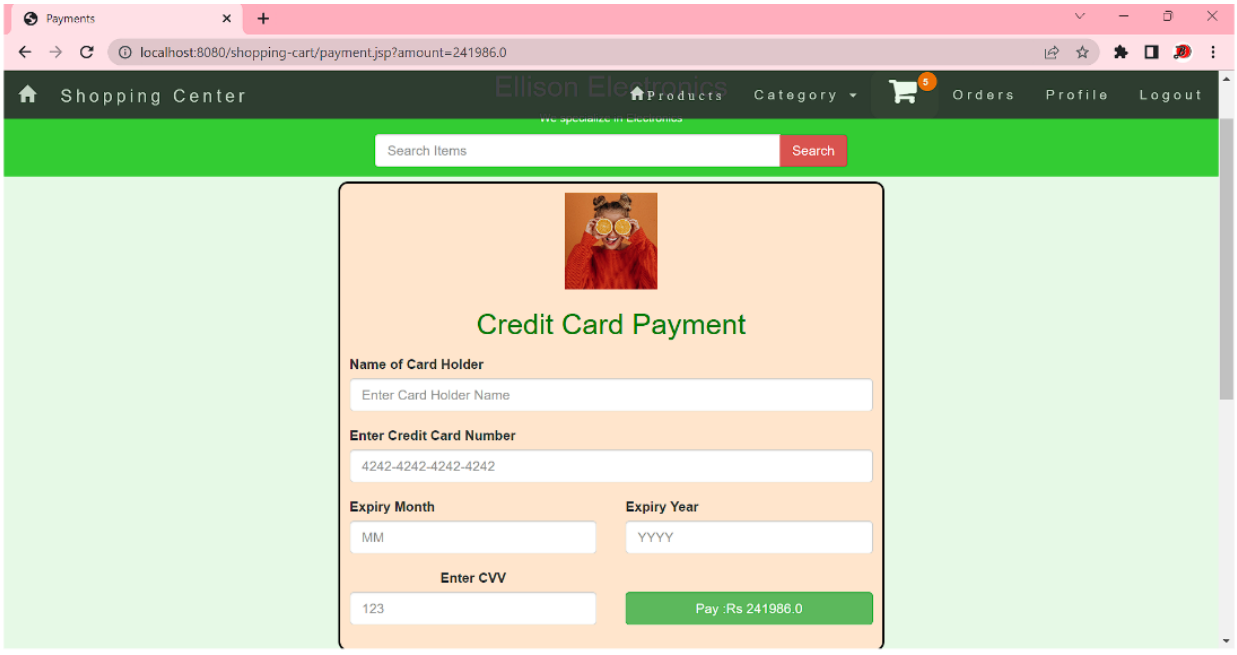


The purchased product will be displayed in the shopping cart symbol button which appears at the top right column of the corner. It shows all the products that users wish to purchase and the total price for each type of product that users choose.



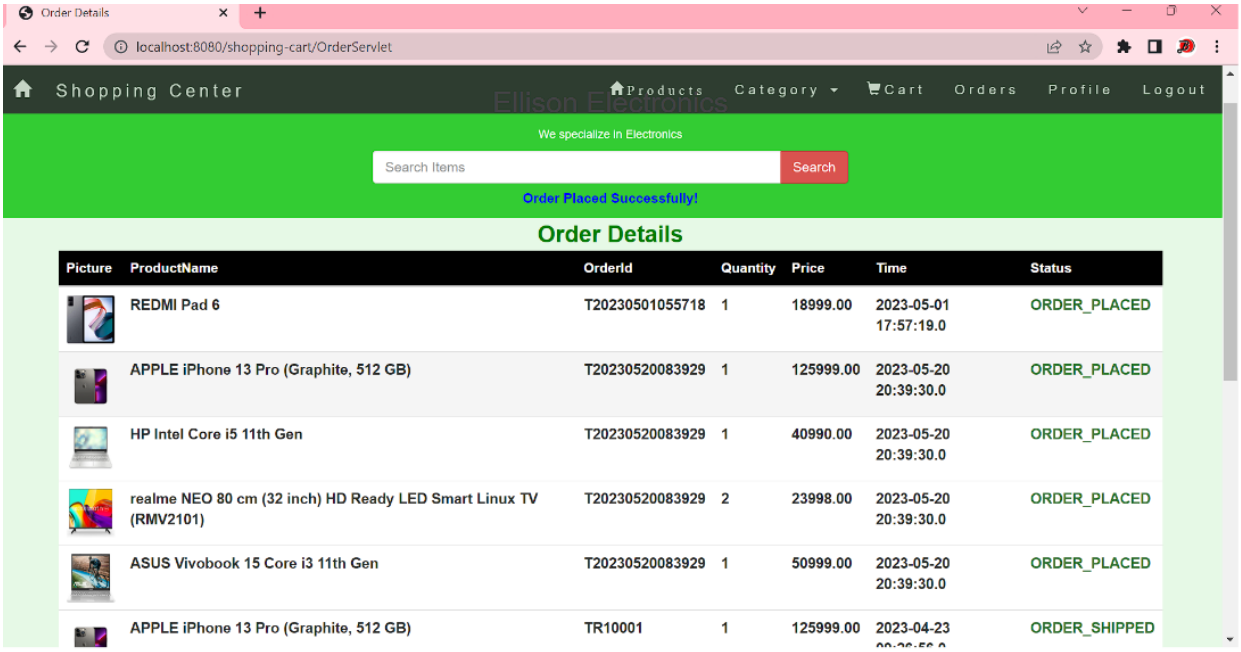
2. Credit card payment

Users could click the “Pay now “button to proceed to the credit card payment page. Users have to fill out all required information to complete their orders.



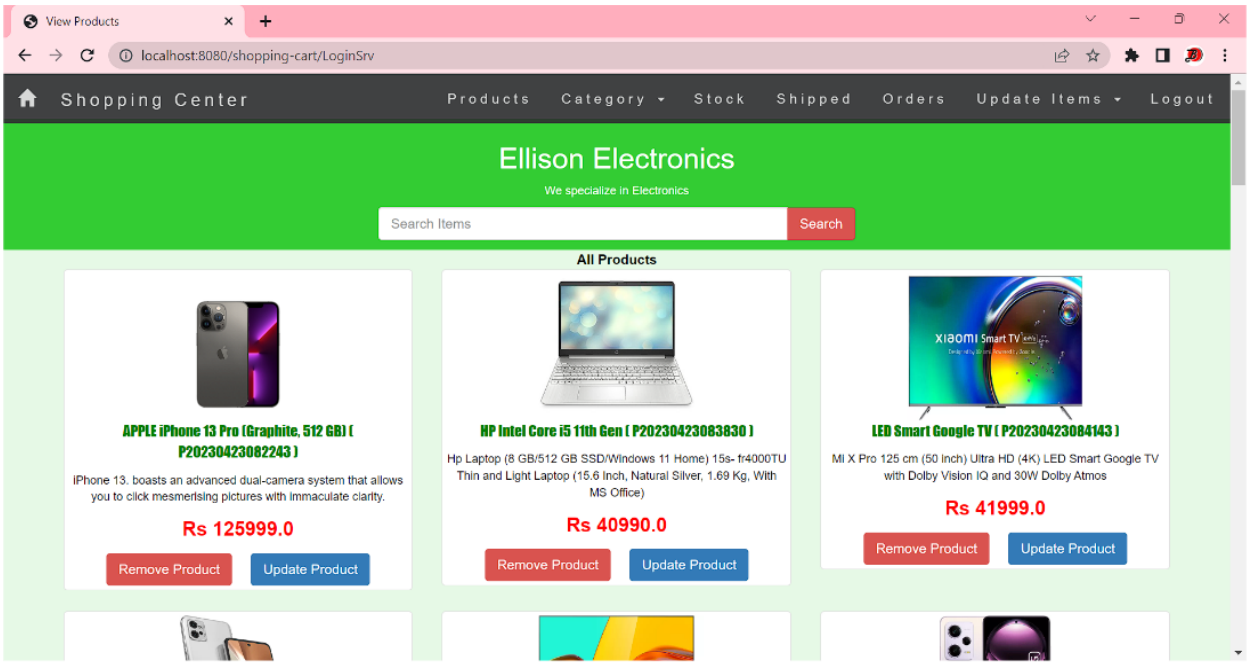
3. Order Details after placing orders

Users could check which orders are placed after users confirm “Pay now” button



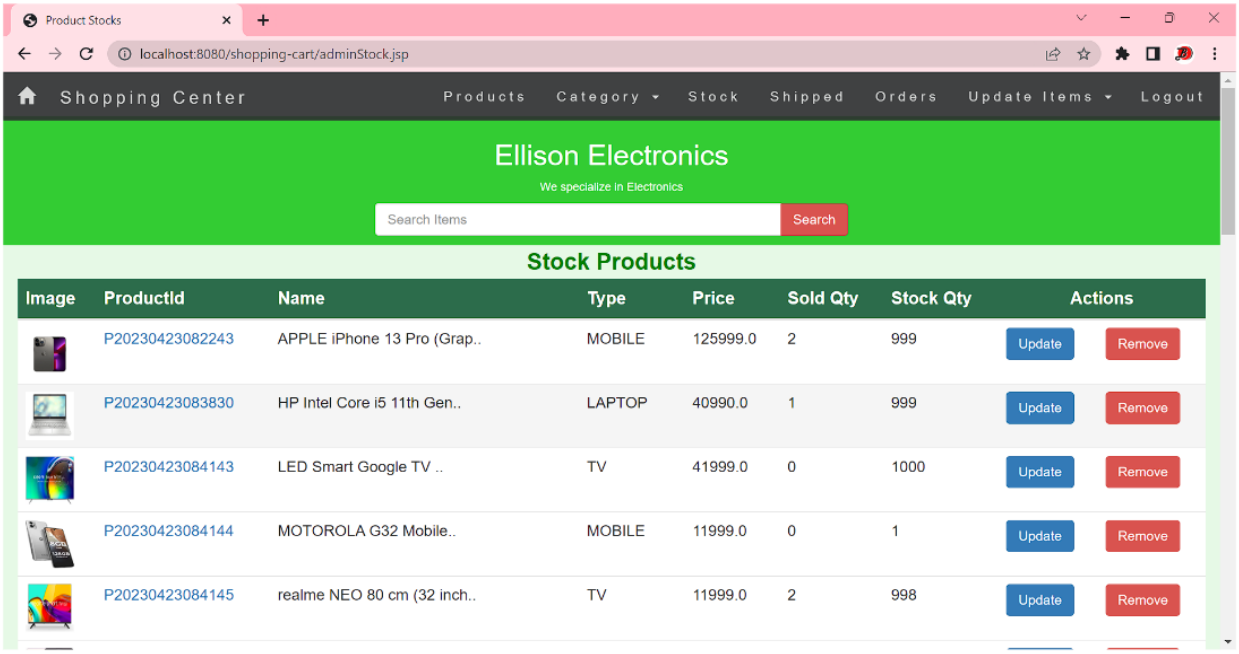
4. Admin management

Admin login with his/her admin home page , it has several options such as Products, Category, Stock , Shipped buttons etc. We will add a “Report” button at the same column of the webpage that helps admin identify most and least selling products ,and it helps admin to decide which products will be on promotion based on the sales report .

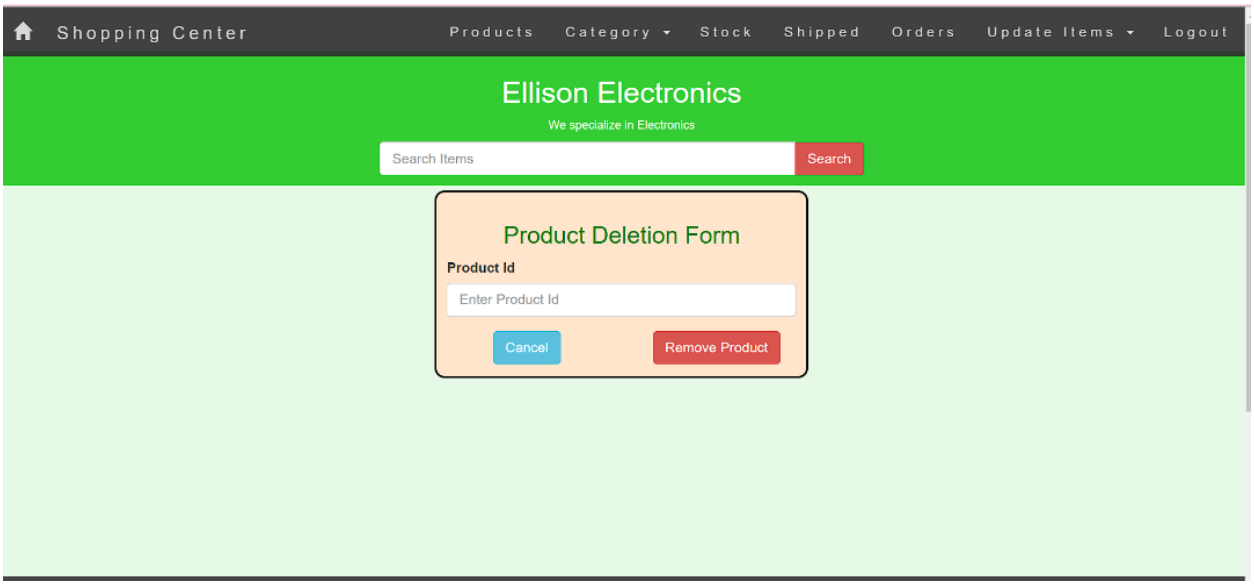


Clicking" Product" button, it will direct admin to overall products in this shopping cart websites Admin could manages the product of shopping cart website so he/ she can remove or update the product by clicking "Remove product" or" Update product" button (admin could also see a alert message at the middle of product image that warns admin that the stack of this product is lower than 3) If admin want to search for s specific item in shopping cart website ,they can click "Category "button to filters products and find it.

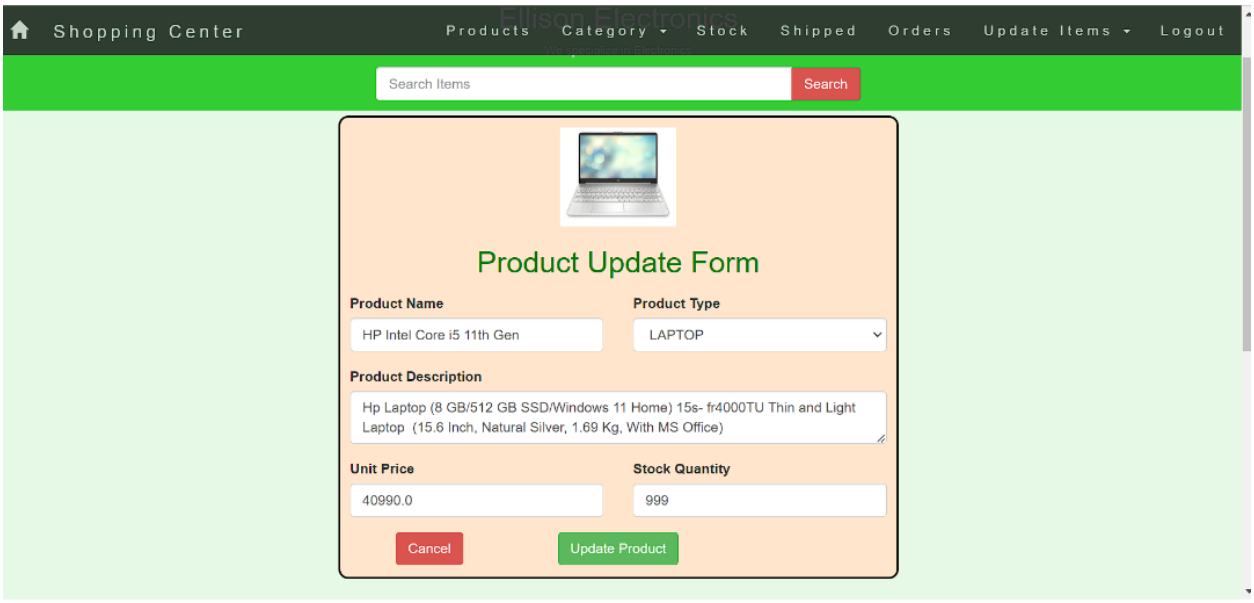
If the admin wishes to check the available products and more details of specific products such as product Id. Admin could click on the "Stock "button to acquire them. It allows the admin to choose which implementation they want to do for products.



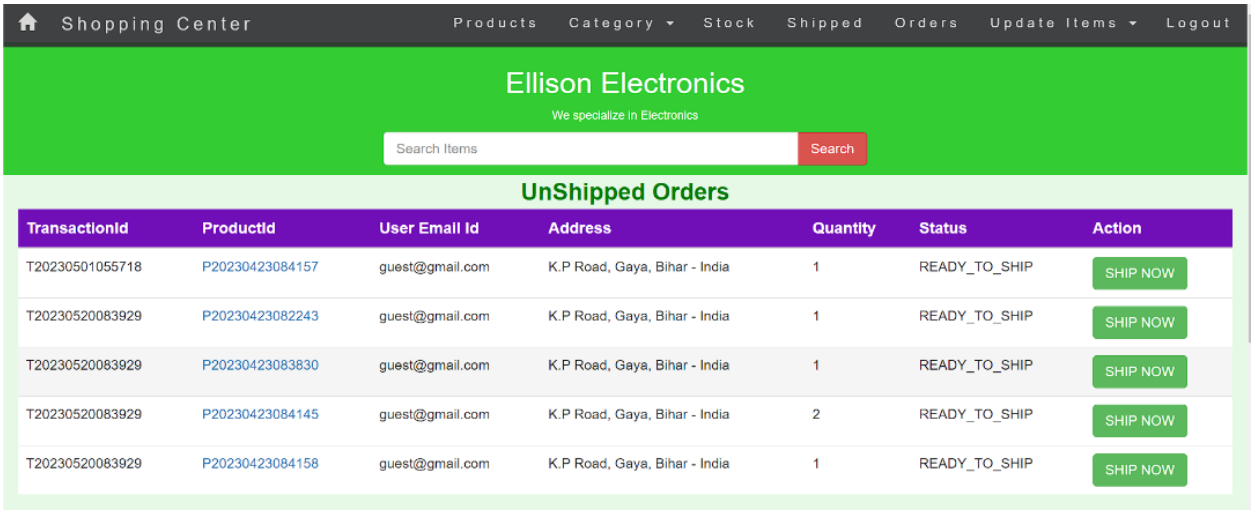
If the admin wants to remove the product, then they can click the “Remove “button and direct to” Remove product “page . Admin only needs to enter product Id and click “Remove Product “to confirm the removal of the product.



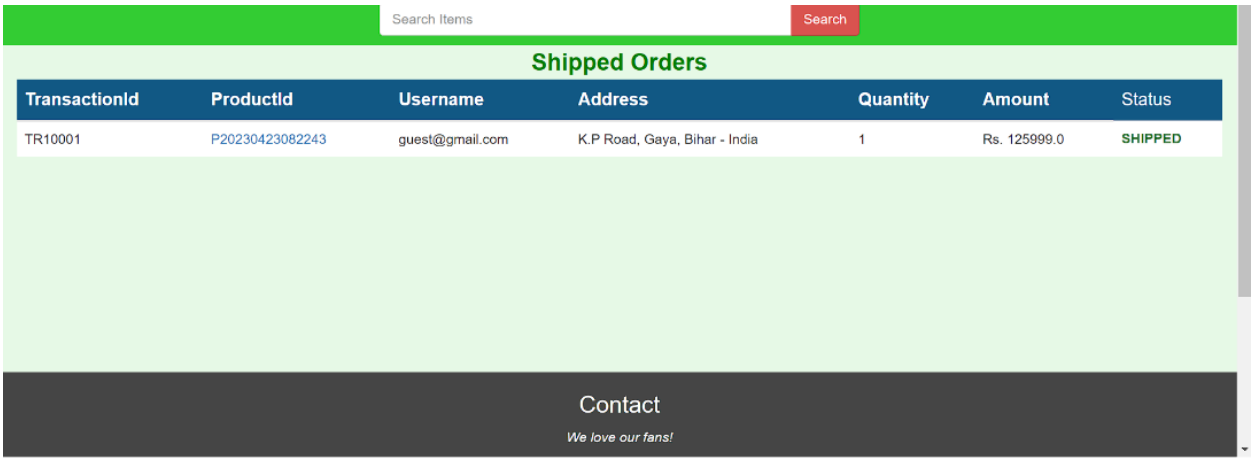
Similarly, if admin want to update products from this shopping cart website,(we will be adding a function that admin can set a product on sale or marked as second hand product in product update form.) Admin can modify the name, price, type, unit price and stock quantity even correct product description for this product. After it is done, admin will need to click the “Update Product” to confirm the modification of the product.



Admin could check placed orders that are purchased by users in shopping cart website. They can click the" Order" button to see user info, amount, and transaction Id etc. If the admin wants to ship the product to users after receiving orders. They can click the "SHIP NOW "button to start shipping products.



If the admin wants to know if the product has been shipped or not, they can click the " Shipped" button to see which products have been shipped by users. The webpage shows all necessary information to admin.



After the admin confirms the shipment of the product, they will automatically receive an email to remind them that the order has been shipped. It contains a receipt and product information in it.

We are glad that you shop with Ellison Electronics!

Your order has been placed successfully and under process to be shipped.

Please Note that this is a demo projet Email and you have not made any real transaction with us till now!

Here is Your Transaction Details:

Order Id: T20230428104705

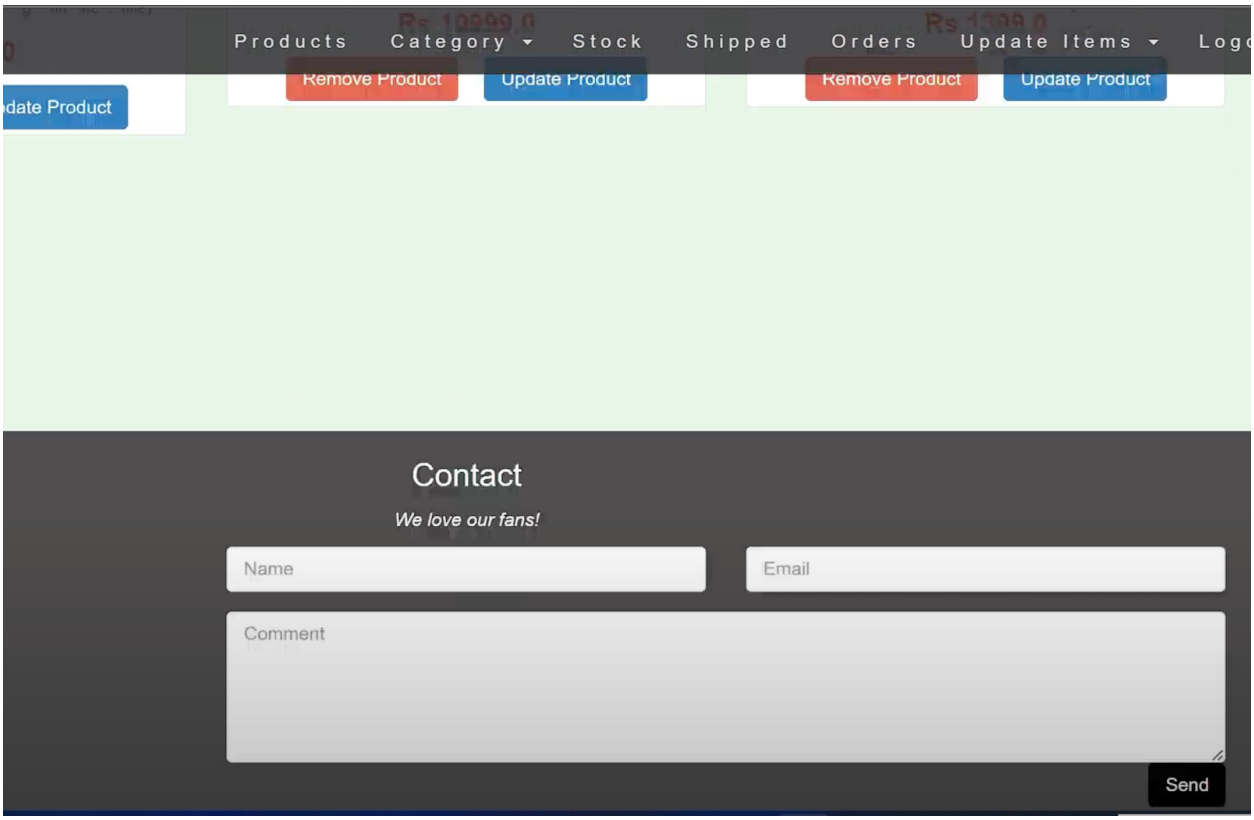
Amount Paid: 124988.0

Thanks for shopping with us!

Come Shop Again!
Ellison Electronics.

5. Comment to shopping Cart website

Users may use comments or contact with admin if they wish to complain or provide some advice to shopping car website service. They could also give feedback to the admin about their selling products. Users need to click at the very bottom of the page which is “Contact” to direct to the feedback block that appears at the bottom page. So, they can write down their feedback by giving their usernames and emails.



15. Appendix

Link to Comp354 Assignment 2:

https://github.com/8inversed/Comp354-Project

Individual contributions

Name	Responsibilities
Weilun Zhang	7. Use-Cases, 8. User-Stories, and 11. Software Design Strategies
Xu Zhang	3.Stakeholder Description, 4. Product Overview, 12. Architectural Level Design and 13. Component Level Design
Haoran Sun	1.Introduction, 2.Positioning, and 14. Interface Level Design
Qianrui Tao	9. UML (Activity Diagram) 10. Road Map 13. Component Level Design
Jingchao Song	9. UML (Use Case Diagram, Sequence Diagrams) 12 Architectural Level Design 13. Component Level Design
Zhaoyang Liu	5.Features, 6. Requirement, and 14. Interface Level Design