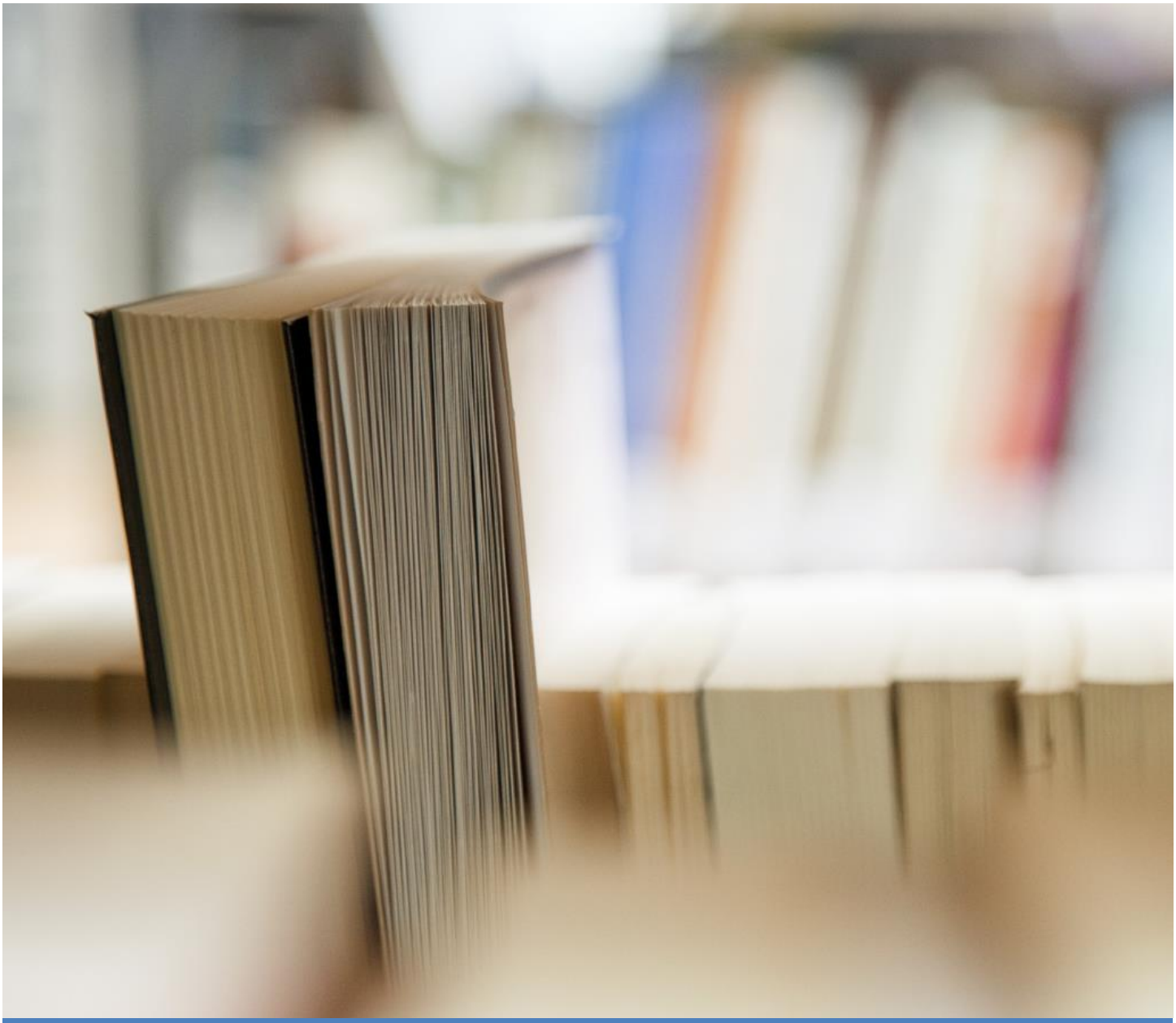

“Boi Lagbe!”-Online Bookstore System

Mohammad Mehedi Hasan Munna

9/4/25

CSE347

EAST WEST UNIVERSITY

“Boi Lagbe!”-Online Bookstore System

Course Name: Information System Analysis and Design

Course Code: CSE347 **Section:** 04

Submitted by:

Group 7

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124
Abdul Wadud	2022-2-60-133
Md Anik Khan Akash	2021-1-60-133

Submitted to:

Anika Tabassum

Lecturer

Department of Computer Science and Engineering

Letter of Transmittal

Anika Tabassum
Lecturer
East West University, Dhaka

Subject: Submission of term report on ““Boi Lagbe!”-Online Bookstore System”.

Ma’am,
With due respect, we would like to submit to you the report on Software Requirement Specification on the above topic you assigned us. The report reflects our effort to gather requirements and analyze them. We have included every step what we have taken throughout the whole time for requirement specification of the topic mentioned.
Therefore, we earnestly hope that you will excuse our error and obliged thereby.

Yours Sincerely,

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124
Abdul Wadud	2022-2-60-133
Md Anik Khan Akash	2023-1-60-074

East West University, Dhaka

Acknowledgement

We are highly indebted to getting such a tremendous opportunity to prepare the report on Online Bookstore System. We would like to thank wholeheartedly for our teacher Anika Tabassum, Lecturer, East West University, Dhaka for giving us guidelines about how We can
prepare this report. Additionally, we appreciate the individual who provided us with some crucial requirements. We also acknowledge the effort of our team members whose dedication and hard work made it possible.

Abstract:

All businesses and industries are working their way to adapt with the revolution of the internet. As a result, they need a management system whether it is a big or small industry. They require a user-friendly system to manage online handling of customer interaction, order processing and transaction management. The SRS document of "Boi Lagbe!"-Online Bookstore System" elaborates the idea of the features required for an optimized management system. The report aims to ensure a user-friendly and secure management system.

Table of Contents:

Table of Figure

List of Tables:

Chapter 1: Introduction:

1.1 Purpose

The objective of this chapter is to define the purpose of this Software Requirements Specification (SRS) document of ""Boi Lagbe!"-Online Bookstore System" and the intended audience of it. It includes a detailed guideline, functional and nonfunctional requirements which is essential for the system.

The requirements contained in the topics are organized, arranged, and numbered in a unique way, and they are independent from one another. This document acts as an essential point to ensure the understanding among all the stakeholders. It also helps us to understand the final output and meet the client's expectations.

1.2 Intended Audience

This specification will be used by the customer to check whether the developer
the team built a product as acceptable to the customer. The intended audience for this SRS includes Business Owner, admin, customer, developer etc.

- The business owner will go through this requirement specification to understand the functionalities that will help to manage the system efficiently
- Admin needs to have a clear idea about the whole E-commerce to handle the customers, adding products, or keeping track of the profit. It will be the most helpful for this.
- Customers can interact with the system smoothly by going through the documentation.
- The project manager can plan and allocate resources effectively to deliver the system at the mentioned time. To ensure that, the developing team must be on track.
- The specification used by developers and designers is to make sure the system development is matching the requirements.
- The testers will use this requirement specification file to derive test cases and verify that the software meets the document specification without any error. They will run the test when one of the sections of the software is complete to ensure that there are no missing things.

1.3 Conclusion:

This analysis of the audience serves as a significant foundation for developing the E-commerce Management System. This overall document will help to understand each stakeholder related to this project to have a better idea about the development process of the system.

Chapter 2: Inception

2.1 Introduction

The Inception phase is the beginning of requirement engineering. It focuses on how a software project gets started and if there is any scope to make it better by understanding the specific problems. The **Goal** of the inception phase is to identify concurrence needs and conflict requirements among the stakeholders of a software project. To establish the groundwork for the Inventory Management System, we have worked with the following factors related to the inception phases:

- Identifying stakeholders
- Understanding multiple viewpoints
- Establishing collaboration
- Addressing key questions

2.1.1 Identifying Stakeholder

A stakeholder is any person or group directly or indirectly affected by the system. Identifying them early is to make sure that their needs and concerns are being heard and integrated in the development process. End users are the system inter-actors and everybody else in an organization that its installation may affect. The primary stakeholders for this system include:

- **Business Owner:** The owner is responsible for overseeing operations and making strategic decisions. They run the business mainly recruits employees and managing any miscellaneous things. Also, she or he takes the cut of the net gain.

- **Employees:** This individual is a person who works part-time or full-time with some contact depending on the business. They have rights and duties recognized with it. They manage daily transactions and interact with customers. The employees also keep track of orders. They follow the instructions of the business owner.

- **Customers:** A person or any company that purchases any goods or services. They browse the system and if they take a liking to anything, they go for a purchase. After buying and using it they provide feedback which helps other customers get a view of the business.
  Customers are mainly attracted by offers and special combos at a cheaper rate than any other business.

- **Software Developers:** Developers are the ones to ensure the efficiency of the system and its functionality. Software developers implement the system by programming, researching, designing, and by developing the process. Primarily, they test the software. The outcome is

their responsibility whether it is efficient or not.

- **Courier Service:** The company that delivers the desired product to the customer by the delivery man working under the courier service. It is a very crucial part of the business.

  Products are delivered to the customer who lives far away from the business area. Pathao, Redx, Steadfast, SA Paribahan, etc. does the job of delivering the product without any hassle. If there is any problem caused by the courier, they try to adjust it through a refund or negotiation.

2.1.2 Recognizing Multiple Viewpoints

It is necessary to appreciate the needs of every stakeholder to develop a software product that satisfies everyone.

1.Owner's viewpoint

• Implementation of Error-Free Framework**:** The platform should function without any glitches.
• Profit Tracer**:** Get unlocked weekly reports and profit details by day.
• Business Keeper**:** All company data should be available in the proper format and easily accessible.
• Net Profit Accountant**:** Calculate the net profit after the reduction of liabilities and expenses.
• Customer Upgrades**:** Notify customers concerning new or upgraded products.
• Budget Balance**:** Operate within the spending limit.
• Security**:** Strong authentication should be used to protect the information.

2. Employee Point of View

• Easy-to-Navigate System**:** Allows a person to explore the interface with ease.
• Error-Free Operations**:** Test the system's reliability and functioning without failure.
• Inventory Management**:** Watch the remaining stock and the low-stock items.
• Daily revenue and cash flow reports**:** Get accurate daily profit and cash flow information.
• Product assessment**:** Determine which items are profitable and which are not.
• Supplier Communication**:** Communicate product needs with suppliers.
• Security**:** Strong authentication protection mechanisms should be implemented.

3. The User's Perspective

• Purchase Receipts**:** Always request a receipt for every purchase made.
• Product Search**:** View and search for products with ease.
• Online Shopping**:** Enjoy shopping with great ease for registered users.
• Special Benefits**:** As a registered client, you receive special benefits.

4. The Developer's Perspective

• Ease of Development**:** Develop software with minimal effort.
• No Software Errors**:** Provide a dependable and productive system.
• Requirements Are Well Defined**:** Execute tasks with clear and well-articulated requirements.
• Reasonable Funding**:** Receive a reasonable budget to fulfill the task effectively.

2.1.3 Working towards collaboration

It is common to encounter both common and conflicting perspectives when engaging with various stakeholders. These differences arise because each stakeholder has unique needs and expectations from the system. To effectively merge these perspectives into final requirements, we implemented the following collaborative strategy:

1. Identify Common and Conflicting Requirements
2. Define Requirement Categories
3. Set Prioritization Criteria
4. Make a conclusive decision on the requirements.

Common Requirements:

1. Advanced Web-Based Interface
2. The application can be accessed from any computer or mobile device that has internet access.
3. Allow any user to search for product
4. Maintain a database for all users and all products in the system

2.1.4 Asking the First Questions

We started by asking general questions to understand our customers and stakeholders, as well as the main **Goal**s and benefits of the Online Bookstore System. These questions helped us identify who will use the website, how it will benefit them, and whether there are other existing solutions that could meet their needs. Next, we asked more specific questions to better understand customers.
problems when shopping online and what features they would like to see. Finally, we focused on how well the website allows users to communicate, including customer support, order tracking, and feedback options, to ensure a smooth and enjoyable shopping experience.

2.2 Conclusion

The inception phase helped us get a basic understanding of the online Bookstore System, identify the customers and businesses that will benefit from an online shopping platform, define the main features of the system and start communication with stakeholders. More research and talks will help both the development team and business owners get a better idea of the project's potential. Our team believes that a fully developed website will create a strong base for future growth and improvements.

Chapter 3: Elicitation

This chapter specifies the Elicitation phase.

3.1 Introduction

Requirements Elicitation is an essential part of requirements engineering that focuses on gathering requirements from users, and other stakeholders. For our Online Bookstore system, effective requirements elicitation involved collaborating closely with stakeholders to ensure that customer needs, user expectations, and system functionalities were accurately captured to deliver a seamless shopping experience. Challenges faced during this process include understanding stakeholder needs, formulating relevant questions, and dealing with limited communication due to time constraints. Despite these obstacles, the requirements were successfully gathered, organized, and systematically.

3.2 Eliciting Requirements

During this phase, the goal was to bring together problem-solving, refinement, negotiation, and specification techniques to make sure we captured the requirements accurately. Close collaboration between all stakeholders played a key role in making this process successful.
Here's a look at the activities we focused on to gather and define the requirements:

1. Collaborative Requirements Gathering
2. Quality Function Deployment
3. Usage Scenarios
4. Elicitation Work Products

3.3 Collaborative Requirements Gathering

For the E-Commerce Management System (ECMS), various methods were used to gather the requirements, tailored to the project's needs:

- Meetings were held with business owners, customers, and administrators to understand their expectations for the system.
- Stakeholders were asked about the challenges they face with current systems and the features they would like to see in the new one.

- Based on the feedback, the final requirements were gathered, organized, and documented to ensure the system would meet the needs of all involved.

3.4 Quality Function Deployment

Quality Function Deployment (QFD) is a method used to turn customer needs into specific technical requirements for the software. It focuses on ensuring customer satisfaction throughout the software development process. For this project, QFD helped identify key requirements that are essential to meet customer expectations and deliver a successful product.

Here's an expanded list with even more ideas:

3.4.1 Normal Requirements

- Accessible online
- Product search functionality
- Admin user validation
- Login/logout functionality
- Role-based access control
- Shopping and checkout functionality
- Profit tracking reports
- Single installation on one machine
- Help/support feature
- Shop maintenance tracking
- Net profit calculation
- Accurate transaction records
- Product popularity analysis
- Supplier contact integration
- User account management
- Order tracking for users
- Automatic updates for product listings
- Integration with external payment gateways
- Refund and return management system
- Detailed sales reports and analysis

3.4.2 Expected Requirements

- Error-free software
- Secure user authentication
- User-friendly interface
- Smooth system performance
- Clear and well-defined features
- Regular data backup
- Cash on delivery and online payment options
- Notifications for product updates
- Admin data management capabilities
- Automatic product availability checks
- Push notifications for updates
- Multi-device compatibility
- Performance optimization for high traffic
- Scalable architecture for business growth
- Secure payment gateways
- Product category management
- User feedback collection for improvements
- Support for various currencies and international shipping

3.4.3 Exciting Requirements

- Interactive error messages
- Mobile login option
- Customer product ratings and reviews
- Social media login integration
- Live chat with admin
- Automatic SMS notifications for new arrivals
- Personalized product recommendations
- Integration with loyalty programs or rewards systems
- Special offers based on user behavior
- Integration with email marketing tools for personalized offers
- Subscription model for regular product deliveries
- Social proof features like trending

Chapter 4: Scenario Based Modeling

4.3 Use Case Descriptions

4.3.1 Authentication

4.3.1.1 Sign Up

- **Use case:** Sign Up
- **Primary Actor:** Customer
- **Goal in context:** To allow a new customer to create an account on the online bookstore.
- **Preconditions:**
    1. The system has a registration interface available.
    2. The customer does not already have an account with the email provided.
- **Trigger:** A customer clicks on the "Sign Up" button and attempts to submit registration details.
- **Scenario:**
    1. The customer visits the bookstore website's registration page.
    2. The customer fills in required information such as name, email, password, and contact details.
    3. The customer clicks the "Sign Up" button.
    4. The system validates all provided input (e.g., valid email format, password strength).
    5. Upon successful validation, the system creates the new customer account.
    6. The system sends a confirmation email or message to the customer, and their account is activated.
- **Exceptions:**
    1. **Duplicate Email/ID:** The email address provided is already registered in the system.
    2. **Invalid or Incomplete Data:** Required fields are empty, or data formats (e.g., email) are incorrect.
    3. **Password Policy Violation:** The chosen password does not meet the system's security requirements (e.g., too short, lacks special characters).
    4. **System Error:** An internal server or database error prevents account creation.
- **Priority:** High, must be implemented
- **When Available:** First increment

4.3.1.2 Sign In

- **Use case:** Sign In
- **Primary Actor:** Customer, Admin
- **Goal in context:** To allow a registered user (customer or admin) to access their account in the online bookstore.
- **Preconditions:**

1. The system has a login interface designed for user authentication.
2. The user attempting to log in must be already registered in the system.
3. The provided credentials (email/username and password) must be valid.

- **Trigger:** A user clicks the "Sign In" button after entering their credentials.
- **Scenario:**
    1. The user visits the login page of the website.
    2. The user enters their registered email/username and password.
    3. The user clicks the "Sign In" button.
    4. The system verifies the provided information against its database.
    5. After successful authentication, the system grants the user access to their respective dashboard (customer or admin).
- **Exceptions:**
    1. **Incorrect Username or Email:** The entered email/username does not exist in the system.
    2. **Wrong Password:** The entered password does not match the registered account.
    3. **Account Not Registered:** The user attempts to log in with credentials not found in the system.
    4. **Account Suspended/Locked:** The account is temporarily or permanently disabled.
- **Priority:** High, must be implemented
- **When Available:** First increment

4.3.1.3 Manage Account

- **Use case:** Manage User Account
- **Primary Actor:** Customer, Admin
- **Goal in context:** To allow a registered user or admin to update their account details, such as name, contact information, email, or password.
- **Preconditions:**
    1. The user (customer or admin) is registered and successfully logged into the system.
    2. The system has a dedicated interface for managing account details.
- **Trigger:** The user initiates a request to update their account information (e.g., clicks "Edit Profile").
- **Scenario:**
    1. The user visits their profile or account settings section.
    2. The user navigates to the "Manage Account" or "Edit Profile" section.
    3. The user updates personal information (e.g., name, shipping address, contact number, email, or password).

4. The system validates the new input to ensure correctness and adherence to rules (e.g., unique email, strong password).
5. The system prompts the user to confirm the changes.
6. The user confirms the changes.
7. The system updates the account in the database and displays a success message.

- **Exceptions:**
    1. **New Data Identical to Existing:** No changes are detected, so no update is performed.
    2. **Email Already Exists:** The new email address is already registered by another user.
    3. **Password Does Not Meet Requirements:** The new password is too short, common, or lacks complexity.
    4. **Invalid Input:** Fields contain invalid characters or are empty when required.
- **Priority:** Essential, must be implemented
- **When Available:** First increment

4.3.1.4 Sign Out

- **Use case:** Sign Out
- **Primary Actor:** Customer, Admin
- **Goal in context:** To allow a logged-in user (customer or admin) to securely terminate their session and log out from the website.
- **Preconditions:**
    1. The user is currently logged into their account.
    2. The system has a clear "Sign Out" or "Logout" interface element.
- **Trigger:** The user clicks the "Sign Out" or "Logout" button.
- **Scenario:**
    1. The user navigates to a page where the "Sign Out" option is available (e.g., header, user menu).
    2. The user clicks the "Sign Out" button.
    3. The system terminates the user's session.
    4. The user is redirected to the login screen or the main homepage, indicating a successful logout.
- **Exceptions:**
    1. **User Tries to Sign Out Without Being Logged In:** The system should prevent this action or indicate the user is not logged in.
    2. **System Error During Session Termination:** An error occurs that prevents a clean logout, potentially leaving session data.

- **Priority:** High, must be implemented
- **When Available:** First increment

4.3.2 Product Management

4.3.2.1 Browse Products

- **Use case:** Browse Products
- **Primary Actor:** Customer
- **Goal in context:** To enable customers to view a comprehensive list of available books in the online bookstore.
- **Preconditions:**
    1. There are books available in the system's catalog.
    2. The system has a user-friendly interface for displaying products.
- **Trigger:** A customer navigates to the "Shop" or "Products" section of the website.
- **Scenario:**
    1. The customer visits the website's homepage or directly navigates to the products page.
    2. The system retrieves and displays a paginated list of available books.
    3. The display includes basic information like book title, author, price, and a thumbnail image.
- **Exceptions:**
    1. **No Products Available:** If the catalog is empty, the system displays a message indicating no products are found.
    2. **System Error:** An error occurs during data retrieval or display.
- **Priority:** Essential, must be implemented
- **When Available:** First increment

4.3.2.2 Search Products

- **Use case:** Search Products
- **Primary Actor:** Customer
- **Goal in context:** To allow customers to efficiently find specific books or categories of books using various search criteria.
- **Preconditions:**
    1. The system has a search functionality integrated into the product browsing interface.
    2. The book catalog contains searchable information (e.g., title, author, ISBN, genre, publisher).
- **Trigger:** A customer enters a query into the search bar or applies specific filters.
- **Scenario:**

1. The customer enters keywords (e.g., book title, author's name) into the search bar.
2. Alternatively, the customer selects filters such as genre, price range, availability, or ratings.
3. The system processes the query or filters.
4. The system displays a list of books that match the specified criteria.
5. If no direct matches, the system may suggest similar items or related categories.

- **Exceptions:**
  1. **No Matching Results:** The search query or filters yield no books.
  2. **Invalid Search Query:** The input is malformed or too general.
  3. **System Error:** An error occurs during the search process.
- **Priority:** Essential, must be implemented
- **When Available:** First increment

4.3.2.3 View Product Details

- **Use case:** View Product Details
- **Primary Actor:** Customer
- **Goal in context:** To enable a customer to view comprehensive information about a specific book before making a purchase decision.
- **Preconditions:**
  1. The selected product exists in the system's catalog.
  2. The system has a detailed product page interface.
- **Trigger:** A customer clicks on a book from a browsing or search results list.
- **Scenario:**
  1. The customer clicks on a book's title or image from the product listing.
  2. The system navigates to the dedicated product detail page.
  3. The system displays extensive information about the book, including:
     - Full title and author.
     - Detailed description/synopsis.
     - Price and availability status.
     - High-resolution images.
     - ISBN, publisher, publication date.
     - Customer reviews and ratings.
     - Related products or recommendations.

4. Options to "Add to Cart" are clearly visible.

- **Exceptions:**
    1. **Product Not Found:** The system cannot find the requested product (e.g., product removed).
    2. **Broken Link:** An invalid URL leads to an error page.
    3. **System Error:** Data retrieval or display fails.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.2.4 Add Product (Admin)

- **Use case:** Add Product
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to add new books to the online bookstore's inventory and make them available for sale.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The system provides an interface for adding new products.
- **Trigger:** The Admin navigates to the "Add Product" section and initiates the process.
- **Scenario:**
    1. The Admin logs into the admin dashboard.
    2. The Admin navigates to the "Products" or "Book Management" section.
    3. The Admin clicks on an "Add New Product" button or link.
    4. The system presents a form for entering product details.
    5. The Admin fills in all necessary information: title, author, ISBN, category, price, stock quantity, description, and uploads an image.
    6. The Admin submits the form.
    7. The system validates the input (e.g., unique ISBN, valid numbers, image format).
    8. Upon successful validation, the system adds the new product to the database and makes it visible in the catalog.
- **Exceptions:**
    1. **Invalid/Incomplete Data:** Mandatory fields are empty, or data formats are incorrect.
    2. **Duplicate ISBN:** A product with the same ISBN already exists.
    3. **Image Upload Failed:** The uploaded image is too large, an unsupported format, or an upload error occurs.

4. **Permission Denied:** The Admin account lacks the necessary permissions to add products.

- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.2.5 Update Product (Admin)

- **Use case:** Update Product
- **Primary Actor:** Admin
- **Goal in context:** To enable an Admin to modify the details of existing books in the product catalog.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The product to be updated exists in the database.
    3. The system provides an interface for editing product details.
- **Trigger:** The Admin selects an existing product for modification.
- **Scenario:**
    1. The Admin logs into the admin dashboard and navigates to the "Products" section.
    2. The Admin searches for or selects the specific product they wish to update.
    3. The Admin clicks an "Edit" button next to the selected product.
    4. yThe system pre-fills the product details form with the current information.
    5. The Admin modifies any desired fields (e.g., price, description, stock, image).
    6. The Admin submits the updated form.
    7. The system validates the modified input.
    8. Upon successful validation, the system updates the product's details in the database.
- **Exceptions:**
    1. **Product Not Found:** The selected product does not exist.
    2. **Invalid Data:** The modified data does not meet validation rules.
    3. **Conflict:** Another Admin updated the same product concurrently.
    4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.2.6 Delete Product (Admin)

- **Use case:** Delete Product
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to permanently remove a book from the product catalog.

- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The product to be deleted exists in the database.
- **Trigger:** The Admin selects an existing product for deletion.
- **Scenario:**
    1. The Admin navigates to the "Products" section in the admin dashboard.
    2. The Admin selects the specific product they wish to delete.
    3. The Admin clicks a "Delete" button.
    4. The system prompts the Admin for confirmation of the deletion (e.g., "Are you sure?").
    5. The Admin confirms the deletion.
    6. The system removes the product and associated data (e.g., image files, stock records) from the database.
- **Exceptions:**
    1. **Product Not Found:** The selected product does not exist.
    2. **Confirmation Cancelled:** The Admin cancels the deletion process.
    3. **Deletion Failed:** An error occurs during the deletion process (e.g., database constraint, active orders associated).
    4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.2.7 Manage Stock (Admin)

- **Use case:** Manage Stock
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to adjust the inventory levels of books in the system.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The product for which stock is managed exists.
    3. The system provides an interface for stock management.
- **Trigger:** The Admin navigates to the stock management section or initiates a stock update for a product.
- **Scenario:**
    1. The Admin accesses the stock management interface (either within product edit or a dedicated stock page).
    2. The Admin identifies the product for which stock needs adjustment.

3. The Admin enters the new stock quantity or adjusts it (e.g., increment/decrement).
4. The Admin confirms the update.
5. The system validates the new stock quantity (e.g., non-negative).
6. The system updates the inventory count for the specific product in the database.

- **Exceptions:**
  1. **Invalid Stock Quantity:** The entered quantity is negative or non-numeric.
  2. **Product Not Found:** The product's stock cannot be managed because the product does not exist.
  3. **System Error:** An error occurs during the update process.
  4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3 Order Management

4.3.3.1 Add to Cart

- **Use case:** Add to Cart
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to add a chosen book to their shopping cart for potential purchase.
- **Preconditions:**
  1. The book is available for purchase (in stock).
  2. The customer is viewing the product details or browsing products.
- **Trigger:** The customer clicks the "Add to Cart" button on a product.
- **Scenario:**
  1. The customer finds a book they wish to purchase.
  2. The customer specifies the desired quantity (if applicable).
  3. The customer clicks the "Add to Cart" button.
  4. The system checks product availability and quantity.
  5. The system adds the specified book and quantity to the customer's shopping cart.
  6. The system provides visual confirmation (e.g., "Item added to cart," cart icon updates).
- **Exceptions:**
  1. **Product Out of Stock:** The book is no longer available in the requested quantity.
  2. **Invalid Quantity:** The customer attempts to add a zero or negative quantity.
  3. **System Error:** An error prevents the item from being added to the cart.
- **Priority:** Essential, Must be implemented

- **When Available:** First increment

4.3.3.2 View Cart

- **Use case:** View Cart
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to review the contents of their shopping cart before proceeding to checkout.
- **Preconditions:**
    1. The customer has at least one item in their shopping cart.
    2. The system provides a clear access point to the shopping cart (e.g., cart icon, "View Cart" button).
- **Trigger:** The customer clicks on the shopping cart icon or a "View Cart" link.
- **Scenario:**
    1. The customer clicks on the shopping cart icon/link.
    2. The system navigates to the shopping cart page.
    3. The system displays a list of all items currently in the cart, including:
        - Product name and image.
        - Quantity of each item.
        - Unit price and total price per item.
        - Overall subtotal for all items in the cart.
    4. Options to update quantities, remove items, or proceed to checkout are available.
- **Exceptions:**
    1. **Cart Is Empty:** The system displays a message indicating the cart is empty.
    2. **System Error:** An error occurs while retrieving or displaying cart contents.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.3 Update Cart

- **Use case:** Update Cart
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to modify the quantities of items within their shopping cart.
- **Preconditions:**
    1. The customer is viewing their shopping cart.
    2. The system allows for modification of item quantities.
- **Trigger:** The customer changes the quantity of an item in the cart.
- **Scenario:**

1. On the cart page, the customer changes the numerical quantity for a specific book.
2. The system automatically (or after a "Update Cart" click) updates the quantity of that item.
3. The system recalculates the subtotal for that item and the overall cart total.
4. The updated cart contents are displayed.

- **Exceptions:**
  1. **Invalid Quantity:** The customer enters a non-positive or non-numeric quantity.
  2. **Quantity Exceeds Stock:** The requested quantity exceeds the available stock for that book.
  3. **System Error:** An error occurs during the update process.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.4 Remove from Cart

- **Use case:** Remove from Cart
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to remove specific items from their shopping cart.
- **Preconditions:**
  1. The customer is viewing their shopping cart.
  2. The item to be removed is present in the cart.
- **Trigger:** The customer clicks the "Remove" or "Delete" button next to an item in the cart.
- **Scenario:**
  1. On the cart page, the customer identifies an item they no longer wish to purchase.
  2. The customer clicks the "Remove" button associated with that item.
  3. The system removes the item from the shopping cart.
  4. The system recalculates the overall cart total.
  5. The updated cart contents are displayed.
- **Exceptions:**
  1. **Item Not Found:** The system cannot find the specified item in the current cart.
  2. **System Error:** An error occurs during the removal process.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.5 Checkout

- **Use case:** Checkout
- **Primary Actor:** Customer

- **Goal in context:** To allow a customer to complete the process of purchasing items in their shopping cart.
- **Preconditions:**
    1. The customer has items in their shopping cart.
    2. The customer is logged in (or can proceed as guest).
    3. All required information for shipping and payment is either available or can be provided.
- **Trigger:** The customer clicks the "Proceed to Checkout" button from the shopping cart.
- **Scenario:**
    1. The customer clicks "Proceed to Checkout".
    2. The system guides the customer through steps:
        - Review order summary (items, total cost).
        - Enter or confirm shipping address.
        - Select a payment method (e.g., bKash, Nagad, Credit/Debit Card).
        - Enter payment details (if not saved).
    3. The customer confirms the order.
    4. The system processes the payment and finalizes the order.
    5. The system displays an order confirmation page and sends an order confirmation (email/SMS).
    6. The system updates inventory for purchased items.
- **Exceptions:**
    1. **Payment Failure:** The payment transaction is declined or fails.
    2. **Invalid Shipping Address:** The provided address is incomplete or invalid.
    3. **Stock Issues:** An item in the cart becomes out of stock before the order is finalized.
    4. **System Error:** An error occurs during order processing.
- **Priority:** Essential, Must be implemented
- **When Available:** First increment

4.3.3.6 View Order History

- **Use case:** View Order History
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to review a list of all their past purchases and their current statuses.
- **Preconditions:**
    1. The customer is logged into their account.
    2. The customer has previously placed at least one order.
    3. The system provides an interface to view order history.

- **Trigger:** The customer navigates to the "Order History" or "My Orders" section of their account.
- **Scenario:**
    1. The customer logs in and navigates to their account dashboard.
    2. The customer clicks on the "Order History" link.
    3. The system retrieves and displays a chronological list of the customer's past orders.
    4. For each order, key details like Order ID, date, total amount, and current status (e.g., Pending, Processing, Completed, Cancelled) are shown.
    5. The customer can click on individual orders to view more detailed information.
- **Exceptions:**
    1. **No Order History Found:** The system displays a message indicating no past orders exist for this account.
    2. **System Error:** An error occurs during data retrieval or display.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.7 Manage Orders (Admin)

- **Use case:** Manage Orders
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to view, filter, and access details of all customer orders placed in the bookstore.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. There are customer orders recorded in the system.
- **Trigger:** The Admin navigates to the "Orders" section in the admin dashboard.
- **Scenario:**
    1. The Admin logs into the admin dashboard.
    2. The Admin clicks on the "Orders" menu item.
    3. The system displays a comprehensive list of all customer orders.
    4. The Admin can use filters (e.g., by status, date range, customer name) and sorting options to manage the list.
    5. Each order in the list shows essential information like Order ID, Customer Name, Total Amount, Date, and current Status.
    6. The Admin can click on any order to view its full details.
- **Exceptions:**

1. **No Orders Found:** The system displays a message indicating no orders are currently in the system.
2. **System Error:** An error occurs during data retrieval or display.
3. **Permission Denied:** The Admin account lacks the necessary permissions to view orders.

- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.8 Process Order (Admin)

- **Use case:** Process Order
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to initiate the fulfillment process for a new or pending customer order.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. An order exists with a status that allows processing (e.g., "Pending").
- **Trigger:** The Admin selects a pending order and chooses to begin processing it.
- **Scenario:**
    1. The Admin views the list of orders and identifies a new or pending order.
    2. The Admin opens the detailed view of that order.
    3. The Admin clicks a "Process Order" or similar button.
    4. The system updates the order's status to "Processing".
    5. The system might trigger internal notifications or processes for warehouse/shipping.
    6. The system may send an automated notification to the customer about the updated status.
- **Exceptions:**
    1. **Order Not Found:** The selected order does not exist or has been removed.
    2. **Invalid Order Status:** The order is already processed or cancelled and cannot be re-processed.
    3. **System Error:** An error occurs during the status update or associated processes.
    4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.9 Update Order Status (Admin)

- **Use case:** Update Order Status
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to change the fulfillment status of a customer order.
- **Preconditions:**

1. The Admin is logged into the admin panel.
2. An order exists and its status can be changed.

- **Trigger:** The Admin manually changes the status of an order.
- **Scenario:**
    1. The Admin navigates to an order's detailed view in the admin panel.
    2. The Admin selects a new status from a dropdown or clicks a status-changing button (e.g., "Mark as Completed," "Cancel Order").
    3. The system prompts for confirmation if the change is significant (e.g., cancelling).
    4. The system updates the order's status in the database.
    5. The system sends an automated notification (email/SMS) to the customer about the status change.
- **Exceptions:**
    1. **Invalid Status Transition:** The selected new status is not a valid progression from the current status (e.g., cannot go from "Completed" to "Pending").
    2. **Order Not Found:** The order being updated does not exist.
    3. **System Error:** An error occurs during the update process.
    4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.3.10 Process Refund (Admin)

- **Use case:** Process Refund
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to issue a refund to a customer for a cancelled or returned order.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. An order exists and is eligible for a refund (e.g., cancelled, returned).
    3. The system is integrated with the payment gateway for refund capabilities.
- **Trigger:** The Admin initiates a refund for a specific order.
- **Scenario:**
    1. The Admin navigates to an order's detailed view in the admin panel.
    2. The Admin clicks a "Process Refund" button.
    3. The system may prompt for the refund amount (full or partial) and a reason.
    4. The Admin confirms the refund details.

5. The system sends a request to the integrated payment gateway to process the refund.
6. Upon confirmation from the payment gateway, the system updates the order's status to "Refunded" and records the transaction.
7. The system sends a refund confirmation notification to the customer.

- **Exceptions:**
  1. **Order Not Found:** The order for which the refund is initiated does not exist.
  2. **Payment Gateway Error:** The refund transaction fails at the payment gateway level.
  3. **Insufficient Funds:** The bookstore's account has insufficient funds to cover the refund (rare, but possible for direct merchant accounts).
  4. **Invalid Refund Amount:** The refund amount exceeds the original order amount or is zero/negative.
  5. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.4 Payment

4.3.4.1 Make Online Payment

- **Use case:** Make Online Payment
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to securely complete payment for their order using various online payment methods integrated with the bookstore.
- **Preconditions:**
  1. The customer has reached the payment step of the checkout process.
  2. The system has integrated online payment gateways (e.g., bKash, Nagad, Credit/Debit Card).
  3. The customer has a valid payment method ready.
- **Trigger:** The customer selects an online payment method and confirms their choice during checkout.
- **Scenario:**
  1. During checkout, the customer selects their preferred online payment method (e.g., bKash, Nagad, Credit/Debit Card).
  2. The system either redirects the customer to the payment gateway's secure page or presents an in-site payment prompt.
  3. The customer provides the necessary payment details (e.g., mobile account number, OTP, card details).
  4. The payment gateway processes the transaction.
  5. Upon successful payment, the payment gateway sends a confirmation back to the bookstore system.

6. The system updates the order status to "Processing" or "Completed" and confirms the payment to the customer.

- **Exceptions:**
    1. **Payment Gateway Not Responding:** The connection to the payment gateway fails.
    2. **Transaction Declined:** The customer's bank or payment provider declines the transaction.
    3. **Insufficient Funds:** The customer's account/card does not have enough balance.
    4. **Invalid Payment Details:** The customer provides incorrect card numbers, expiry dates, or OTPs.
    5. **Network Error:** A disruption occurs during the payment process.
- **Priority:** Essential, Must be implemented
- **When Available:** First increment

4.3.5 Contact/Support

4.3.5.1 Send Message

- **Use case:** Send Message
- **Primary Actor:** Customer
- **Goal in context:** To allow a customer to send a message (query, feedback, or support request) directly to the bookstore's administration or support team.
- **Preconditions:**
    1. The system has a functional contact form or messaging interface available to customers.
- **Trigger:** A customer fills out the contact form and clicks the "Send" or "Submit" button.
- **Scenario:**
    1. The customer navigates to the "Contact Us" page on the website.
    2. The customer fills in their name, email address, subject, and the message content.
    3. The customer clicks the "Send Message" button.
    4. The system validates the input (e.g., valid email format, message content not empty).
    5. Upon successful validation, the system saves the message in the database and may send an automated confirmation to the customer.
    6. The system notifies the Admin about the new message.
- **Exceptions:**
    1. **Invalid Input:** Required fields are left empty, or the email format is incorrect.
    2. **Message Submission Failed:** A server or database error prevents the message from being saved.
    3. **Spam Detection:** The message is flagged as spam and rejected.

- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.5.2 View Messages (Admin)

- **Use case:** View Messages
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to view and review messages sent by customers through the contact form.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. There are customer messages recorded in the system.
- **Trigger:** The Admin navigates to the "Messages" section in the admin dashboard.
- **Scenario:**
    1. The Admin logs into the admin dashboard.
    2. The Admin clicks on the "Messages" menu item.
    3. The system displays a list of all customer messages, typically showing sender, subject, and date.
    4. The Admin can click on individual messages to view their full content and associated contact information.
- **Exceptions:**
    1. **No Messages Found:** The system displays a message indicating that there are no customer messages.
    2. **System Error:** An error occurs during data retrieval or display.
    3. **Permission Denied:** The Admin account lacks the necessary permissions to view messages.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.5.3 Reply to Message (Admin)

- **Use case:** Reply to Message
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to send a response to a customer's message.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The Admin has selected and opened a specific customer message.
    3. The system provides a mechanism for sending replies.
- **Trigger:** The Admin chooses to reply to an opened message.
- **Scenario:**
    1. The Admin views a customer message in detail.

2. The Admin clicks a "Reply" button or accesses a reply form.
3. The system opens a reply interface, pre-filling the customer's email address.
4. The Admin composes their response.
5. The Admin clicks "Send Reply."
6. The system sends the reply via email to the customer and optionally records the reply in the message history.

- **Exceptions:**
  1. **Message Not Found:** The original message for which the reply is intended does not exist.
  2. **Reply Sending Failed:** An error occurs during the email sending process.
  3. **Invalid Reply Content:** The reply is empty or contains problematic content.
  4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.6 User Management (Admin)

4.3.6.1 View Users (Admin)

- **Use case:** View Users
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to view a comprehensive list of all registered customer and admin accounts in the system.
- **Preconditions:**
  1. The Admin is logged into the admin panel.
  2. There are user accounts (customer or admin) registered in the system.
- **Trigger:** The Admin navigates to the "Users" or "User Management" section in the admin dashboard.
- **Scenario:**
  1. The Admin logs into the admin dashboard.
  2. The Admin clicks on the "Users" menu item.
  3. The system displays a paginated list of all registered users.
  4. For each user, basic details like name, email, and user type (customer/admin) are shown.
  5. The Admin can apply filters or sorting options to manage the list (e.g., filter by user type).
- **Exceptions:**
  1. **No Users Found:** The system displays a message indicating no registered users.

2. **System Error:** An error occurs during data retrieval or display.
3. **Permission Denied:** The Admin account lacks the necessary permissions to view users.

- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.6.2 Add User (Admin)

- **Use case:** Add User
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to manually create a new user account (e.g., for internal staff, or a customer who registered offline).
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The system provides an interface for adding new users.
- **Trigger:** The Admin navigates to the "Add User" section and initiates the process.
- **Scenario:**
    1. The Admin logs into the admin dashboard and navigates to the "Users" section.
    2. The Admin clicks on an "Add New User" button or link.
    3. The system presents a form for entering user details.
    4. The Admin fills in required information: name, email, password, and selects the user type (customer or admin).
    5. The Admin submits the form.
    6. The system validates the input (e.g., unique email, valid password).
    7. Upon successful validation, the system creates the new user account in the database.
- **Exceptions:**
    1. **Duplicate Email:** The email address provided is already associated with an existing account.
    2. **Invalid/Incomplete Data:** Mandatory fields are empty, or data formats are incorrect.
    3. **Password Policy Violation:** The chosen password does not meet system requirements.
    4. **Permission Denied:** The Admin account lacks the necessary permissions to add users.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.6.3 Update User (Admin)

- **Use case:** Update User

- **Primary Actor:** Admin
- **Goal in context:** To enable an Admin to modify the details of an existing user account.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The user account to be updated exists in the database.
    3. The system provides an interface for editing user details.
- **Trigger:** The Admin selects an existing user account for modification.
- **Scenario:**
    1. The Admin logs into the admin dashboard and navigates to the "Users" section.
    2. The Admin searches for or selects the specific user account they wish to update.
    3. The Admin clicks an "Edit" button next to the selected user.
    4. The system pre-fills the user details form with the current information.
    5. The Admin modifies any desired fields (e.g., name, email, password, user type).
    6. The Admin submits the updated form.
    7. The system validates the modified input.
    8. Upon successful validation, the system updates the user's details in the database.
- **Exceptions:**
    1. **User Not Found:** The selected user account does not exist.
    2. **Invalid Data:** The modified data does not meet validation rules.
    3. **Email Already Exists:** The new email address is already used by another account.
    4. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.6.4 Delete User (Admin)

- **Use case:** Delete User
- **Primary Actor:** Admin
- **Goal in context:** To allow an Admin to permanently remove a user account from the system.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. The user account to be deleted exists in the database.
- **Trigger:** The Admin selects an existing user account for deletion.

- **Scenario:**
    1. The Admin navigates to the "Users" section in the admin dashboard.
    2. The Admin selects the specific user account they wish to delete.
    3. The Admin clicks a "Delete" button.
    4. The system prompts the Admin for confirmation of the deletion.
    5. The Admin confirms the deletion.
    6. The system removes the user account and associated data (e.g., profile info, order history if cascade delete is enabled) from the database.
- **Exceptions:**
    1. **User Not Found:** The selected user account does not exist.
    2. **Confirmation Cancelled:** The Admin cancels the deletion process.
    3. **Deletion Failed:** An error occurs during the deletion process (e.g., database constraint if orders are linked and not cascaded).
    4. **Permission Denied:** The Admin account lacks the necessary permissions or attempts to delete the primary admin account.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.7 Admin Dashboard/Reporting

4.3.7.1 View Dashboard Summary (Admin)

- **Use case:** View Dashboard Summary
- **Primary Actor:** Admin
- **Goal in context:** To provide the Admin with a concise, high-level overview of the bookstore's operational status and key performance indicators upon logging in.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. System data (e.g., sales, products, users, orders) is available.
- **Trigger:** The Admin successfully logs into the admin panel or navigates to the "Dashboard" section.
- **Scenario:**
    1. The Admin logs in.
    2. The system automatically displays the main dashboard page.
    3. The dashboard presents key summary statistics and quick links, such as:
        - Total sales amount.

        - Number of active products.
        - Number of registered users.
        - Number of pending orders.
        - Recent sales trends (e.g., simple line chart).
        - Quick links to manage products, orders, and users.
- **Exceptions:**
    1. **Data Not Available:** Specific metrics might not load if underlying data is missing or corrupted.
    2. **System Error:** An error occurs during dashboard data aggregation or display.
    3. **Permission Denied:** The Admin account lacks permission to view certain metrics.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3.7.2 View Sales Reports (Admin)

- **Use case:** View Sales Reports
- **Primary Actor:** Admin
- **Goal in context:** To allow the Admin to analyze detailed sales performance, trends, and revenue generated over various periods.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. Sales transaction data is recorded in the system.
- **Trigger:** The Admin selects "Sales Reports" from the dashboard or navigation menu.
- **Scenario:**
    1. The Admin navigates to the "Sales Reports" section.
    2. The system presents options to filter reports by date range (e.g., daily, weekly, monthly, yearly) or specific criteria.
    3. The Admin selects desired filters.
    4. The system generates and displays detailed sales data, which may include:
        - Total revenue for the selected period.
        - Number of orders.
        - Average order value.
        - Sales trends visualized through charts (e.g., line graph for sales over time).
        - Top-selling books or categories.

- **Exceptions:**
    1. **No Sales Data:** No sales records exist for the selected period.
    2. **Report Generation Error:** An error occurs during data processing or chart rendering.
    3. **Invalid Date Range:** The selected date range is illogical (e.g., end date before start date).
    4. **Permission Denied:** The Admin account lacks the necessary permissions to view sales data.
- **Priority:** Important, Should be implemented
- **When Available:** First increment

4.3.7.3 View Category Reports (Admin)

- **Use case:** View Category Reports
- **Primary Actor:** Admin
- **Goal in context:** To allow the Admin to gain insights into the performance and popularity of different book categories.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. Books are categorized, and sales data by category is available.
- **Trigger:** The Admin selects "Category Reports" from the dashboard or navigation menu.
- **Scenario:**
    1. The Admin navigates to the "Category Reports" section.
    2. The system generates and displays data related to book categories, which may include:
        - Number of books in each category.
        - Sales figures per category.
        - Popularity rankings of categories (e.g., using a doughnut chart for distribution).
        - Trends in category performance over time.
- **Exceptions:**
    1. **No Category Data:** Categories are not defined or associated with books, or no sales data for categories exists.
    2. **Report Generation Error:** An error occurs during data processing or chart rendering.
    3. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** Important, Should be implemented
- **When Available:** First increment

4.3.7.4 View Order Summaries (Admin)

- **Use case:** View Order Summaries
- **Primary Actor:** Admin
- **Goal in context:** To provide the Admin with aggregated information about orders, such as counts by status, to assess operational workload.
- **Preconditions:**
    1. The Admin is logged into the admin panel.
    2. Order data with various statuses exists in the system.
- **Trigger:** The Admin accesses the Dashboard or navigates to an "Order Summary" section.
- **Scenario:**
    1. The Admin views the dashboard or a dedicated order summary page.
    2. The system displays aggregated order data, such as:
        - Total number of orders.
        - Count of pending orders.
        - Count of processing orders.
        - Count of completed orders.
        - Count of cancelled orders.
        - Visualizations (e.g., bar charts) representing these counts.
    3. The Admin can use this summary to quickly identify backlogs or trends in order fulfillment.
- **Exceptions:**
    1. **No Order Data:** No orders have been placed in the system.
    2. **Summary Generation Error:** An error occurs during data aggregation or display.
    3. **Permission Denied:** The Admin account lacks the necessary permissions.
- **Priority:** High, Must be implemented
- **When Available:** First increment

4.3 Use Case Diagram

Figure 1: Level 0 of Online Bookstore System

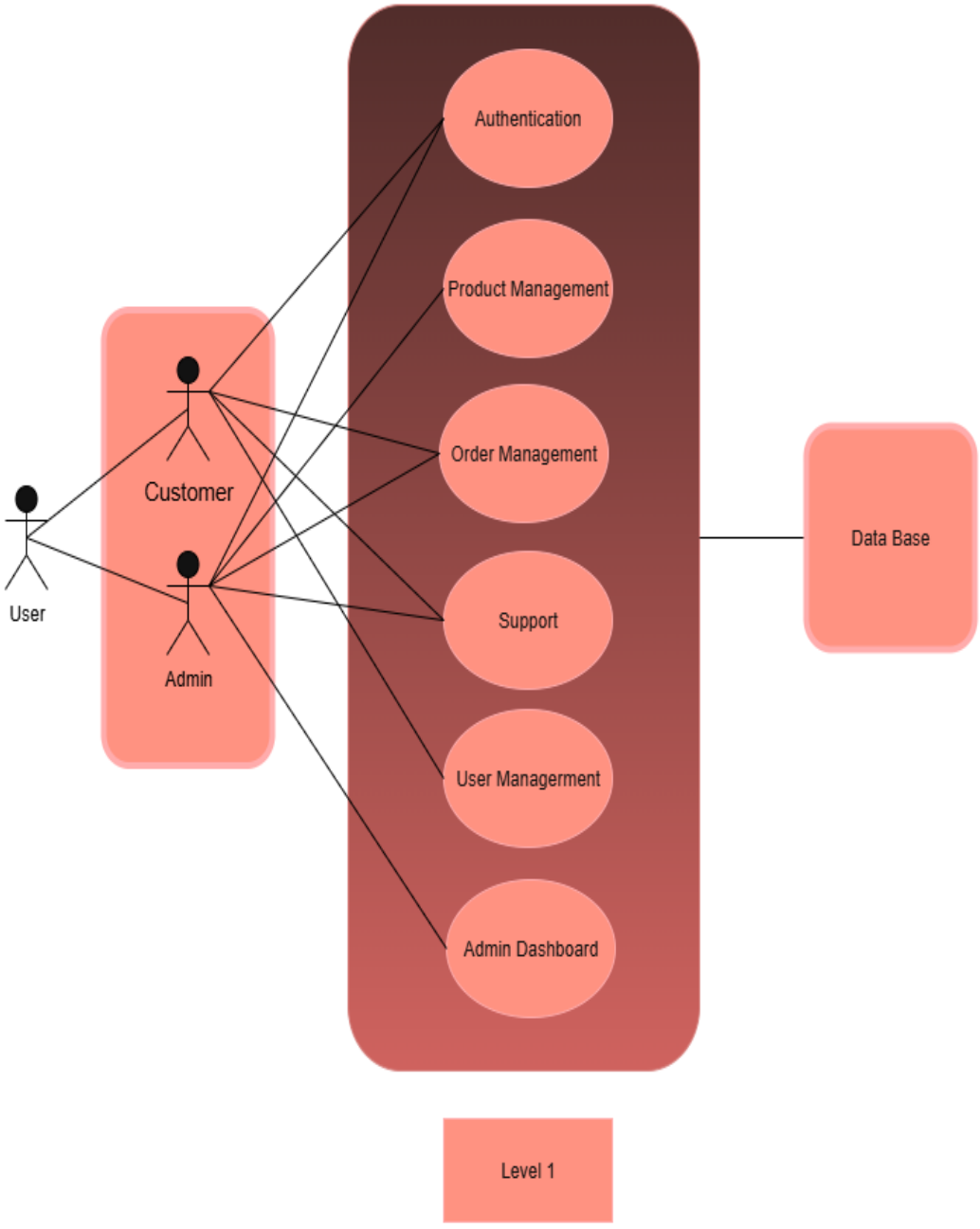

Figure 2: Level 1 of Online Bookstore System

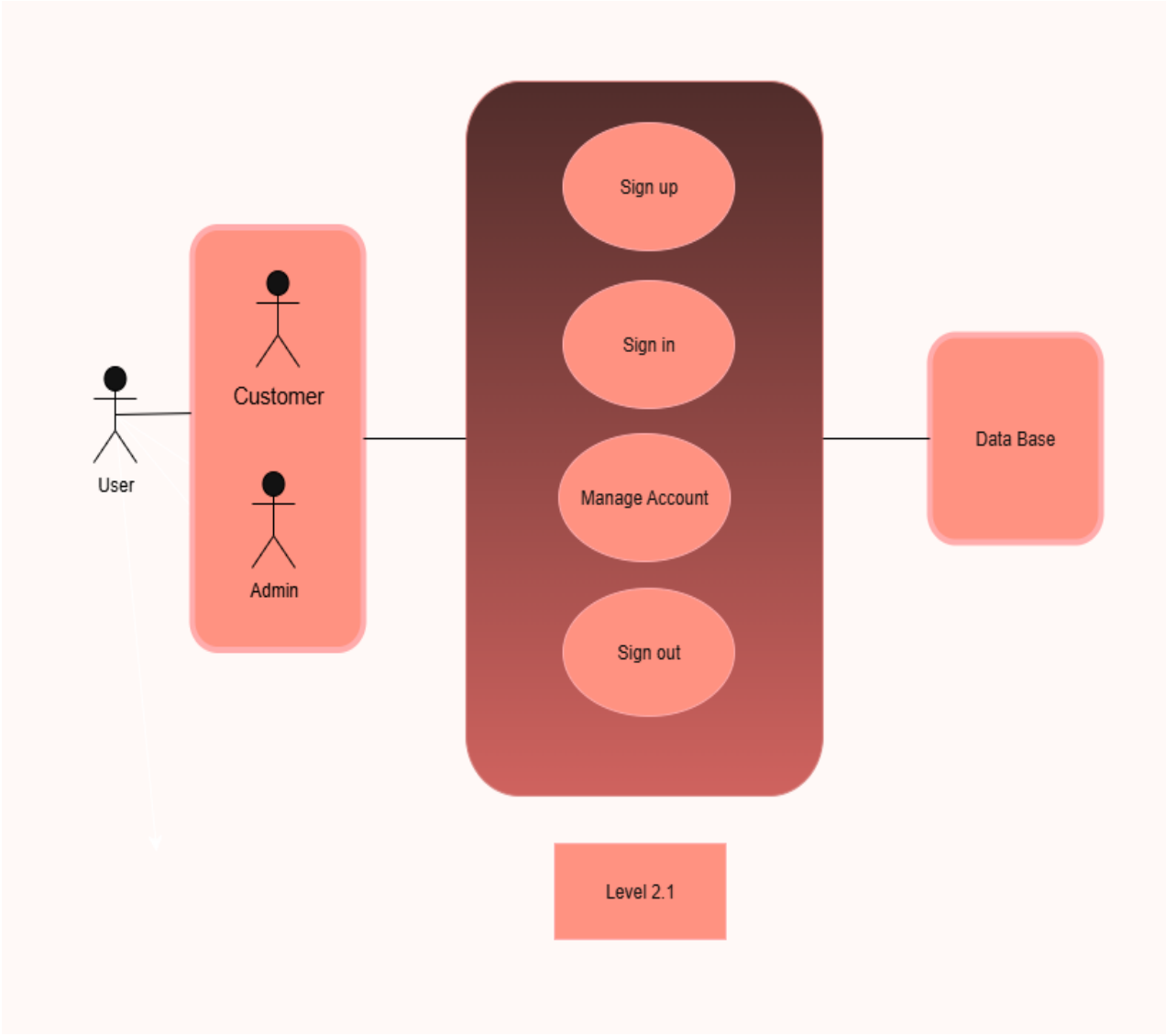

Figure 3: Level 2.1 of Online Bookstore System (Authentication)

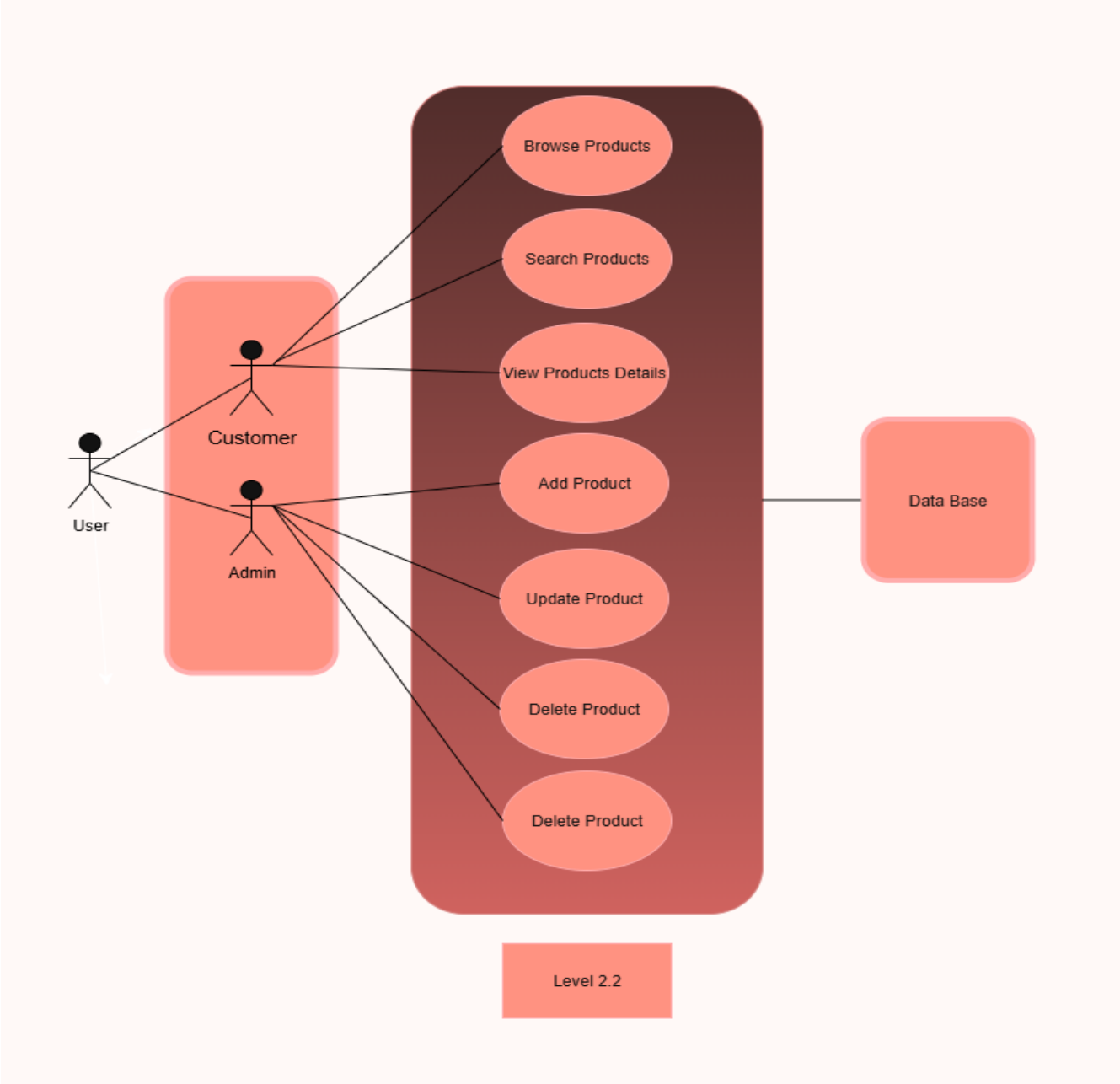

Figure 4: Level 2.2 of Online Bookstore System (Product Management)

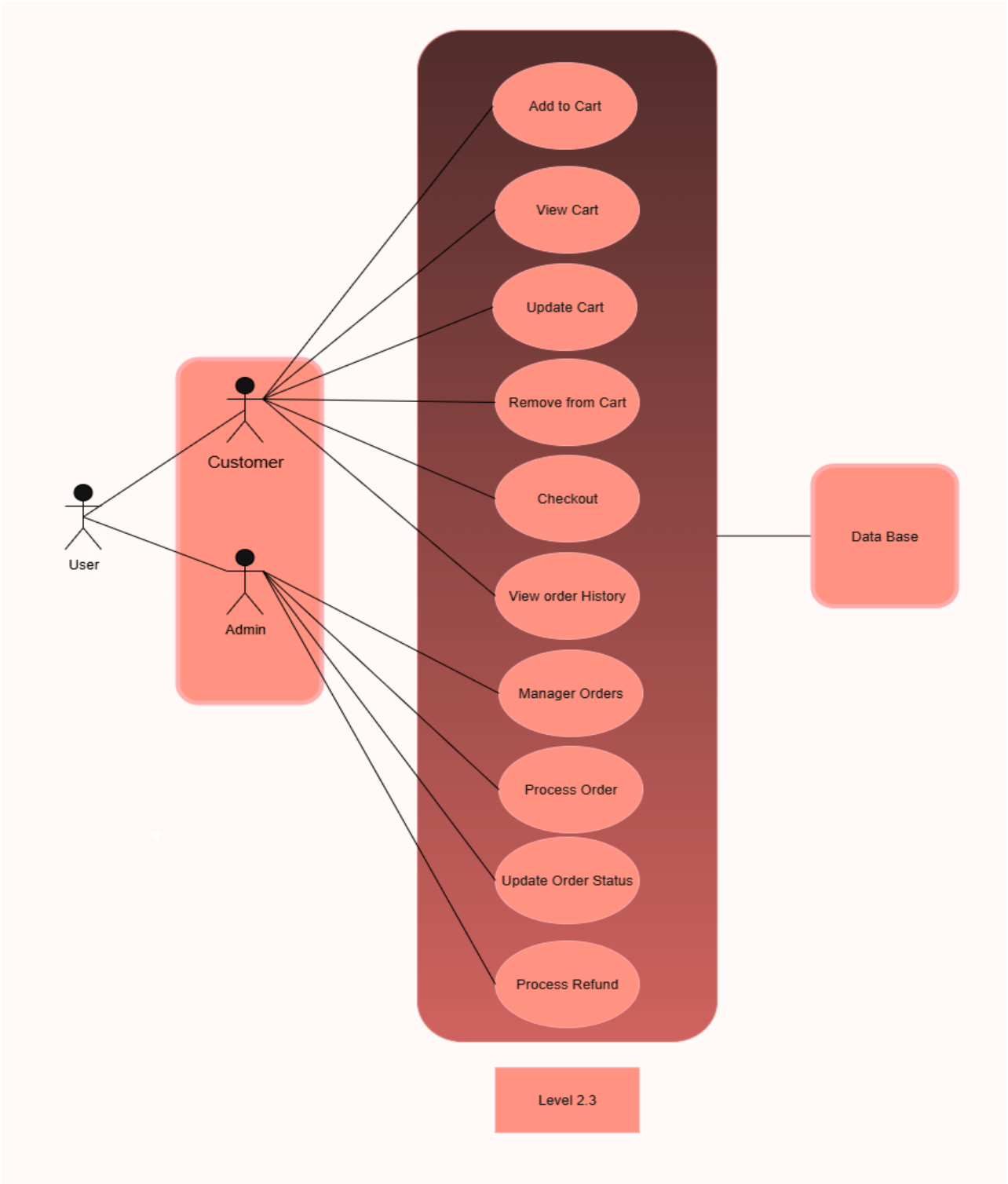

Figure 5: Level 2.3 of Online Bookstore System (Order Management)

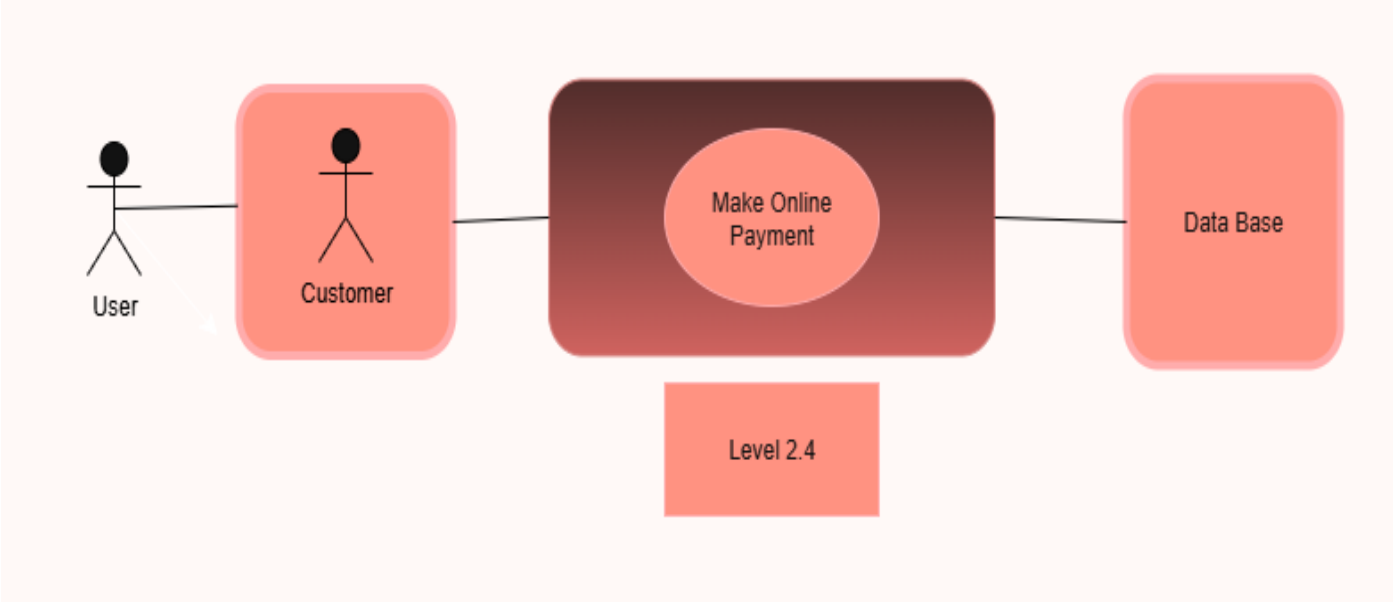

Figure 6: Level 2.4 of Online Bookstore System (Payment)

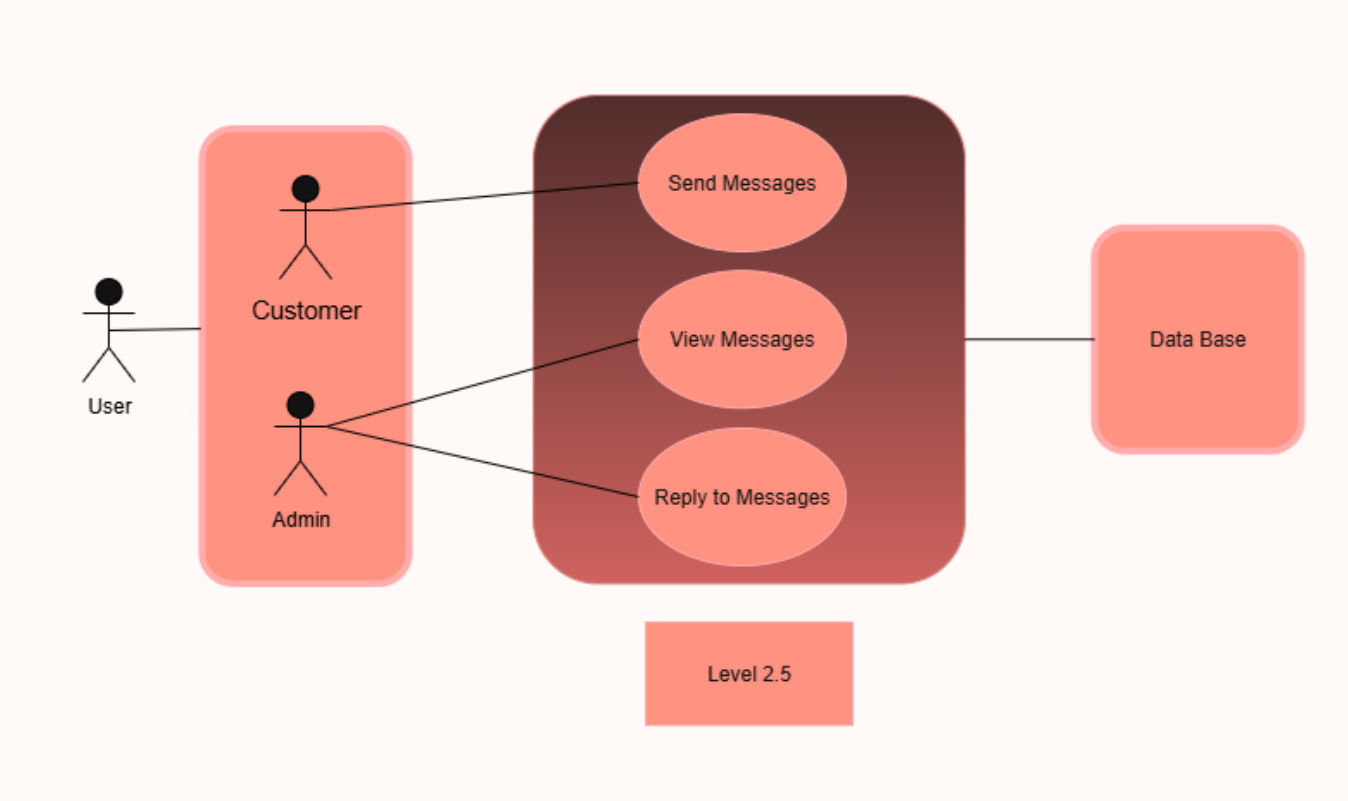

Figure 7: Level 2.5 of Online Bookstore System (Contact/Support)

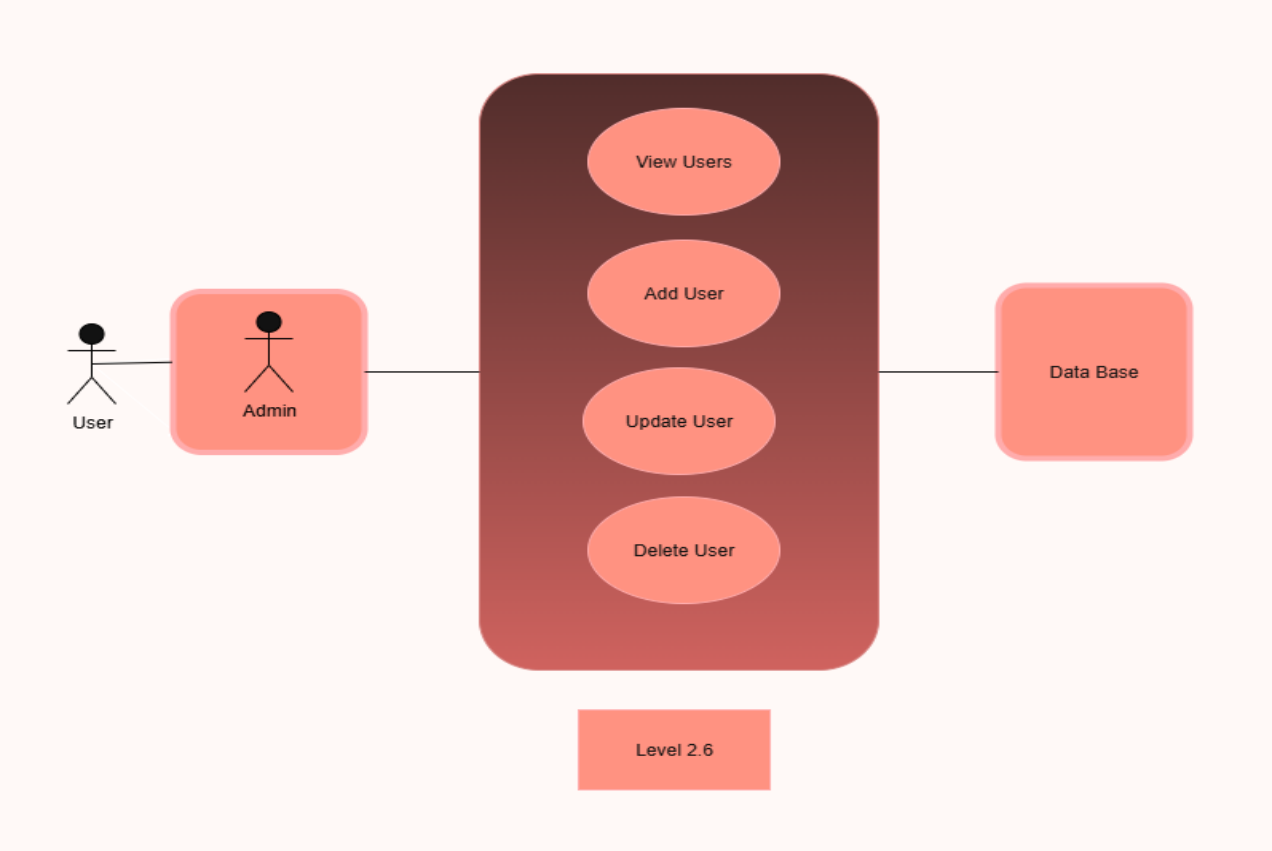

Figure 8: Level 2.5 of Online Bookstore System (User Management - Admin)

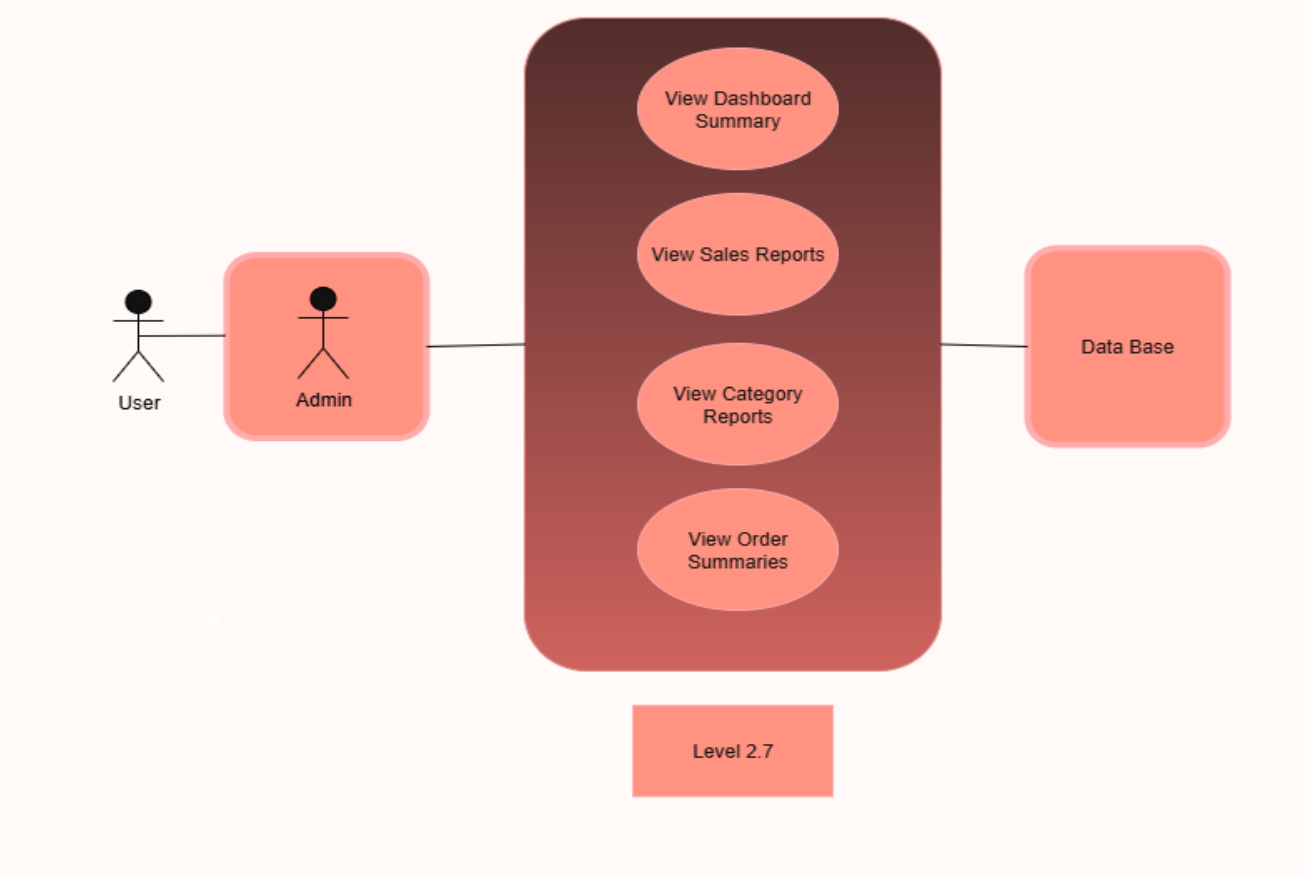

Figure 9: Level 2.5 of Online Bookstore System (Admin Dashboard/Reporting)

4.4 Activity Diagram and Swimlane Diagram of generated Use Cases:

Use case 1: Sign Up

Activity Diagram:

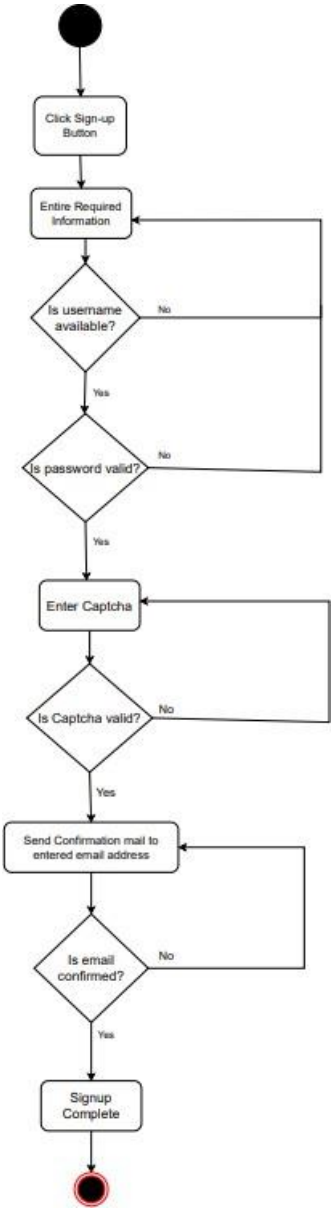

Figure 10: Activity Diagram for Sign-up

Use Case 2: Sign In
Activity Diagram

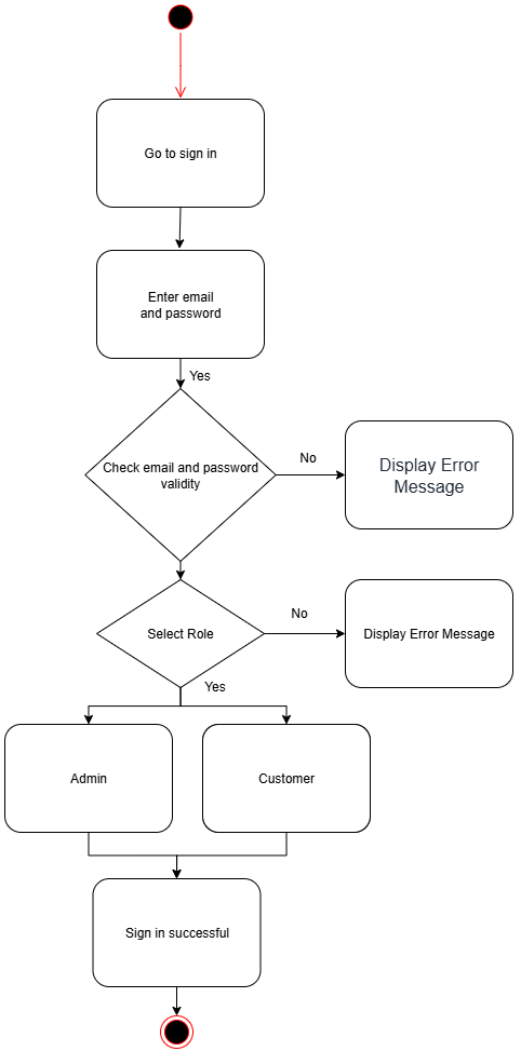

Figure 11: Activity Diagram for Sign-in

Use Case 3: Sign out

Activity Diagram

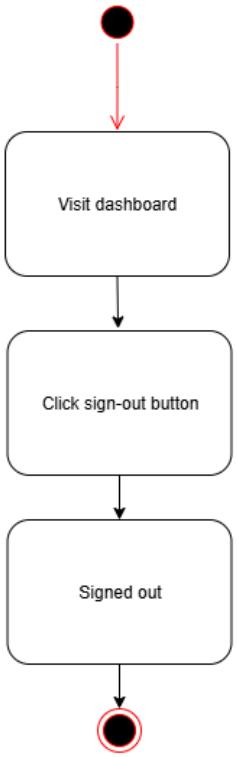

Figure 12: Activity Diagram for Sign-Out

Use Case 4: User Dashboard
Activity Diagram

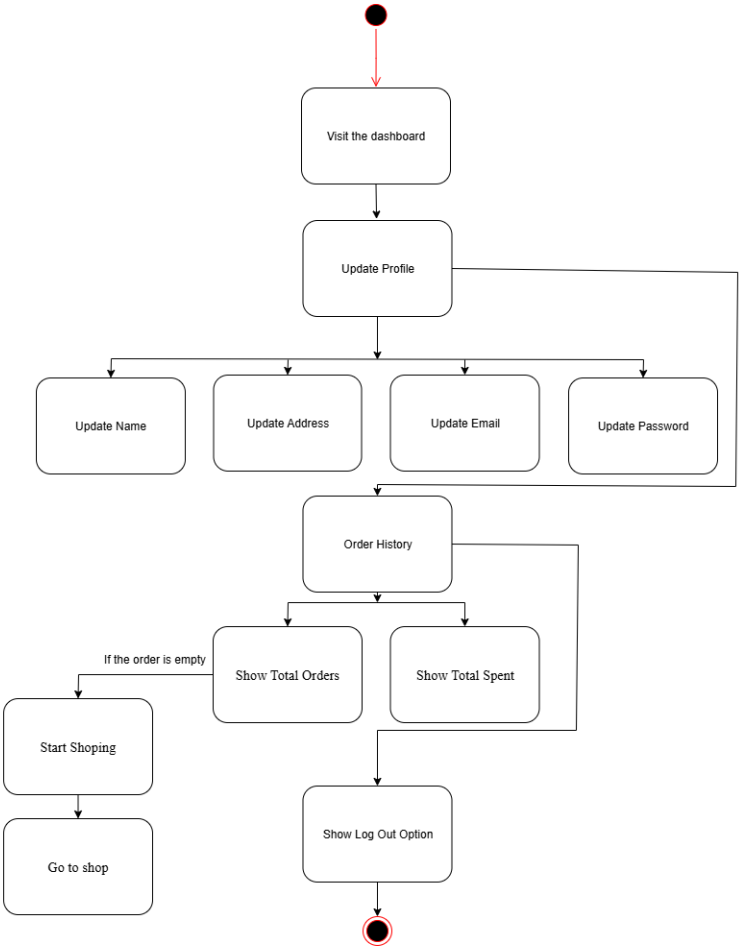

Figure 13: Activity Diagram for User Dashboard

Use Case 5: Admin Dashboard
Activity Diagram

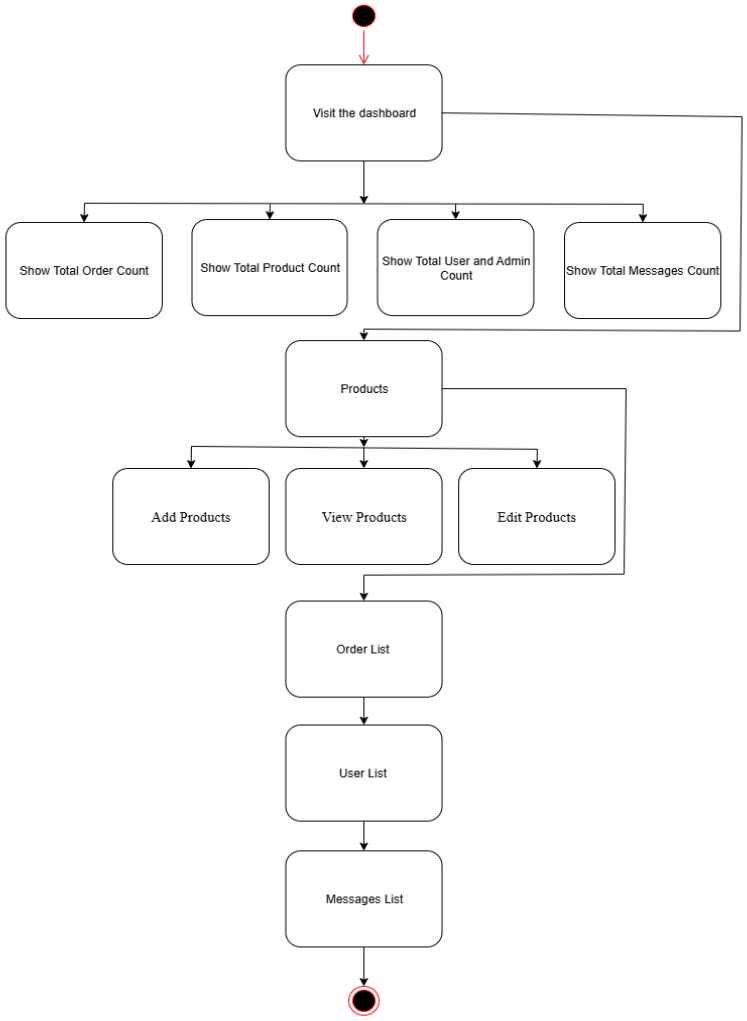

Figure 14: Activity Diagram for Admin Dashboard

Use Case 6: Add to cart

Activity Diagram

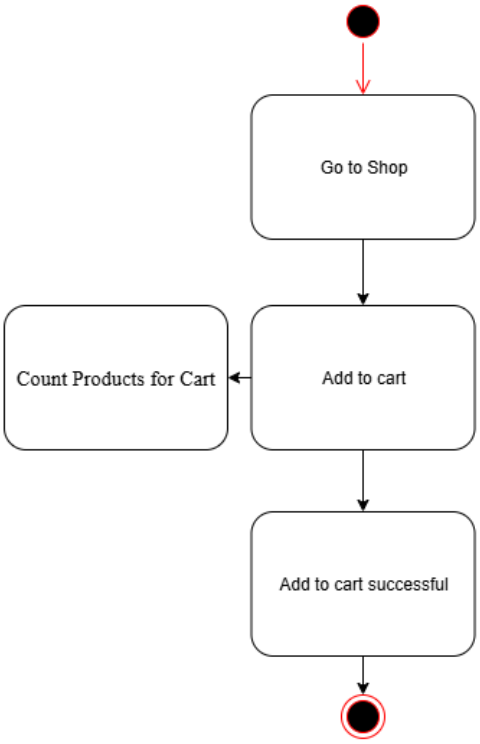

Figure 15: Activity Diagram for Add to Cart

Use Case 7: Check Out
Activity Diagram

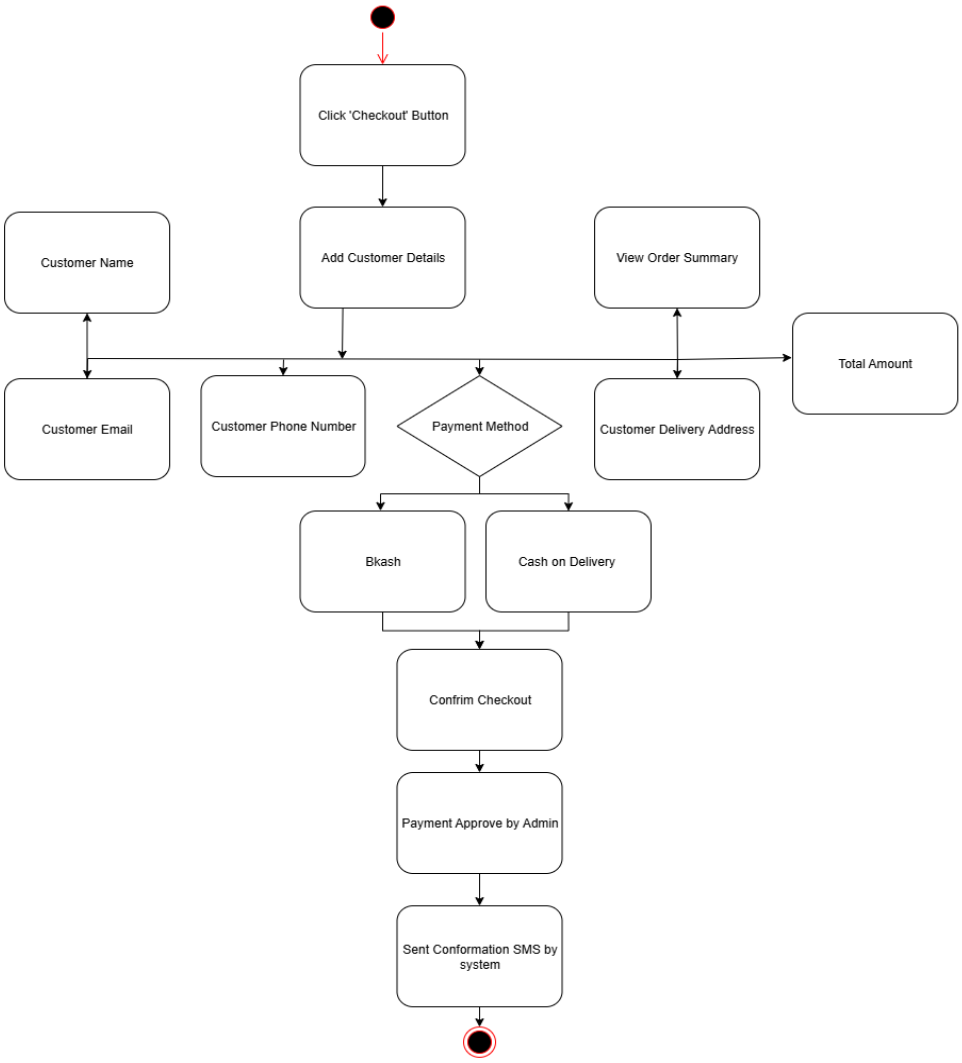

Figure 16: Activity Diagram for Checkout

Use Case 8: Products Cart Process

Activity diagram

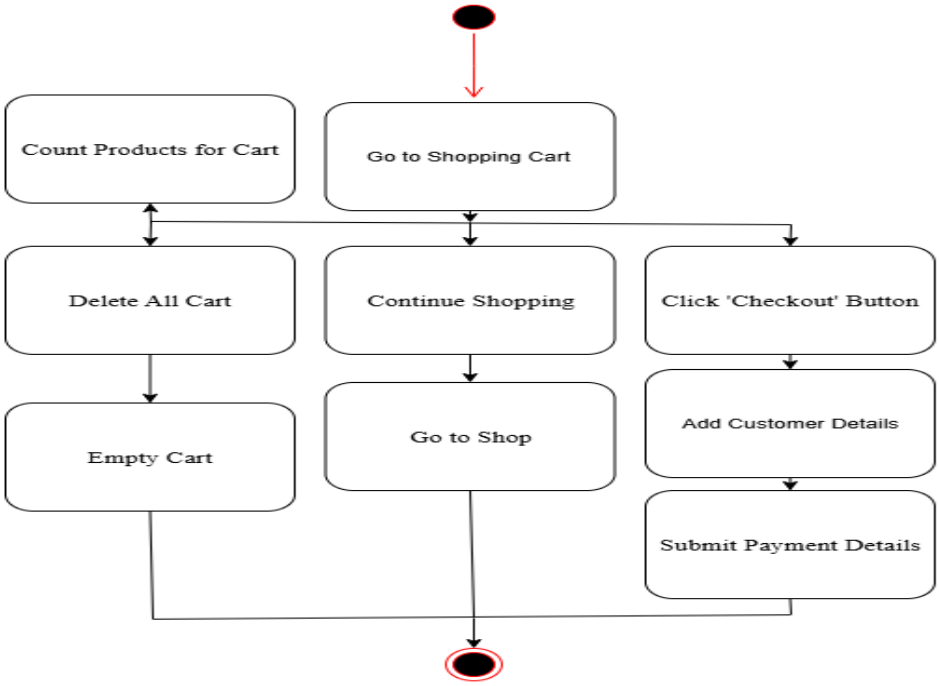

Figure 17: Activity Diagram for Products Cart Process

Use Case 9: Display Messages (Admin)

Activity diagram

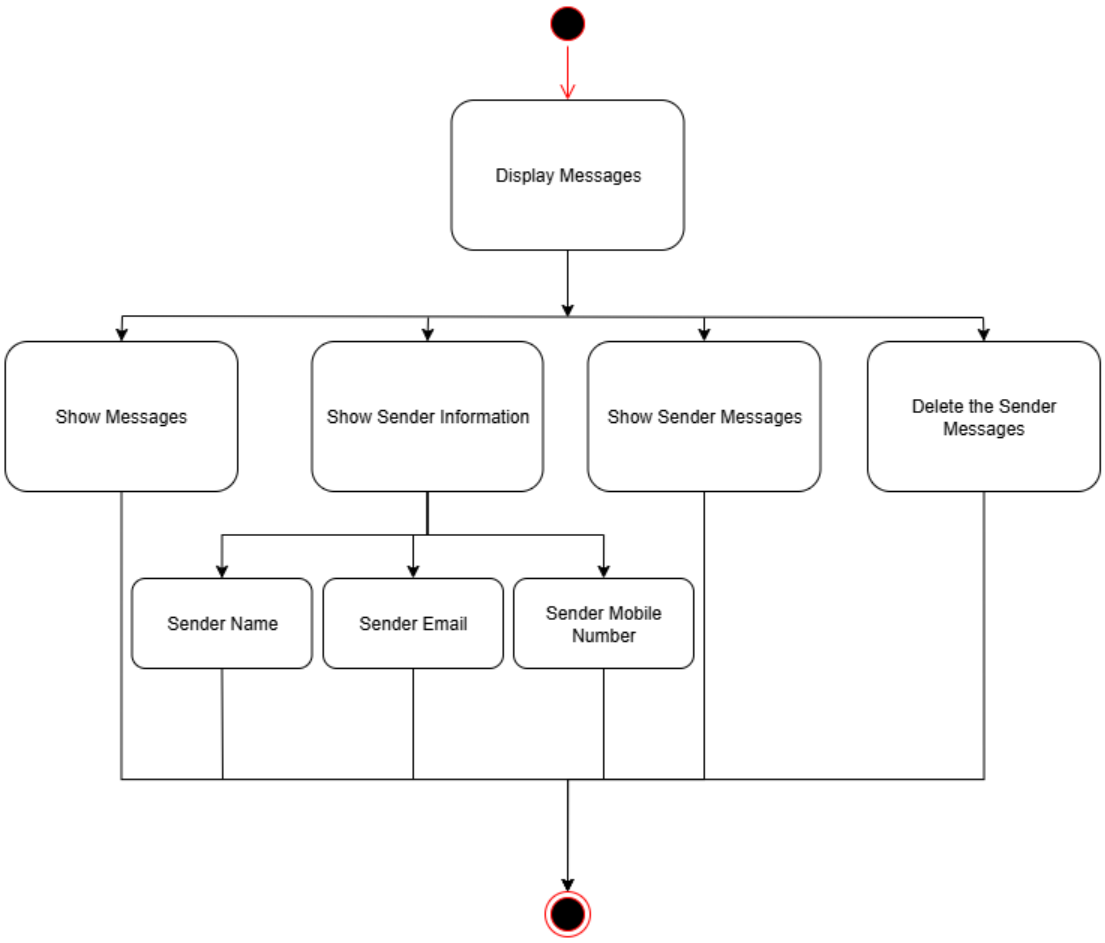

Figure 18: Activity Diagram for Display Messages (Admin)

Use Case 10: View Order List (Admin)

Activity diagram for Contact:

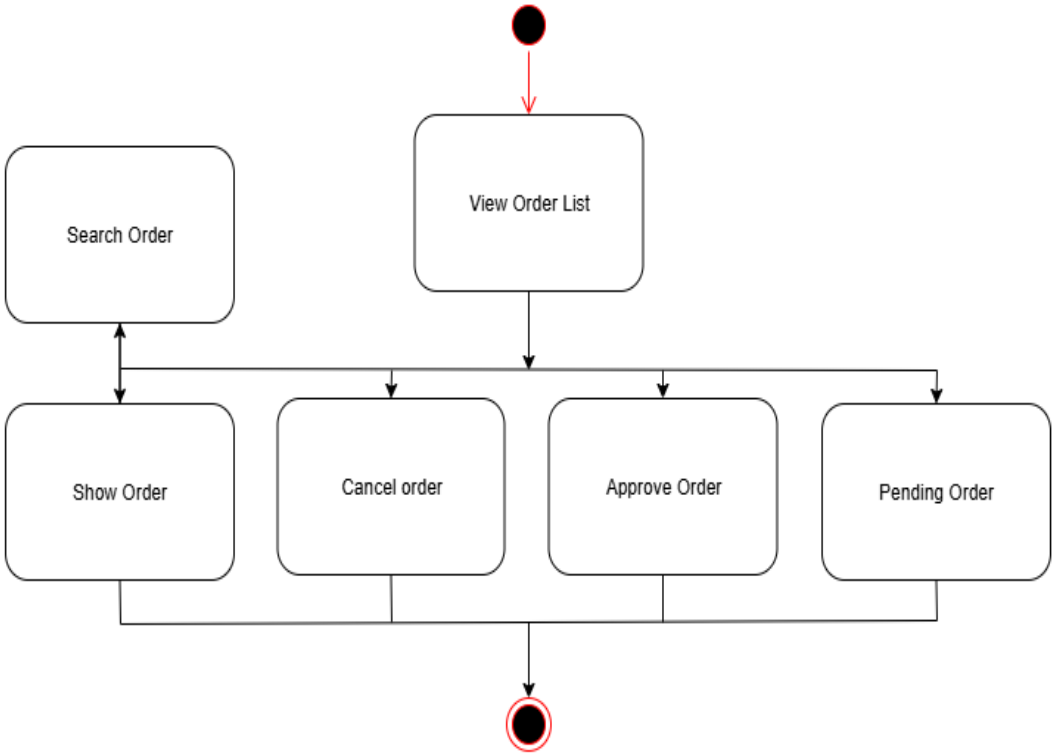

Figure 19: Activity Diagram for View Order List (Admin)

Use Case 11: Display User List (Admin)

Activity diagram

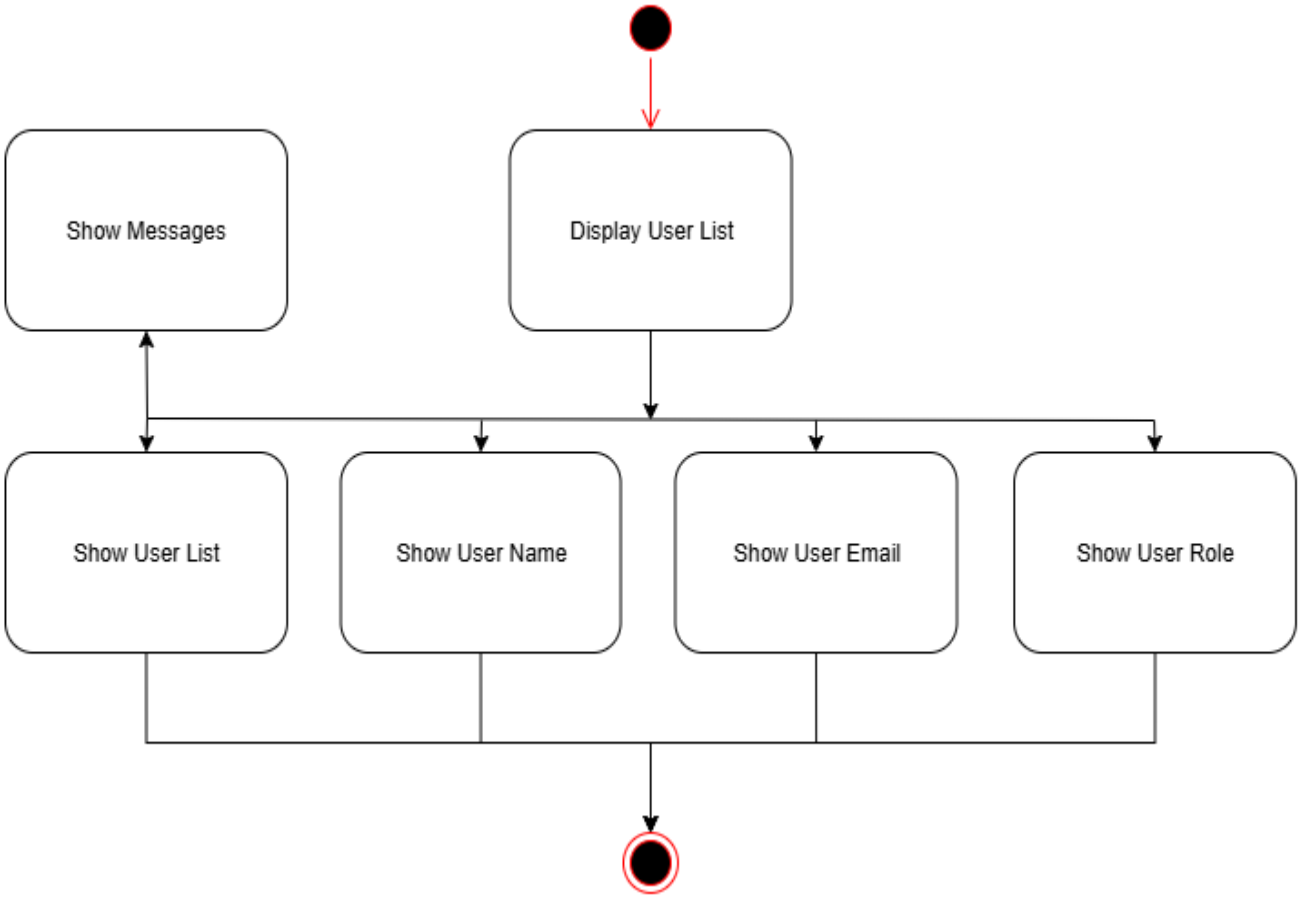

Figure 20: Activity Diagram for Display User List (Admin)

Use Case 11: Edit Items

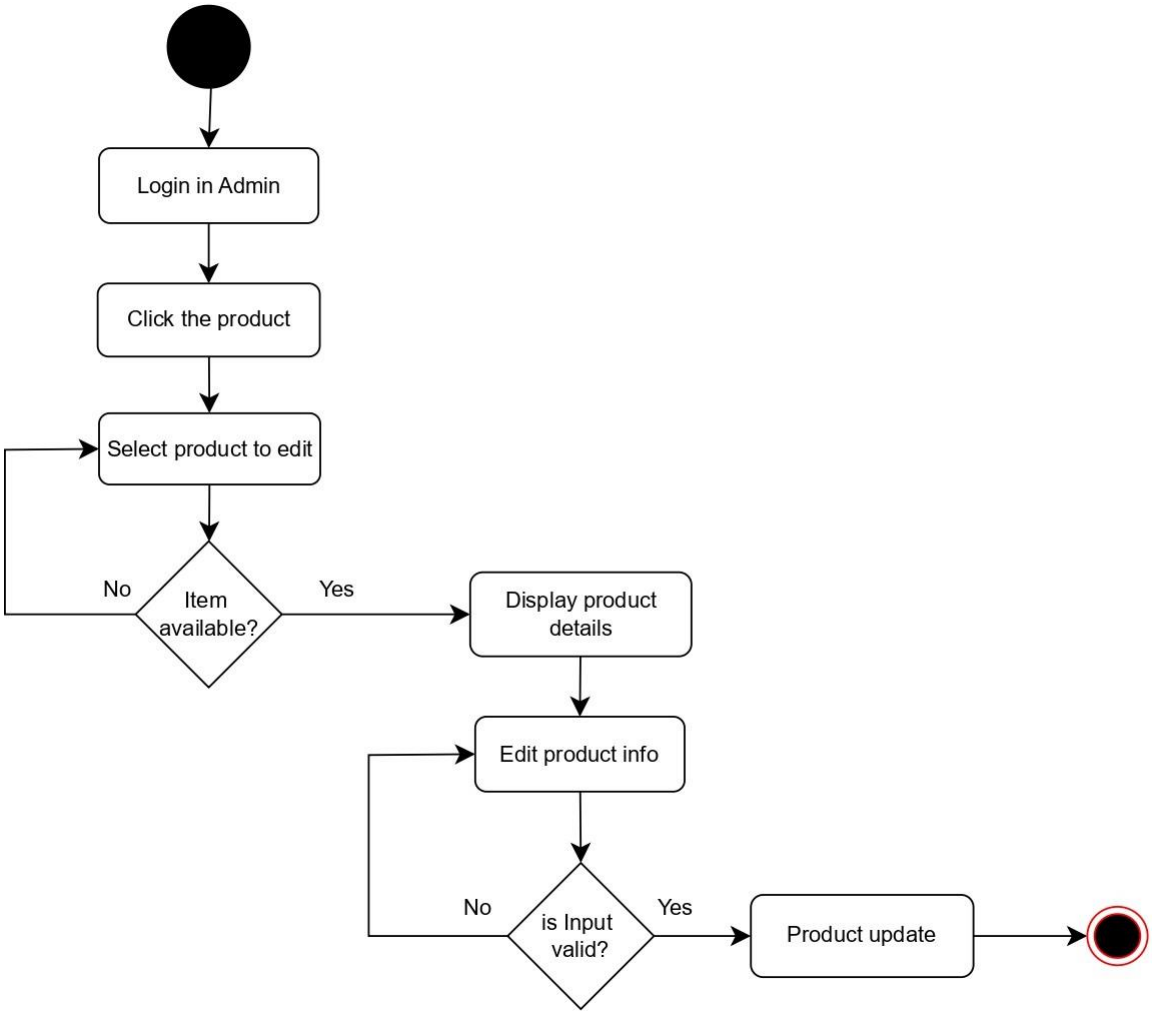

Figure 21: Activity Diagram for Edit Items

Chapter 5: Class Based Modelling:

5.1 Class Based Modeling Concept

Class-based modelling depicts the objects that the system will work with, the actions that will be taken on them, the connections between the objects, and the cooperation between the defined classes.

5.2 Attribute Selection:

Table 1: Attribute Selection of Classes

No.	Potential Class	Noun
1.	Account	• Verification code • User
2.	User	• Id • First name • Last name • E-mail • Phone no • Address
3.	Product	• Id • Name • Quantity • Size • Color • Price • Discount • Insufficient quantity • Rate • Advertisement
4.	Order	• Quantity • Product • Price • Customer
5.	Payment	• Payment method • Payment info

6.	Admin	• Address • Message • Status (available or not)
7	Customer	• Type • Amount paid online • Total amount paid • Message
8.	Browse	• User • Product • Date • Keyword • Price

9.	Notification	• Id • Receiver • Type
10.	Review	• reviewId • productId • userId • Rating • Comment
11.	System	• systemLogId • description • eventType • timestamp
12.	Delivery Service	• deliveryId • orderId • courierName • trackingNumber • estimateDate • deliveryFee
14.	Social Media	• socialMediaID • platform • accountURL
15.	Chat Support	• ticketId • userId • adminId • status • message

		• createdAt
16.	Database	• recordId • tableName • operation • oldData • NewData

17.	Activity	• activityId • userId • action • details • timestamp
18.	Catagory	• catagoryId • name • description • isActive

5.3 Method Tables for Identification:

Table 2: Methods of Class

No.	Class	Methods
1.	Account	• signUp() • login() • signOut() • lockAccount() • sendVerificationCode() • recoverPassword() • verifyEmail() • verifyUser() • manageForgotPassword() • changePassword() • updateProfile()
2.	User	• setCustomerID() • getCustomerID() • setFirstName() • getFirstName() • setLastName() • getLastName() • setAddress() • getAddress() • setEmail() • getEmail() • setPhoneNo() • getPhoneNo() • getPurchaseHistory() • setPaymentMethod()

3.	Payment	• processPayment() • refundPayment() • setPaymentGateway() • getPaymentStatus() • payViaCreditCard() • payViaBKash() • storePaymentInfo() • storeCashOnDeliveryinfo()
4.	Admin	• approveRequest() • chat() • addProduct() • removeProduct() • editProduct() • advertiseProduct() • manageUserRoles() • generateReports() • manageDiscount() • editProfile() • manageDeliveryService() • manageSocialMedia()

5.	Customer	• setCustomerID() • getCustomerID() • setFirstName() • getFirstName() • setLastName() • getLastName() • setEmail() • getEmail() • setPhoneNo() • getPhoneNo() • setAddress() • getAddress() • receiveMesseges() • chat() • orderProduct() • purchase() • editProfile() • comment() • rateProduct() • shareWithFriends()

13.	ChatSupport	• initiateChat() • sendMessage() • attachFile() • escalateToAdmin() • closeTicket()
14.	Database	• addInfo() • updateInfo() • removeInfo() • storeInfo()
15.	Activity	• storeAuthenticationInfo() • storeRatingAndReview() • StoreProductModificationInfo() • storeAdvertisementRecord() • storeOrderInfo()

16.	Catagory	• createCategory() • renameCategory() • deleteCategory() • addProductToCategory() • removeProductFromCategory() • getCategoryHierarchy() • listAllCategories() • filterProductsByCategory()

5.4 Finalizing Classes:

To determine the final classes, it was necessary to examine the potential for hierarchies, class mergers, and the inclusion of additional attributes, methods, or classes. The following insights were gathered:

- The system includes two distinct types of users. While it might seem appropriate to define a general User class as a parent to both Admin and Customer classes, the significant differences in their attributes and methods make a shared parent class unnecessary. Thus, separate Admin and Customer classes should be maintained without inheriting from a common User class.

5.5 Class Cards:

Table 3: Class Card of User

Attributes	Methods
- id - name - email - password (hashed) - user_type ('user' or 'admin')	- register() - login() - logout() - updateProfile() - viewOrderHistory() - addToCart() - sendMessage()
Responsibilities	Collaborators
Store personal and credential information	Database
Manage user session (login/logout)	Account
Manage a personal shopping cart	Cart
Place and view past orders	Order
Send inquiries to administration	Message

Table 4: Class Card of Admin

Attributes	Methods
- id - name - email - password (hashed) - user_type ('user' or 'admin')	- addProduct() - editProduct() - deleteProduct() - viewAllOrders() - updateOrderStatus() - viewAllUsers() - deleteUser() - viewMessages() - viewDashboard()
Responsibilities	Collaborators
Manage product catalog	Product
Oversee and update customer orders	Order
Manage the user base	User
Respond to customer inquiries	Message
Monitor store performance	Dashboard

Table 5: Class Card of Product

Attributes	Methods
- id - name - details - price - image	- getDetails() - getPrice() - updateStock() - displayProductCard()
Responsibilities	Collaborators
Represent a single book for sale	Database
Store descriptive and pricing data	Admin
Be managed by an administrator	Cart
Be placed in a shopping cart	Cart

Table 6: Class Card of Cart

Attributes	Methods
- user_id - list_of_cart_items	- addItem(product_id, quantity) - removeItem(product_id) - updateQuantity(product_id, new_quantity) - calculateSubtotal() - clearCart() - viewItems()
Responsibilities	Collaborators
Hold items a user intends to purchase	CartItem
Belong to a single user session	Product
Calculate total cost of contained items	User

Table 7: Class Card of Order

Attributes	Methods
- id - user_id - name - email - address - total_products (summary string) - total_price - placed_on (date) - payment_status	- createFromCart(cart) - updateStatus(new_status) - getOrderDetails() - sendConfirmation()
Responsibilities	Collaborators
Represent a completed purchase transaction	User
Record customer and payment information	Database
Be monitored/managed by an admin	Admin

Table 8: Class Card of Massage

Attributes	Methods
- id - name - user_id - email - number - message (content)	- saveMessage() - markAsRead() - deleteMessage()
Responsibilities	Collaborators
Store an inquiry from the contact form	User
Be read and managed by an admin	Admin
Persist the communication record	Database

6.1 UML Class Diagram

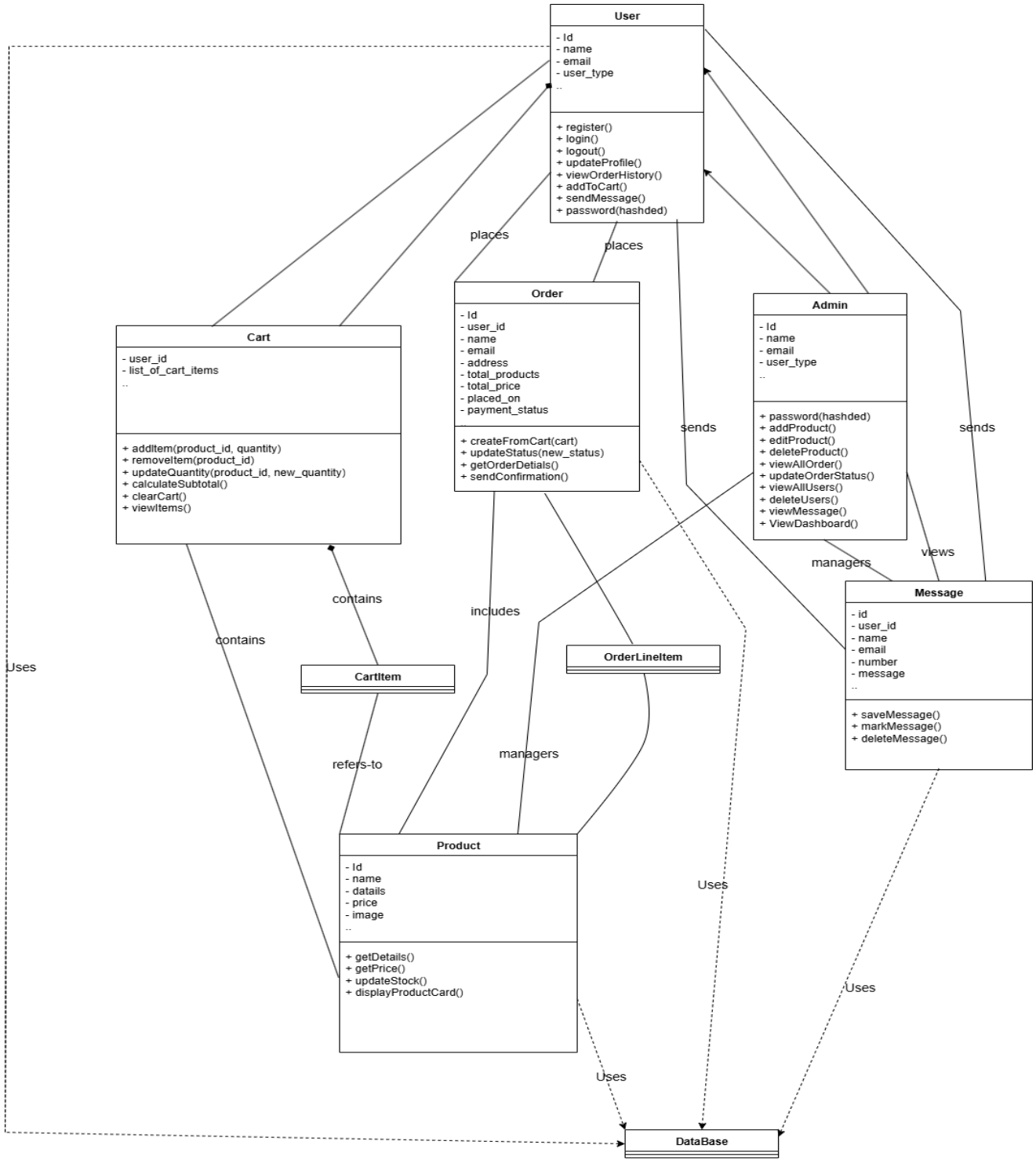

Figure 22: Class Diagram

6.2 Deployment Diagram

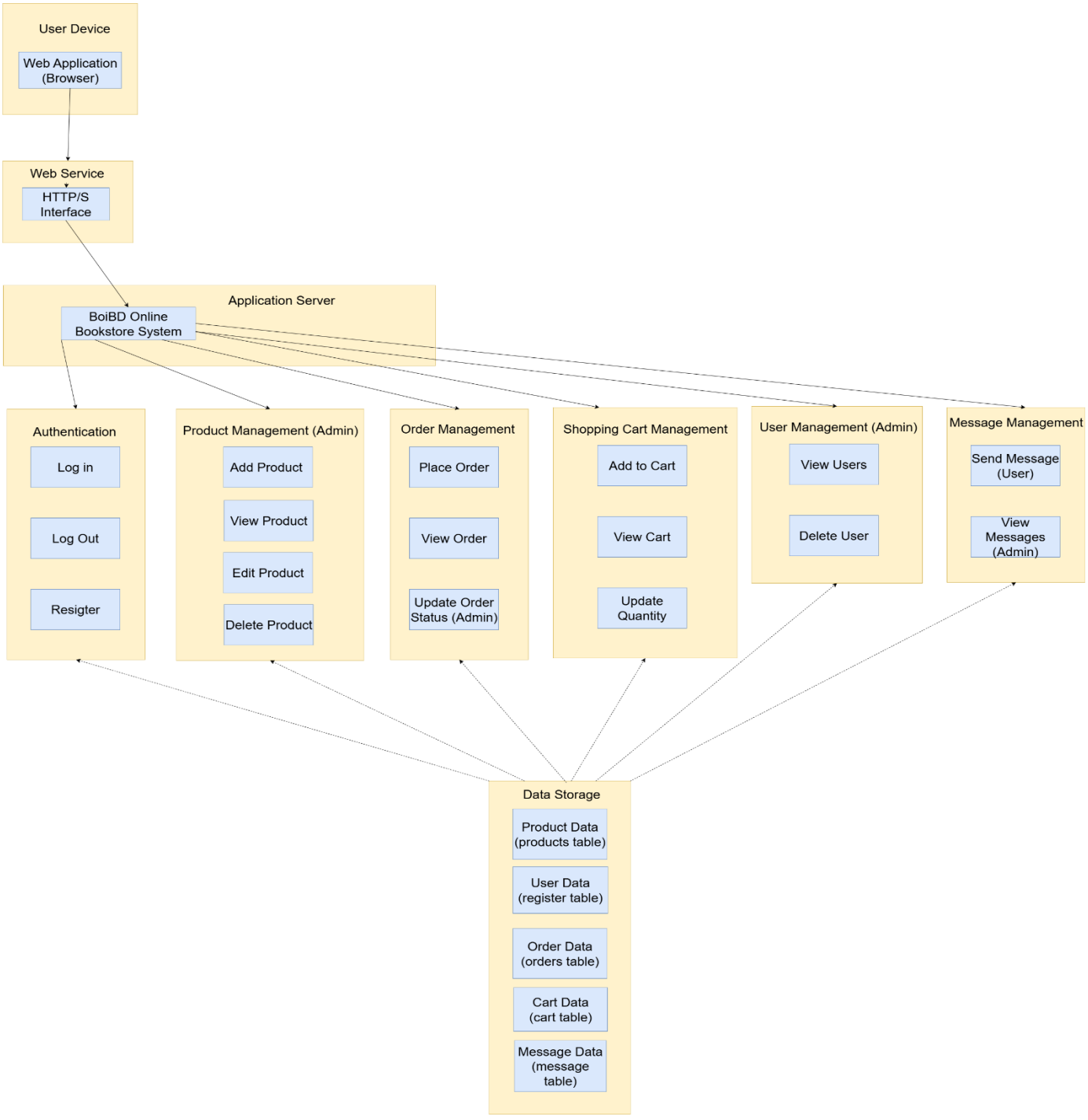

Figure 23: Deployment Diagram

Chapter 7: Flow-Oriented Model

This chapter focus on the flow oriented modelling.

7.1 Introduction

While some software engineers consider data flow-oriented modeling to be outdated, it remains one of the most used techniques for requirements analysis. This approach continues to offer valuable understanding of system behavior and information flow.

7.2 Data Flow Diagram

A Data Flow Diagram (DFD) represents a system from an input-process-output perspective. In such diagrams, data objects are shown as labeled arrows, while processes or transformations are depicted as circles.

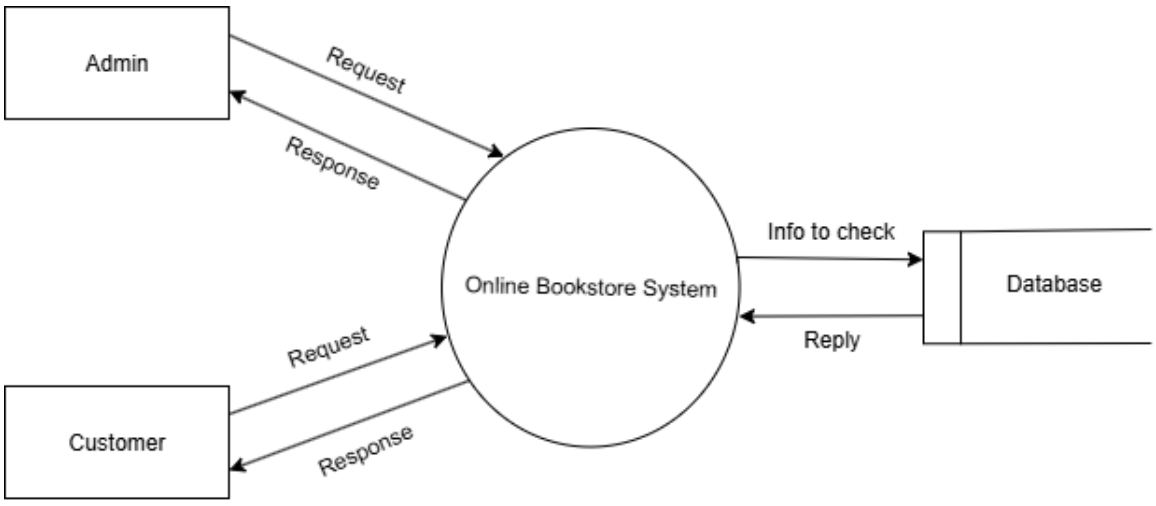

Figure 24: Level 0 of Dataflow diagram for Entire System of Context

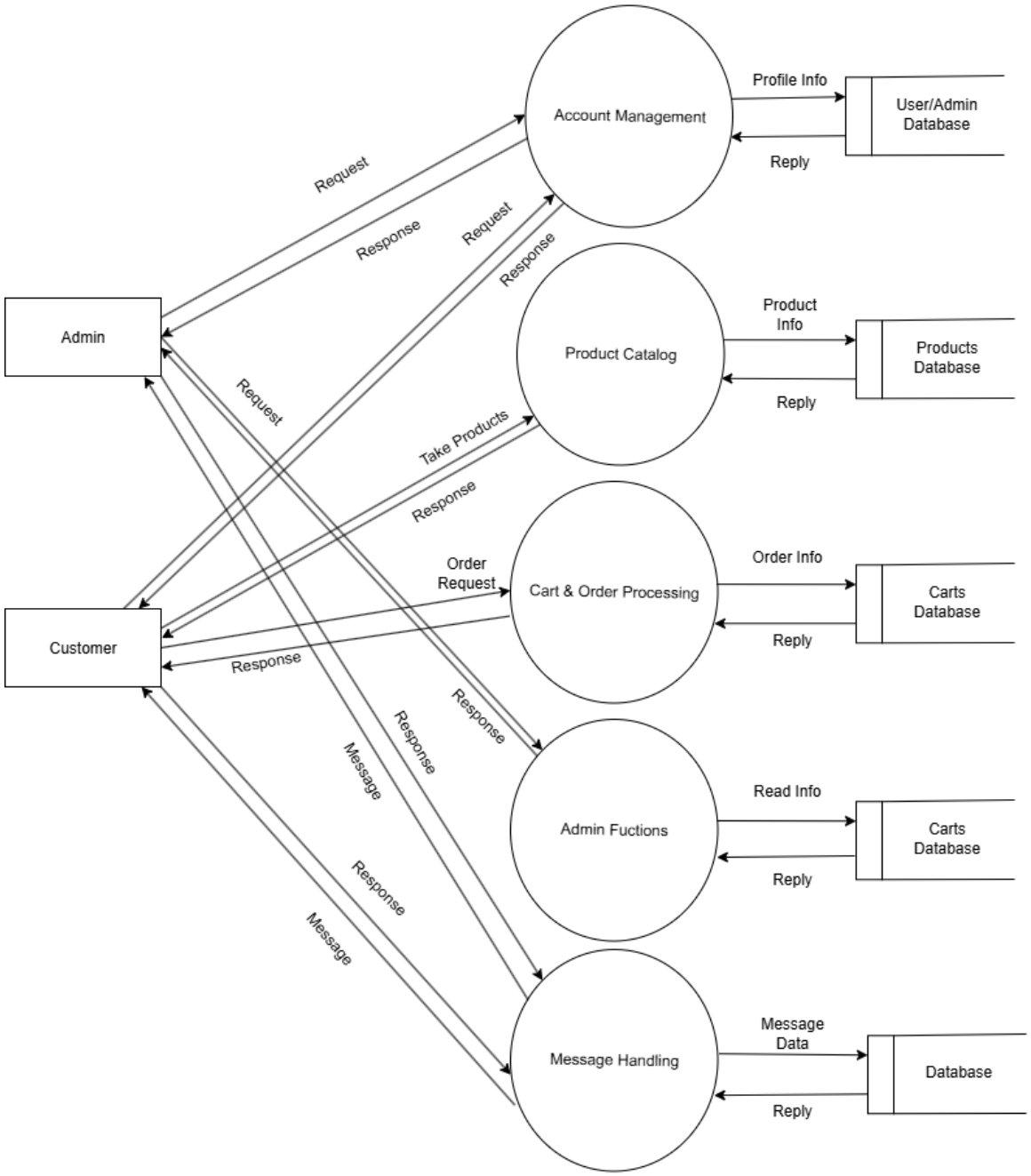

Figure 25: Level 1.1 of Dataflow diagram for Main Processes

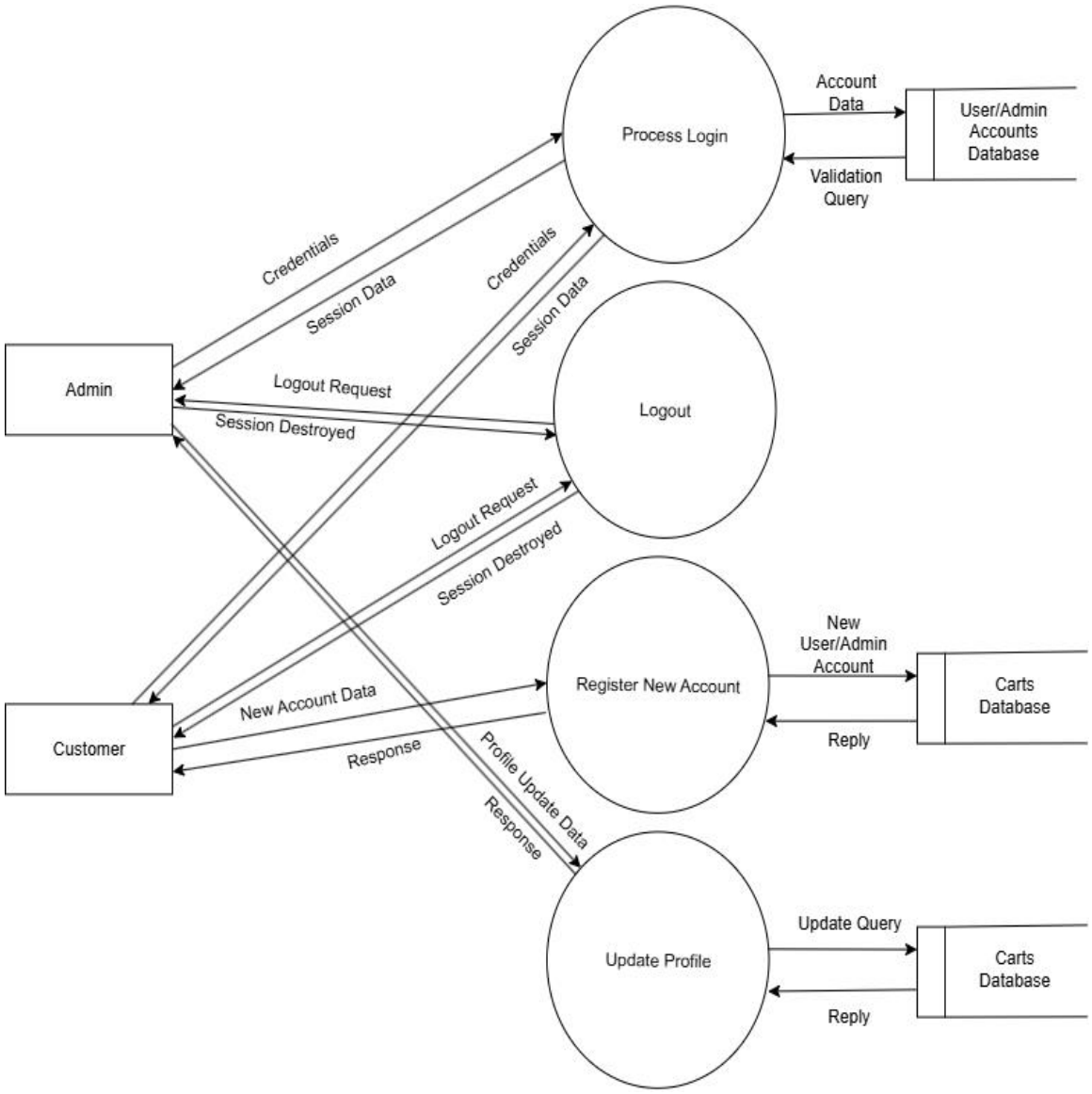

Figure 26: Level 1.2 of Dataflow diagram for Account Management

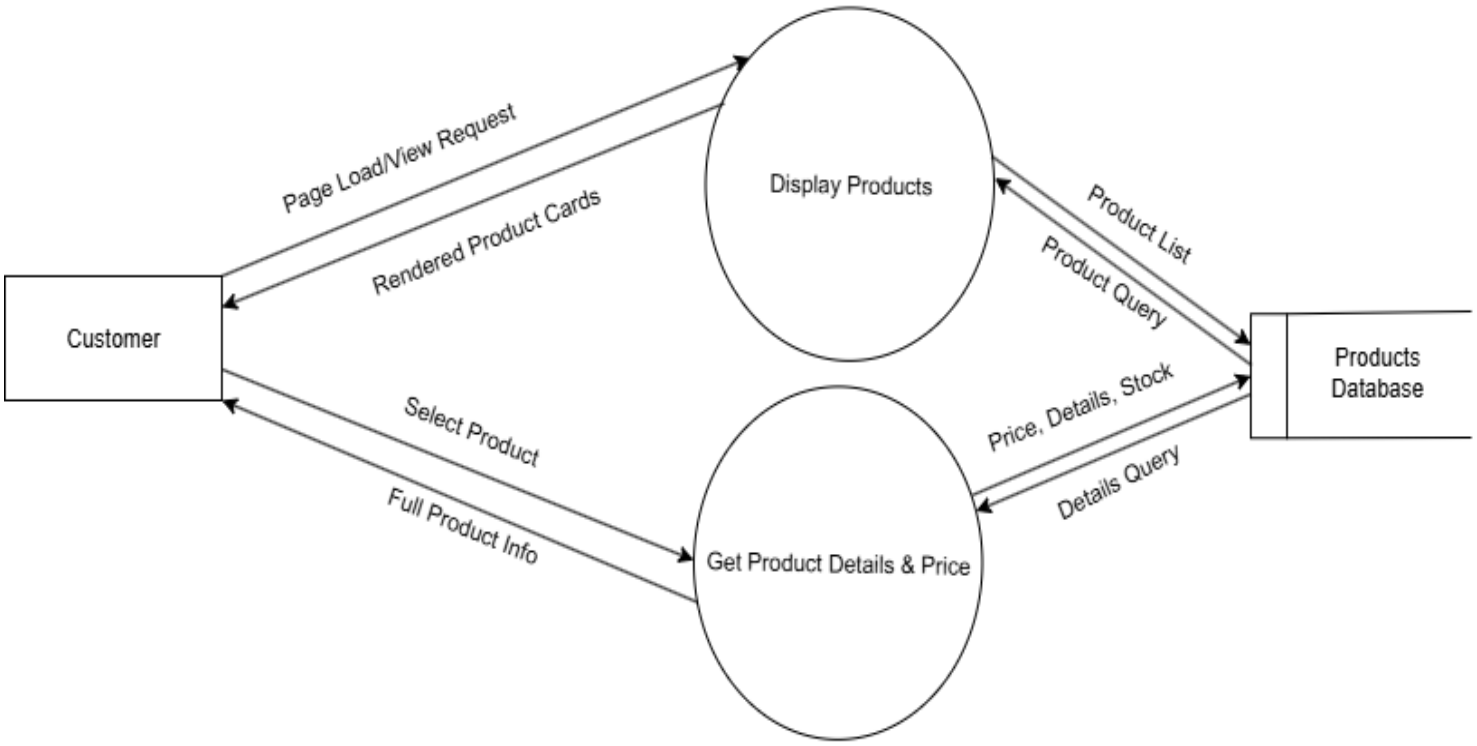

Figure 27: Level 2.1 Data Flow Diagram for Product Listings

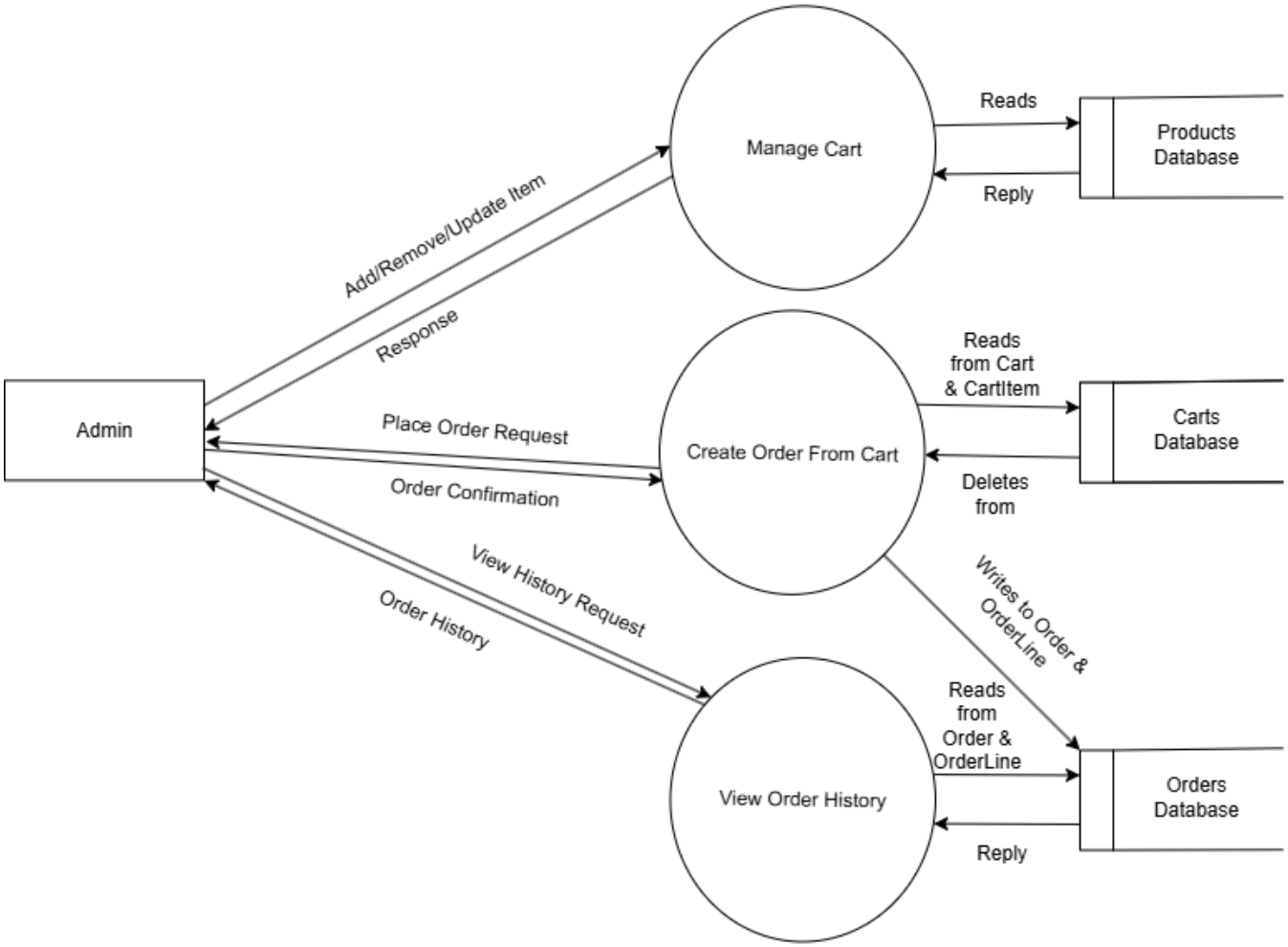

Figure 28: Level 2.2 of Data Flow Diagram for Cart and Order Processing

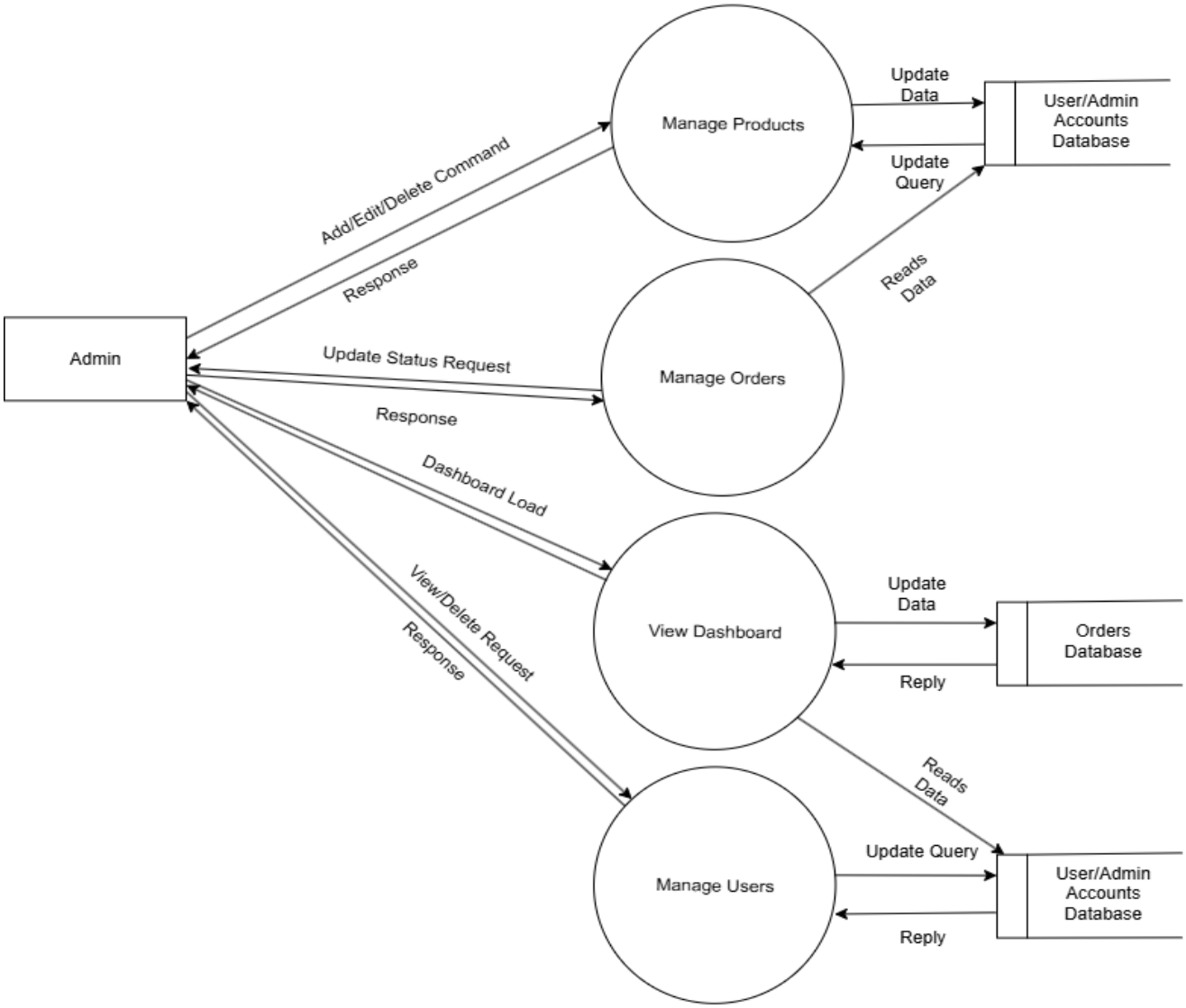

Figure 29: Level 2.3 Data Flow Diagram for Admin Management

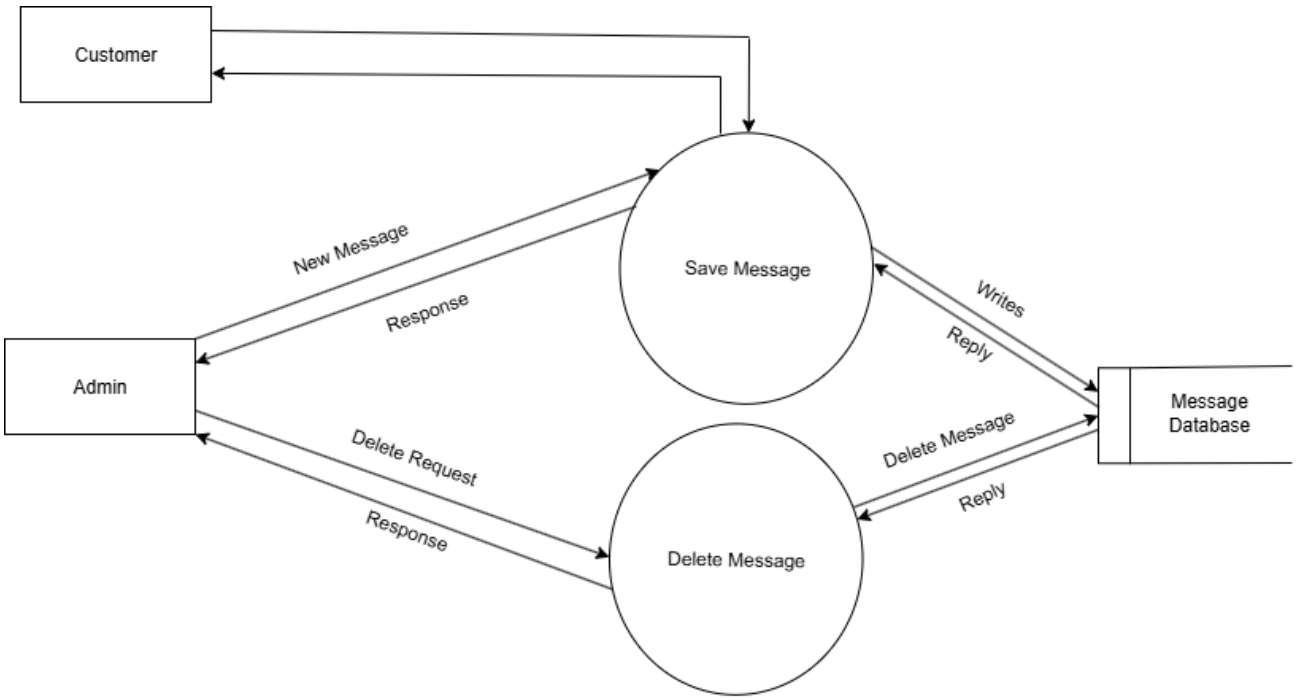

Figure 30: Level 2.4 Data Flow Diagram for Message Handling

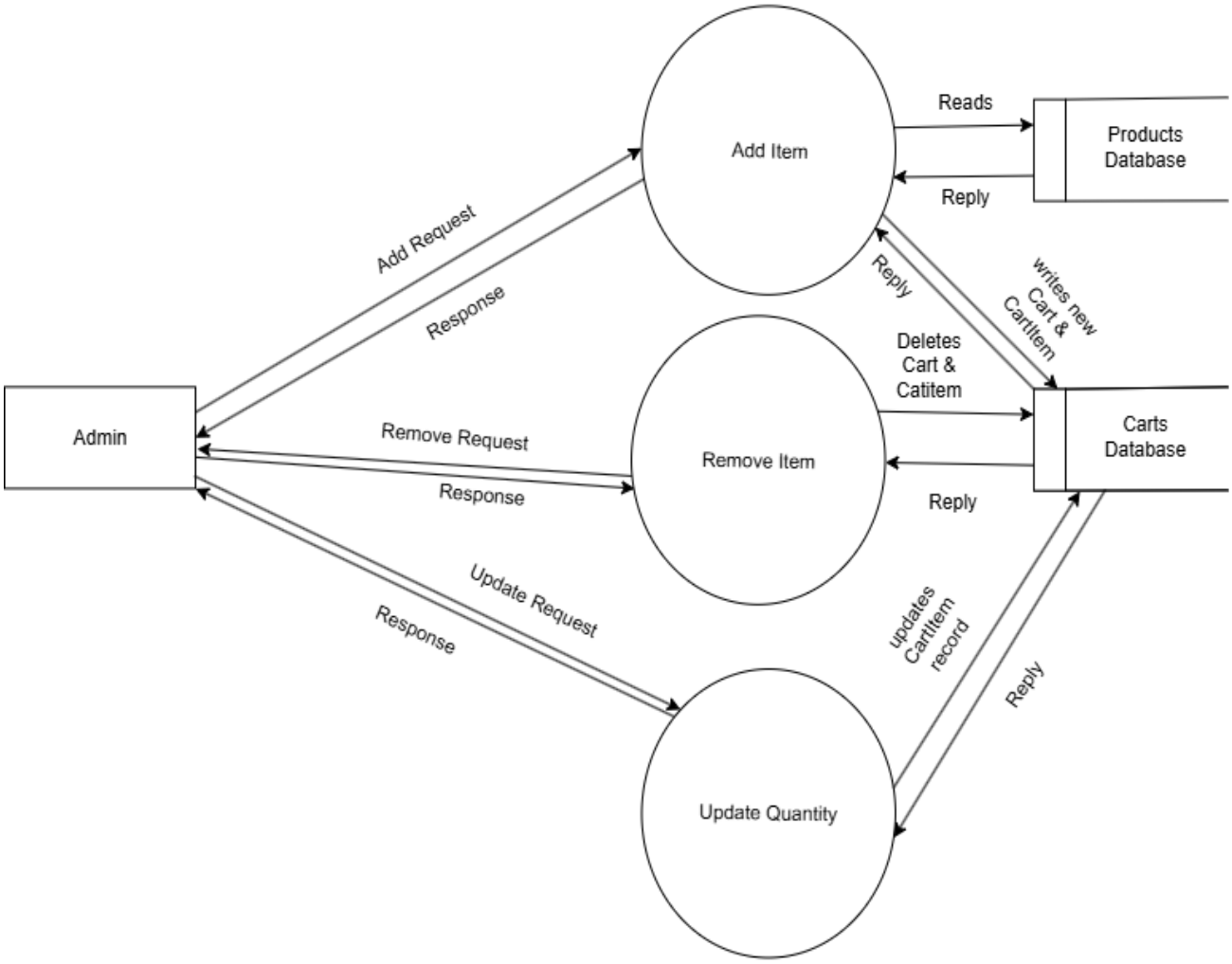

Figure 31: Level 2.5 Data Flow Diagram for Manage Cart

Chapter 8: Behavioral Model

The behavioral model indicates how software will respond to external events.

8.1 State Diagram

State diagram represents active states for each class, the events (**Trigger**s).

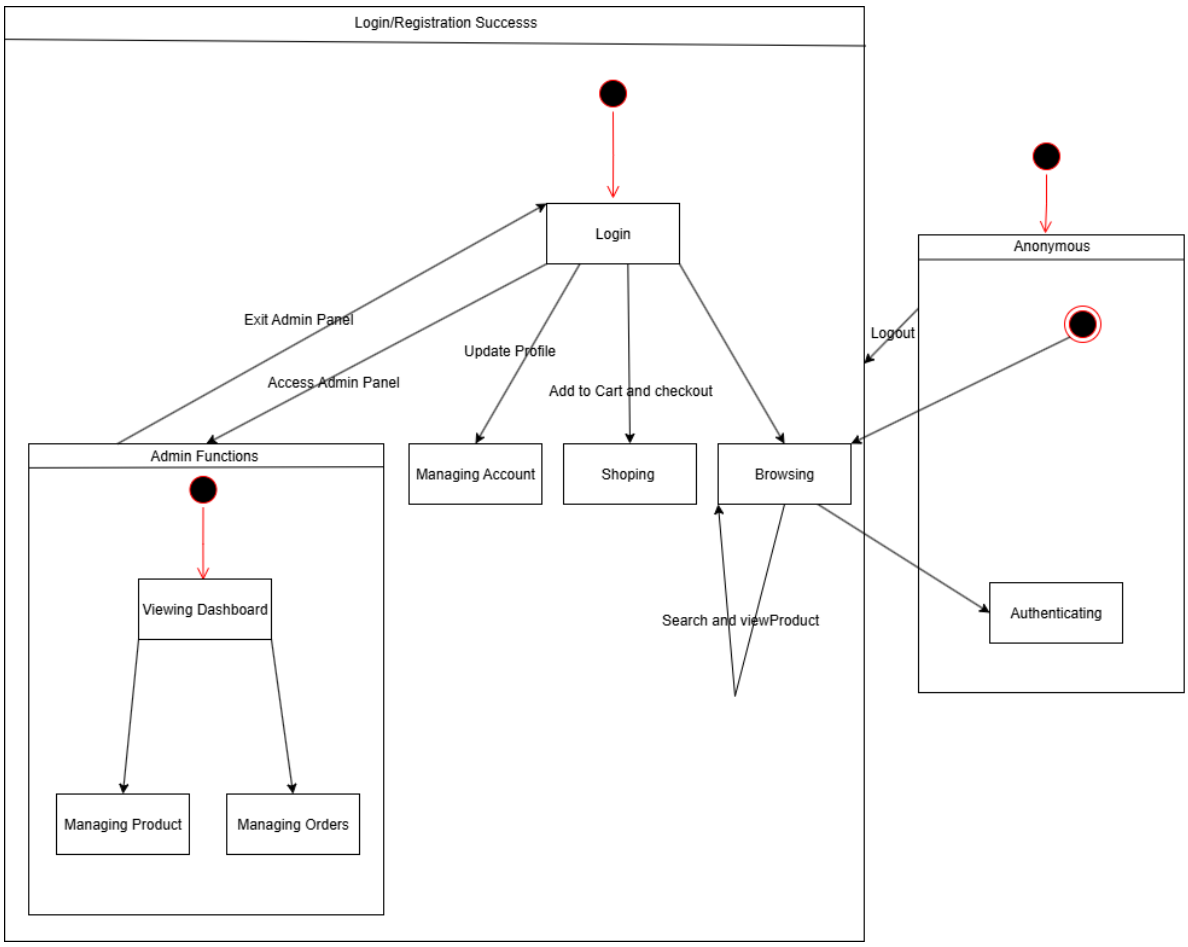

Figure 32: State Diagram for User Session Lifecycle

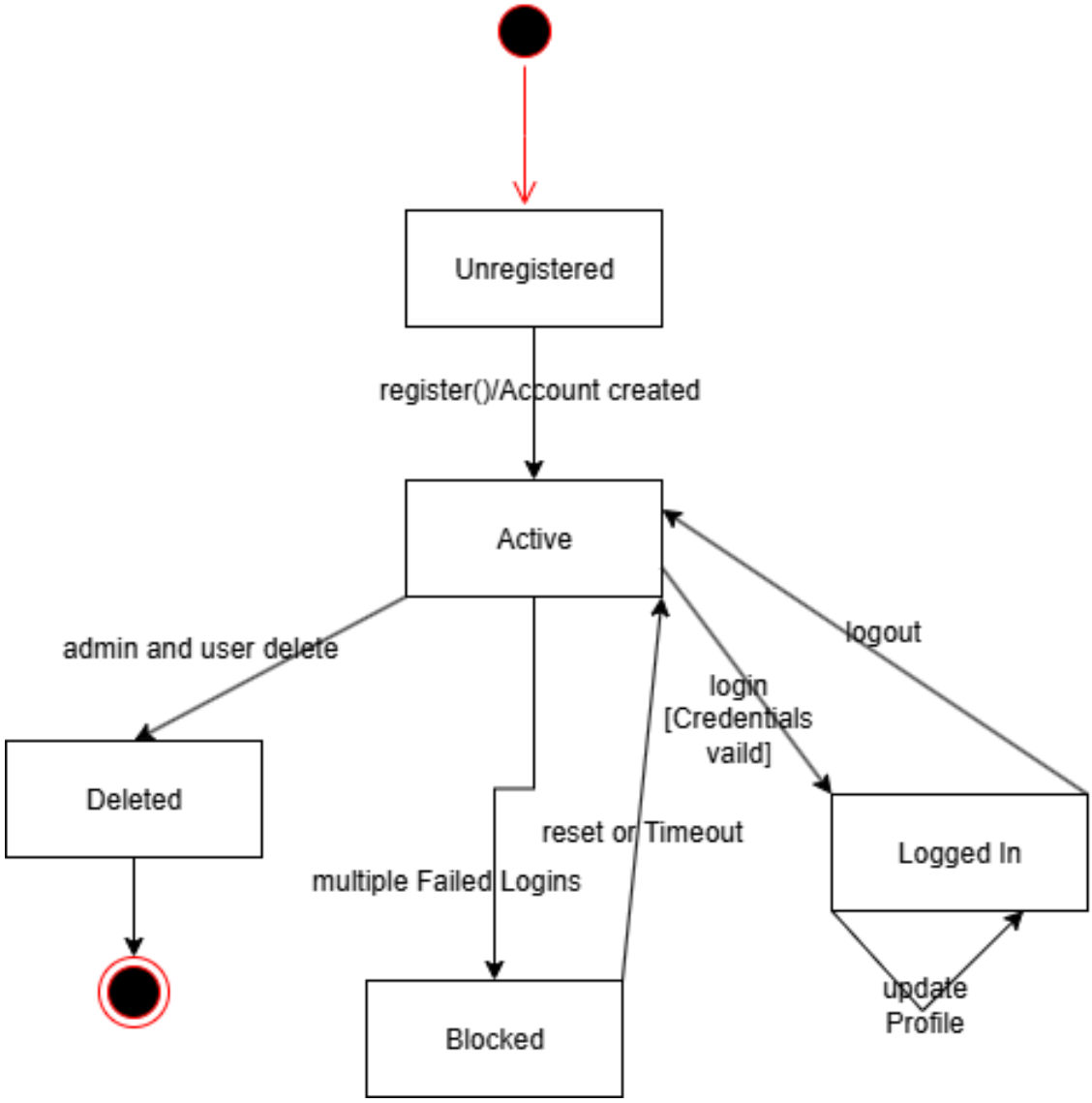

Figure 33: State Diagram Register for Admin or User (Account Object)

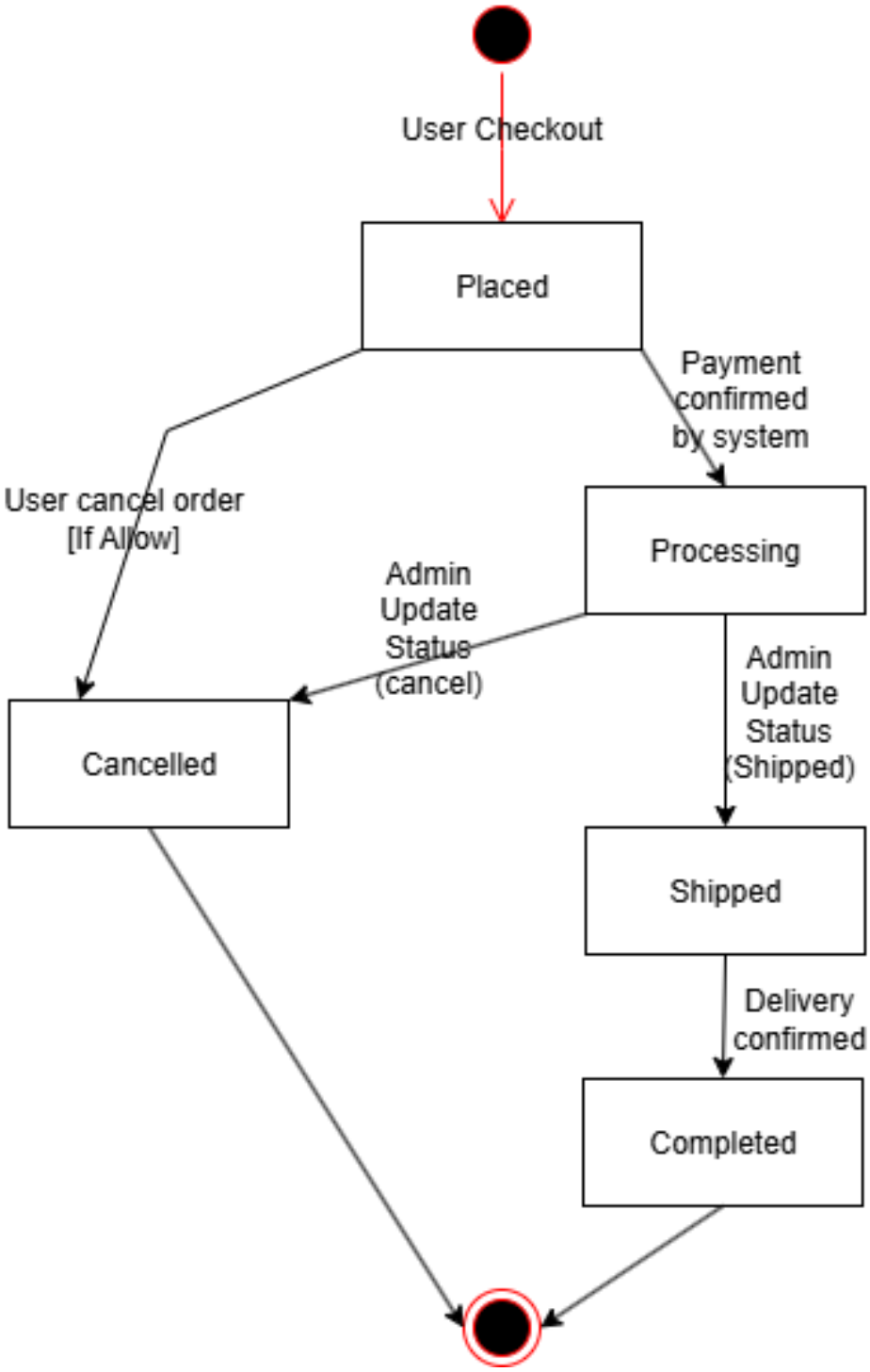

Figure 34: State Diagram for Order based on payment status (Order Object)

8.2 Sequence Diagram

Sequence diagram indicates how events cause transitions from object to object.

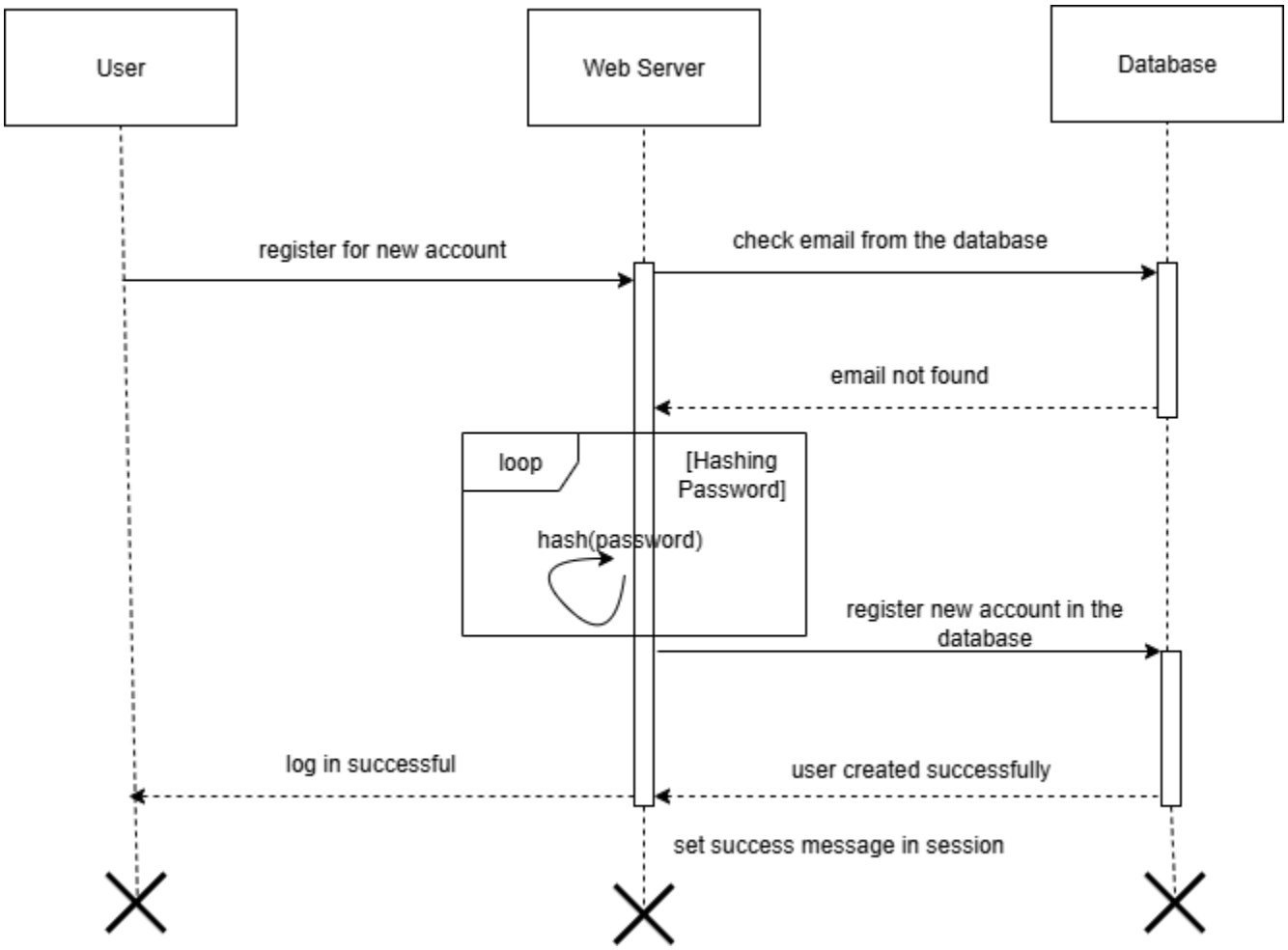

Figure 35: Sequence Diagram for New User Registration

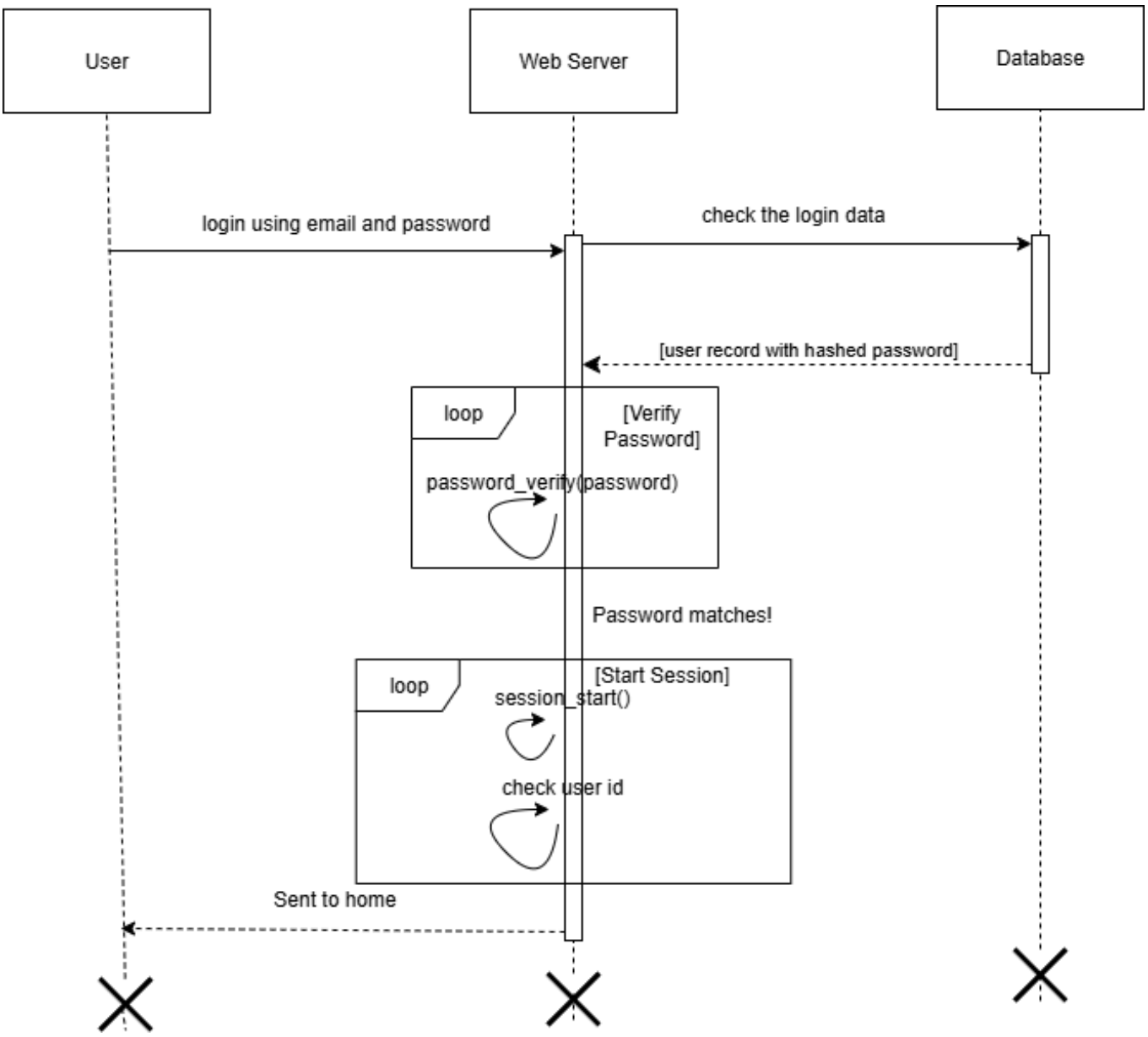

Figure 36: Sequence Diagram for User Login (Success)

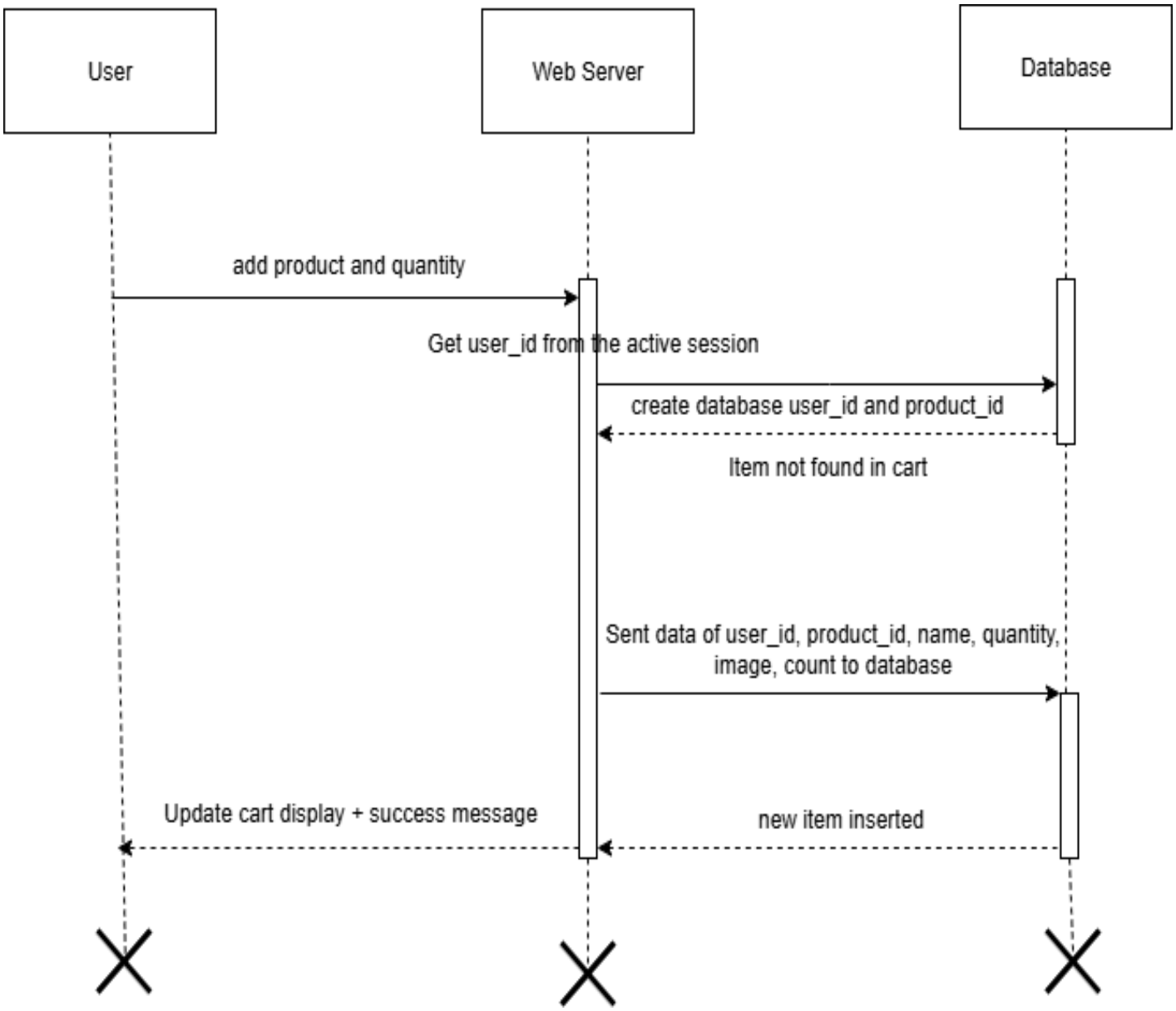

Figure 37: Sequence Diagram for Add a Product to the Cart

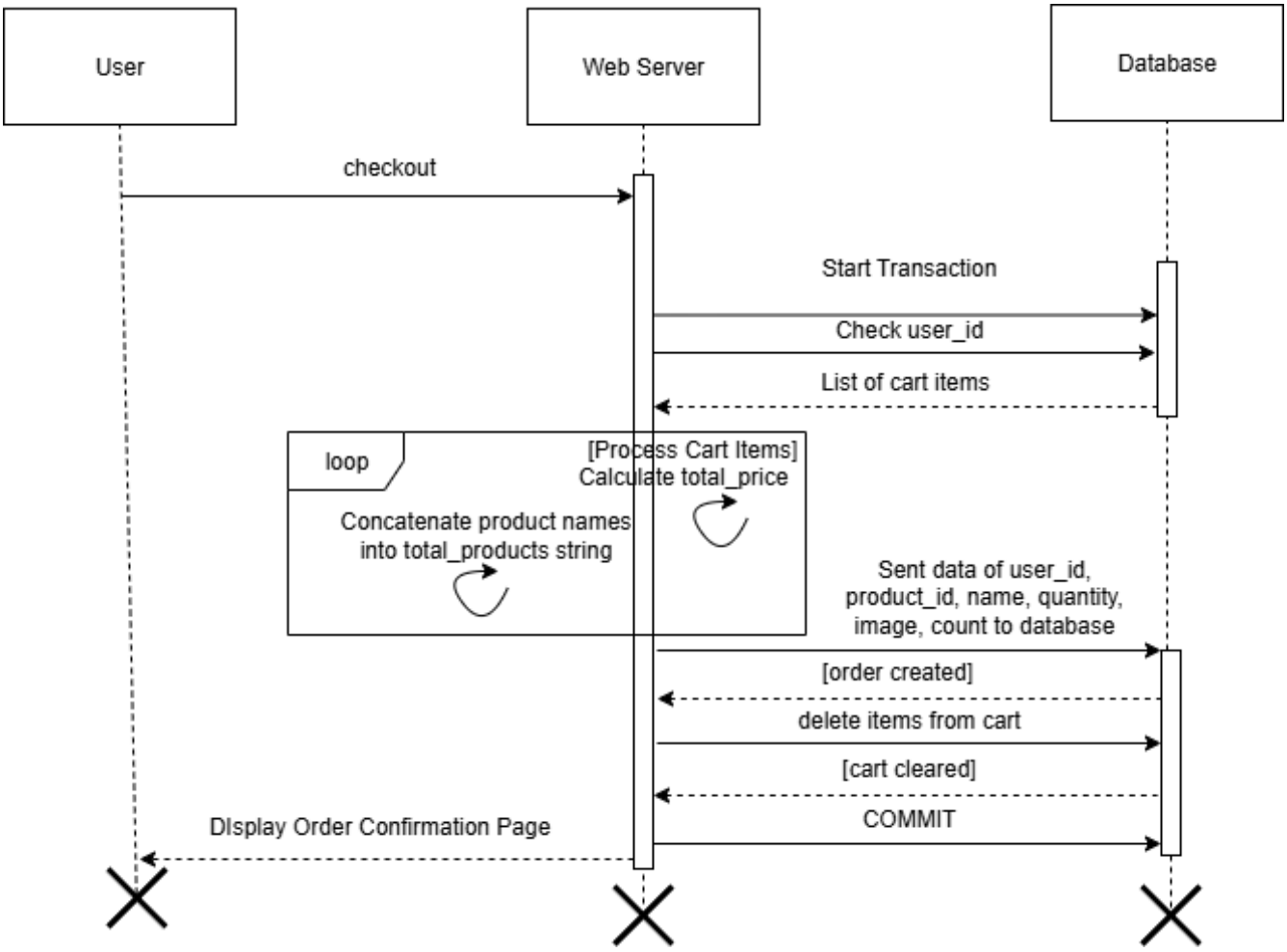

Figure 38: Sequence Diagram for Customer Placing an Order (Checkout)

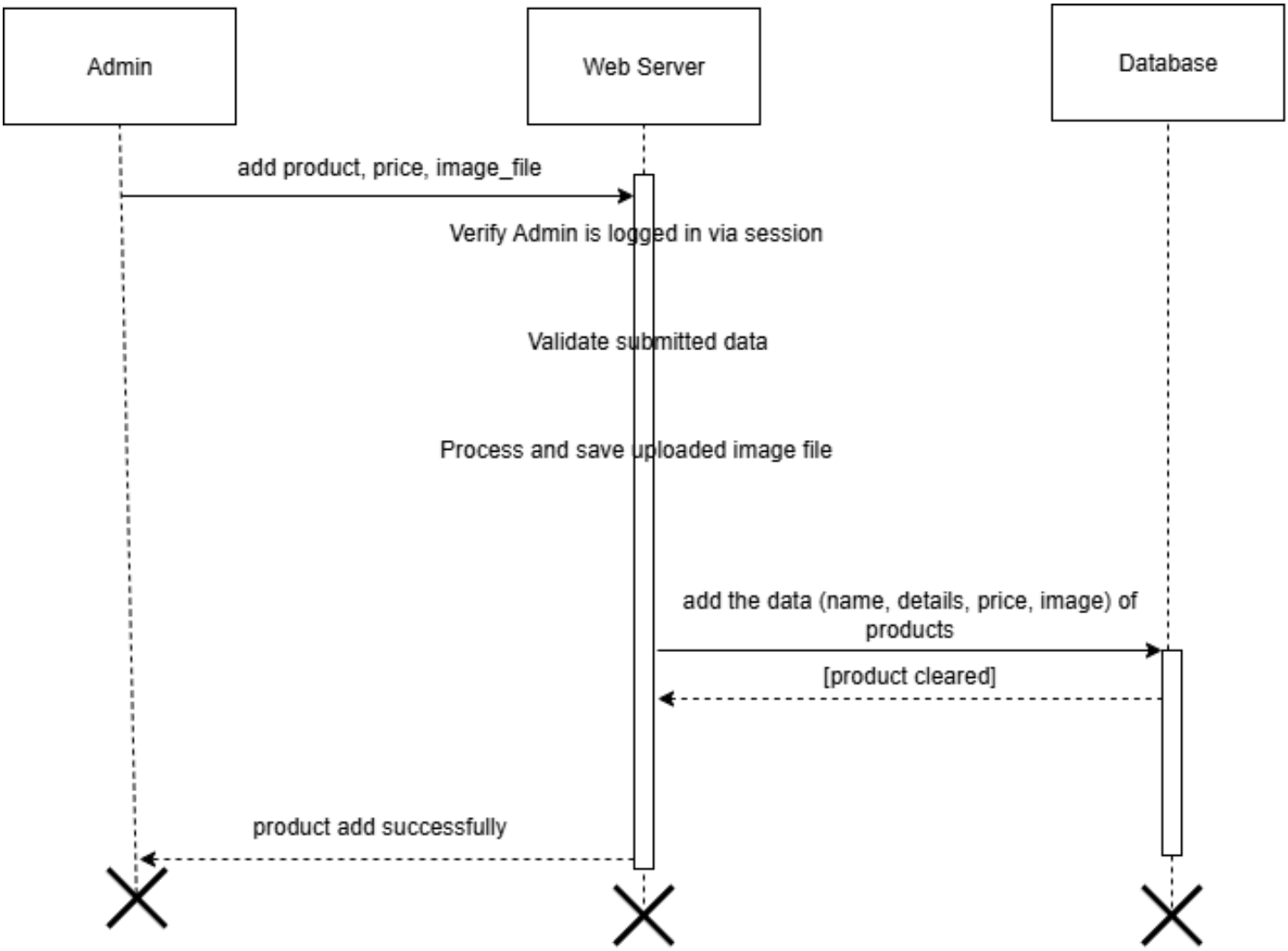

Figure 39: Sequence Diagram for Admin Adds a New Product

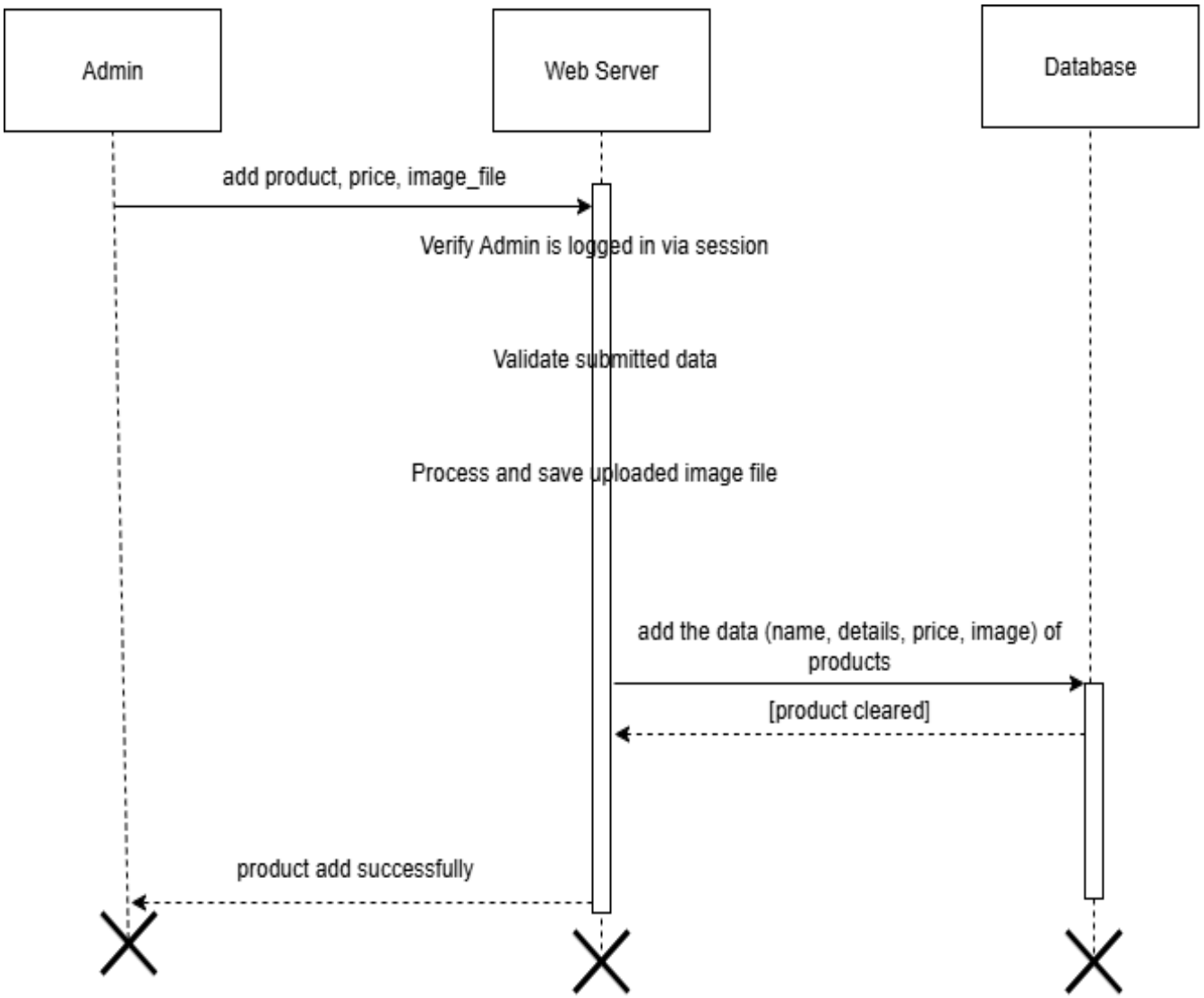

Figure 40: Sequence Diagram for Admin Updates an Order Status

Chapter 9: Conclusion

This document about the E-Commerce Based Client Management System gives a clear and detailed overview of how to manage an online store. It is created to help throughout the entire software development process, from the first planning stage to the final implementation.

By following the guidelines and requirements in this document, developers and stakeholders can better understand the system's features, avoid mistakes, and ensure the project runs smoothly. This detailed document also helps reduce confusion and keeps the work consistent.

Additionally, this document is useful not only for developers but also for students, giving them practical knowledge about how a real e-commerce system is designed and documented. Overall, it serves as both a guide for building the system and a learning resource for future projects in e-commerce.

Appendix:

Meeting Schedule

1.

Date: 28/6/25

Discussion: Preparing the steps for stakeholder identification

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124
Abdul Wadud	2022-2-60-133
Md Anik Khan Akash	2021-1-60-133

2.

Date: 2/7/25

Discussion: Discussing the minimum requirement from the Facebook Page Owner

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

3.

Date: 2/7/25

Discussion: Identifying Viewpoints of Stakeholder

Member:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

4.
Date: 3/7/25

Discussion: Preparing the requirement Engineering

Member:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

5.
Date: 3/7/25

Discussion: Correcting the requirement Engineering

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

6.
Date: 10/7/25

Discussion: Preparing the Use Case Scenario

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124
Abdul Wadud	2022-2-60-133

7.
Date: 12/7/25

Discussion: Correcting the Use Case Scenario

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

8.
Date: 14/7/25

Discussion: Doing the Activity Diagram

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

9.
Date:16/7/25

Discussion: Correcting the Activity Diagram

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

10.
Date: 1/8/25

Discussion: Discussing Data Flow Diagram, Class Diagram and Class Card
Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

13.
Date: 2/8/25

Discussion: Preparing Data Flow Diagram, Class Diagram and Class Cards

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

14.
Date: 3/8/25

Discussion: Correcting Data Flow Diagram and Class Diagram and Class Cards

Members:

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

15.
Date: 8/8/25

Discussion: Preparing Sequence Diagram

Members:

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

16.

Date: 12/8/25

Discussion: Correcting Sequence Diagram

Members:

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

17.

Date: 14/8/25

Discussion: Preparing State Diagram and Deployment Diagram

Members:

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

18.

Date: 18/8/25

Discussion: Correcting State Diagram and Deployment Diagram

Members:

Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124

19.

Date:19/8/25

Discussion: Discussing About the report

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124
Abdul Wadud	2022-2-60-133

20.
Date: 20/8/25

Discussion: Finalizing the report

Members:

Name	Student Id
Mohammad Mehedi Hasan Munna	2022-1-60-138
Aditya Debnath	2022-2-60-124