1.1 SCOPE OF THE APP

The **scope** of the e-shop application includes the following components:

1. User Interface: A user-friendly website or mobile application that allows customers to browse and purchase products.
2. Product Catalog: A database that stores information about the products offered by the e-shop platform, including product descriptions, images, and prices.
3. Shopping Cart: A virtual shopping cart that allows customers to add items to their order and proceed to checkout.
4. Payment Processing: A secure payment processing system that supports various payment methods, such as credit cards, PayPal, and bank transfers.
5. Shipping and Delivery: A system for shipping and delivering products to customers, including options for shipping methods, delivery dates, and tracking.
6. Order Management: A system for managing orders, including tracking the status of orders, processing returns and refunds, and generating invoices.
7. Customer Management: A system for managing customer accounts, including customer profiles, order history, and communication with customers.
8. Seller Management: A system for managing seller accounts, including seller profiles, product listings, and payment processing.
9. Shops: A system for managing the individual shops on the e-shop platform, including shop profiles, product listings, and payment processing.
10. Shop Categories: A system for categorizing the shops on the e-shop platform, allowing customers to easily find the products and services they are looking for.
11. Administration: A system for managing the e-shop platform, including managing user accounts, product catalog, and site configuration.

The scope may also include features such as customer reviews, ratings, and recommendations, as well as promotional tools such as discounts, coupons, and gift cards. The scope of the project should be defined and agreed upon by the stakeholders in the Inception phase to ensure that the development effort stays focused on delivering value to the customers and stakeholders.

1.2 DEFINING PROBLEMS TO BE RESOLVED

Here are some common **problems** that could arise with the components of an e-shop application:

1. User Interface: Slow page load times, confusing navigation, and poor mobile responsiveness can lead to a poor user experience and decreased sales.
2. Product Catalog: Incorrect product information, out-of-date inventory, and missing product images can lead to customer confusion and decreased sales.
3. Shopping Cart: Technical issues with the shopping cart, such as lost items or incorrect pricing, can result in customer frustration and lost sales.
4. Payment Processing: Security vulnerabilities, slow payment processing times, and payment errors can lead to lost sales and decreased customer trust.
5. Shipping and Delivery: Incorrect shipping information, delayed shipments, and lost or damaged packages can result in customer dissatisfaction and lost sales.
6. Order Management: Inconsistent order tracking, slow processing times, and incorrect order information can lead to customer frustration and decreased trust.
7. Customer Management: Poor customer service, slow response times, and inadequate customer support can lead to decreased customer satisfaction and lost sales.
8. Seller Management: Inconsistent seller policies, delayed payments, and slow resolution of disputes can lead to decreased seller satisfaction and lost sales.
9. Shops: Inconsistent shop policies, slow shop management, and poor shop quality can lead to decreased customer satisfaction and lost sales.
10. Shop Categories: Inconsistent or incorrect shop categorization, slow category updates, and poor category navigation can lead to decreased customer satisfaction and lost sales.
11. Administration: Slow site updates, inconsistent site policies, and inadequate site management can lead to decreased customer satisfaction and lost sales.

Rational Unified Process

RUP (Rational Unified Process) is a software development methodology that provides a structured and iterative approach to software development. It divides the software development lifecycle into four phases: Inception, Elaboration, Construction, and Transition.

1. Inception: In this phase, the scope of the project is determined and the high-level requirements for the e-shop platform are defined. This includes determining the stakeholders involved, conducting a feasibility study, and creating a high-level vision for the platform.
2. Elaboration: In this phase, the high-level requirements are refined, and the architecture of the e-shop platform is developed. This includes defining the functional and non-functional requirements, creating a detailed design, and conducting risk analysis.
3. Construction: In this phase, the e-shop platform is built, including the user interface, product catalog, shopping cart, payment processing, shipping and delivery, order management, customer management, and administration. This phase also includes testing and integration of all the components.
4. Transition: In this phase, the e-shop platform is deployed and transitioned to the production environment. This includes final testing, user acceptance testing, and deployment of the platform. Ongoing maintenance and support activities are also planned during this phase.

Throughout each phase of the RUP process, regular reviews and evaluations are conducted to ensure that the platform is on track to meet the desired outcome. The end goal of the RUP process for an e-shop application is to deliver a high-quality, user-friendly, and reliable platform that meets the needs of customers, sellers, and administrators.

Adjusting RUP in E-Shop Application

Adjusting RUP process in our application the phase become clearer:

1. Inception: In this phase, the scope of the e-shop application project is determined and the high-level requirements for the platform are defined. This includes determining the stakeholders involved, such as the customers, sellers, administrators, and shops, conducting a feasibility study, and creating a high-level vision for the platform.
2. Elaboration: In this phase, the high-level requirements are refined, and the architecture of the e-shop platform is developed. This includes defining the functional and non-functional requirements, such as creating a detailed design for the user interface, product catalog, shopping cart, payment processing, shipping and delivery, order management, customer management, and administration, as well as defining the shop and shop category components.
3. Construction: In this phase, the e-shop platform is built, including the user interface, product catalog, shopping cart, payment processing, shipping and delivery, order management, customer management, and administration. This phase also includes testing and integration of all the components, including the shops and shop categories.
4. Transition: In this phase, the e-shop platform is deployed and transitioned to the production environment. This includes final testing, user acceptance testing, and deployment of the platform. Ongoing maintenance and support activities are also planned during this phase.

3.1 Inception Phase

Based on the stakeholder interviews, the requirement capture process during the Inception phase of the RUP methodology for our application include the following steps:

1. **Requirements catalog**: Creating the catalog of the requirements gathered from the stakeholder interviews we noticed that the catalog should include functional and non-functional requirements, such as:

- User registration and login
- Product search and filtering
- Product details and reviews
- Shopping cart and checkout
- Payment processing
- Order tracking and management
- Customer service and support
- Administration and management of seller accounts
- Management of shop categories and product categories

2. **Requirements prioritization**: Prioritizing the above requirements can be done using different techniques, including MoSCoW (Must have, Should have, Could have, Won't have), Kano Model, and Cost-Benefit Analysis. Here, we prioritized them using the MoSCoW method:

Must Have:

• User registration and login
• Product search and filtering
• Product details and reviews
• Shopping cart and checkout
• Payment processing
• Order tracking and management

Should Have:

• Customer service and support

Could Have:

• Administration and management of seller accounts
• Management of shop categories and product categories

Won't Have: N/A

These priorities are based on the fact that the must-have requirements are the core functionalities of the e-shop platform and are critical to its success. The should-have requirements add value but are not critical to the platform's functioning. The could-

have requirements are nice-to-have features that could potentially be added in the future if resources and budget allow.

3. **High-level requirements**: Develop high-level requirements define the scope of the platform, including the functional and non-functional requirements. These requirements include:

- A user-friendly and intuitive interface for customers to browse and purchase products
- Secure payment processing to protect customer data and financial information
- A flexible and scalable platform to accommodate growth and changing requirements
- Fast and reliable performance to ensure a positive customer experience
- Integration with existing systems and tools to streamline the management of the platform

4. **Requirements Analysis**: During the requirements analysis phase, we identified the following requirements for the platform:

- User authentication and authorization for customers, sellers, and administrators.
- A shopping cart for customers to add and purchase products.
- A product catalog with categories, images, and descriptions for customers to browse and search for products.
- A seller dashboard for sellers to manage their products and orders.
- An administrative dashboard for administrators to manage the platform and its users.
- Payment gateway integration for customers to make secure online payments.
- A reporting and analytics system to monitor the platform's performance and customer behavior.

Based on these requirements, we analyzed the risks, challenges, and technical constraints associated with implementing each of these features and will develop a comprehensive set of use cases and user stories to further define and clarify the requirements.

5. **Feasibility Study**: A feasibility study is an analysis of a proposed project or system that determines whether it is technically and economically viable. The purpose of a feasibility study is to identify potential risks and challenges that may arise during the implementation of the project, as well as to estimate the potential costs and benefits of the project. Our feasibility study for the e-shop platform made us notice that:

- The required technology and infrastructure are readily available and accessible.
- The estimated cost of development and ongoing operation and maintenance is within budget.
- The required resources, skills, and infrastructure are available to support the platform's operation and maintenance.

Based on these findings, we determined that the project is viable and should proceed to the next phase of development.

6. **High-level Vision**: A high-level vision is a concise, high-level description of what a project or system is intended to achieve. The high-level vision for the e-shop platform include the following elements:

- The platform's goal is to provide an online marketplace for customers to purchase products from multiple sellers in a secure and convenient manner.
- The platform's functional requirements include user authentication and authorization, a shopping cart, a product catalog, seller and administrative dashboards, payment gateway integration, and reporting and analytics.
- The platform's non-functional requirements include security, scalability, availability, and usability.
- The platform's target market is consumers looking to purchase products online from multiple sellers.
- The platform will be built using modern web technologies and will be hosted in a secure and scalable cloud infrastructure.

This process will help to ensure that the platform meets the needs of the stakeholders and is feasible to build. The requirements gathered during this process will be used in the Elaboration phase to refine and further define the requirements for the platform.

7. **Potential Risks**: In an application project, there are several potential risks that need to be considered, including:

1. Technical risks: This includes the risk of technical failures during the development process, as well as the risk of compatibility issues with different devices and platforms.
2. Security risks: This includes the risk of data breaches, hacking, and fraud, which could compromise the personal and financial information of users and sellers.
3. User experience risks: This includes the risk of poor user experience due to usability and navigation issues, slow page load times, or confusing checkout processes.
4. Business risks: This includes the risk of low adoption or usage of the platform, low revenue, or decreased customer loyalty.
5. Compliance risks: This includes the risk of non-compliance with data privacy regulations, such as GDPR, as well as payment processing and fraud protection regulations.
6. Scalability risks: This includes the risk of the platform being unable to handle a large volume of traffic or transactions.
7. Integration risks: This includes the risk of integration issues with third-party systems, such as payment processors or shipping providers.

To mitigate these risks, it's important to have a thorough risk management plan in place and to regularly assess and reassess the potential risks throughout the development process.

8. **Requirements review**:

1. User registration and login: This requirement will be reviewed for usability and security. The review will consider how user accounts will be created, managed, and deleted, and what information will be required for registration. The review will also consider the security measures in place to protect user data, such as password complexity and encryption.
2. Product search and filtering: This requirement will be reviewed for functionality and ease of use. The review will consider how products can be searched for, filtered, and displayed, and how users will be able to access product information.
3. Product details and reviews: This requirement will be reviewed for completeness and accuracy. The review will consider what information will be displayed for each product, such as images, descriptions, and specifications, and how reviews will be collected and displayed.
4. Shopping cart and checkout: This requirement will be reviewed for functionality and user experience. The review will consider how items can be added to the cart, how the cart can be reviewed and edited, and how checkout will be processed, including payment options and shipping information.
5. Payment processing: This requirement will be reviewed for security and compliance. The review will consider how payments will be processed, including the types of payment methods accepted, and the security measures in place to protect payment data.
6. Order tracking and management: This requirement will be reviewed for functionality and user experience. The review will consider how users will be able to track their orders, receive updates, and communicate with customer service.
7. Customer service and support: This requirement will be reviewed for availability and quality. The review will consider how customers will be able to contact customer service, what support options will be available, and how response times will be measured.
8. Administration and management of seller accounts: This requirement will be reviewed for functionality and security. The review will consider how seller accounts will be managed, how seller data will be collected and stored, and how seller payments will be processed.
9. Management of shop categories and product categories: This requirement will be reviewed for functionality and user experience. The review will consider how shop and product categories will be created, managed, and deleted, and how users will be able to access and filter products based on category.

In the requirement review, the requirements will be evaluated against the project's goals, constraints, and assumptions, and any necessary changes will be documented. This review will ensure that the requirements are complete, accurate, and aligned with the project's objectives, and will help to identify any potential risks or issues that need to be addressed.

3.2 Analysis – Design

Before, we start the Designing of the UML Diagrams we must define our Classes, Objects and Relationships. Considering the analysis of the 3.1, here is the definition of our Classes, Objects and Relationships:

Classes:

1. Admin: This class represents the system administrator and includes attributes such as username, password, and contact information.
2. Seller: This class represents the sellers on the platform and includes attributes such as username, password, and contact information, as well as a list of products they are selling.
3. Customer: This class represents the customers of the platform and includes attributes such as username, password, billing information, and a list of purchased products.
4. Product: This class represents the products available for purchase on the platform and includes attributes such as name, description, price, and image.
5. Order: This class represents the orders placed by customers and includes attributes such as order date, delivery address, and a list of ordered products.
6. Shop: This class represents the online shops on the platform and includes attributes such as shop name, shop address, and a list of products being sold.
7. Shop Category: This class represents the categories of shops available on the platform and includes attributes such as category name and a list of shops that belong to that category.
8. Product Category: This class represents the categories of products available on the platform and includes attributes such as category name and a list of products that belong to that category.
9. Payment: This class represents an abstract or concrete entity in the system that defines the payment process for the e-commerce application
10. Payment Category: This class represents the categories of payments available.
11. Cart: This class represents the shopping cart used by customers to store the items they want to purchase.

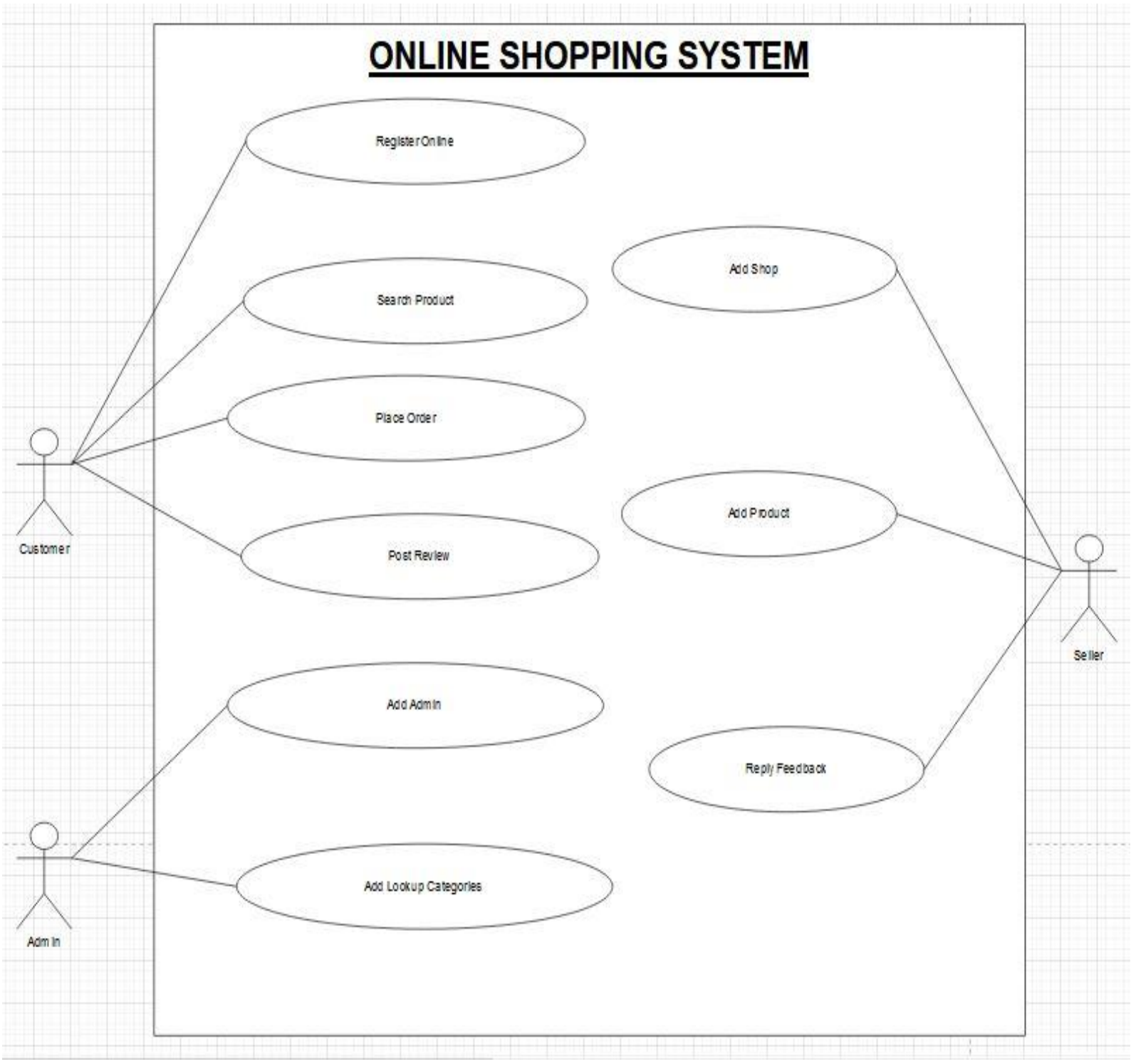
Objects:

1. Admin objects: Instances of the Admin class, representing individual administrators of the platform.
2. Seller objects: Instances of the Seller class, representing individual sellers on the platform.
3. Customer objects: Instances of the Customer class, representing individual customers of the platform.
4. Product objects: Instances of the Product class, representing individual products available for purchase.
5. Order objects: Instances of the Order class, representing individual orders placed by customers.
6. Shop objects: Instances of the Shop class, representing individual online shops on the platform.
7. Shop Category objects: Instances of the Shop Category class, representing individual categories of shops on the platform.
8. Product Category objects: Instances of the Product Category class, representing individual categories of products on the platform.
9. Payment objects: An instance of the Payment class would represent a specific payment made by a customer.
10. Payment Category objects: An instance of the Payment Category class would represent a specific payment category.
11. Cart objects: An instance of the Cart class would represent a specific shopping cart used by a customer.

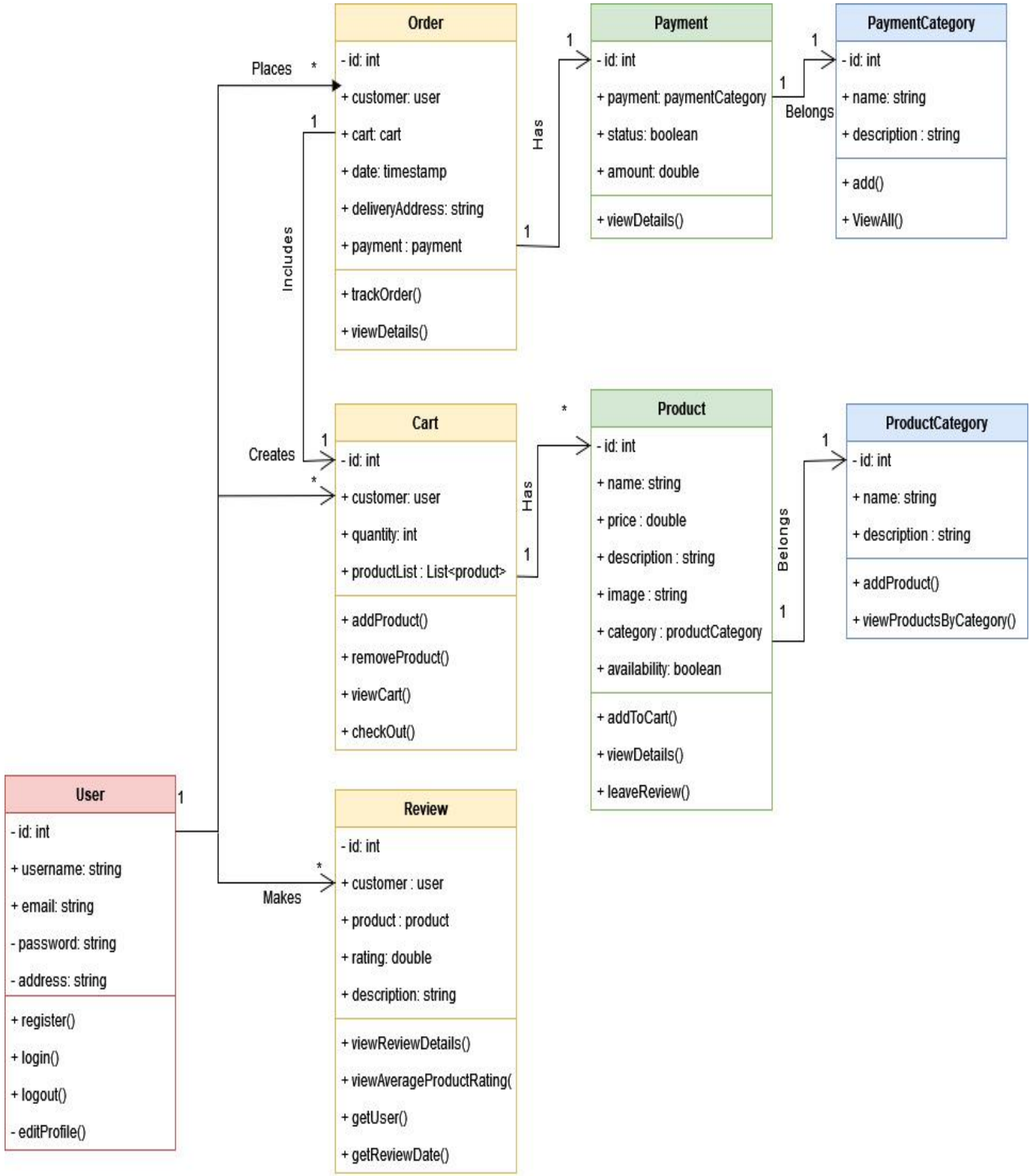
Relationships:

1. Inheritance: The Seller, Customer, and Payment classes might inherit from a common User class, which includes shared attributes and methods.
2. Association: The Order class might have an association with the Customer class, indicating that a customer places an order. The Shop class might have an association with the Seller class, indicating that a seller runs a shop. The Payment class might have an association with the Order class, indicating the payment process for an order.
3. Aggregation: The Product class might have an aggregation relationship with the Order class, indicating that an order can contain multiple products. The Shop class might have an aggregation relationship with the Product class, indicating that a shop sells multiple products. The Cart class might have an aggregation relationship with the Product class, indicating that a cart can contain multiple products.
4. Composition: The Seller class might have a composition relationship with the Shop class, indicating that a seller has control over the shop they run. The Shop Category class might have a composition relationship with the Shop class, indicating that a shop belongs to a specific category. The Product Category class might have a composition relationship with the Product class, indicating that a product belongs to a specific category. The Payment Category class might have a composition relationship with the Payment class, indicating that a payment belongs to a specific category.

3.1 Use Case Diagram (1st Version)



3.2 Class Diagram (1st Version)



4.1 Elaboration Phase

The **Elaboration Phase** is the second phase in the Rational Unified Process (RUP) software development methodology. It is focused on defining the detailed architecture of the system and ensuring that the requirements, risks and high-level vision are fully understood and properly addressed. During the Elaboration Phase, the development team will create detailed models and diagrams to represent the architecture of the system, such as use case diagrams, class diagrams, object diagrams, collaboration or communication diagrams, sequence diagrams, activity diagrams, state chart diagrams, component diagrams, and deployment diagrams. These diagrams help to describe the relationships and interactions between objects, classes, and components, as well as the flow of data and processes within the system.

The use case diagram represents the system's functions and services from the user's point of view. The class diagram represents the objects and classes within the system and their relationships. The object diagram represents the instances of the classes at a specific moment in time. The collaboration or communication diagram represents the interactions between objects and classes. The sequence diagram represents the order of messages between objects over time. The activity diagram represents the flow of activities within the system. The state chart diagram represents the possible states of the system and how they transition between states. The component diagram represents the physical components and the relationships between them. The deployment diagram represents the hardware and software components on which the system will run.

By creating these detailed models and diagrams, the development team can identify potential problems, design solutions, and develop a detailed plan for implementing the system. This phase is crucial for ensuring that the system is designed correctly and meets the needs of the stakeholders.

The following are some **steps** that we should do in the Elaboration phase of the RUP methodology for our e-shop application:

1. **Use Case Diagram (v.2):** This diagram is used to represent the functional **requirements** of the system and the **interactions** between the system and its actors. It helps to identify the main functions of the system and the relationships between the actors and the system. In this **updated version**, we noticed that it was important to add **shop** and **shop category** classes and objects.

2. **Class Diagram (v.2):** This diagram is used to represent the **structure** of the system, including classes, their attributes, and their relationships. It provides a clear understanding of the objects and the relationships between them in the system. In this **updated version**, we noticed that it was important to add **shop** and **shop category** classes and objects.

3. **Object Diagram (v.1):** This diagram is used to represent the **instances** of classes and the relationships between them at a specific point in time. It helps to understand the behavior of the system at runtime.

4. **Collaboration or Communication Diagram (v.1):** This diagram is used to represent the flow of **messages** between objects or components in a system. It helps to understand how the objects interact and communicate with each other.

5. **Sequence Diagram (v.1):** This diagram is used to represent the **interaction between objects or components over time**, including the order of messages and the conditions under which they are sent. It helps to understand the flow of events in the system and the dependencies between the objects.

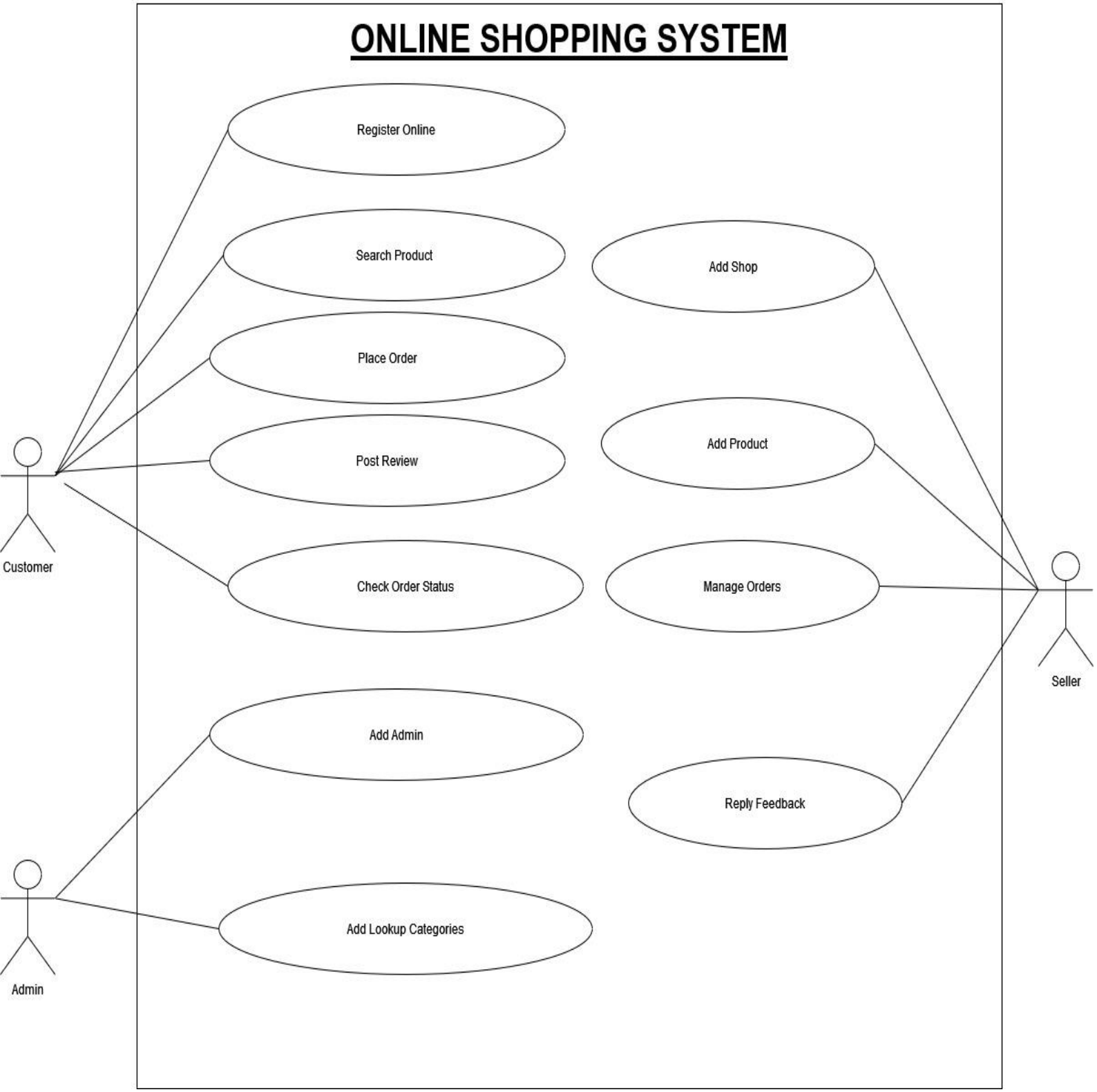
6. **Activity Diagram (v.1):** This diagram is used to represent the **flow of activities** within a system. It helps to understand the logic of a process and the relationships between the steps involved.

7. **State Chart Diagram (v.1):** This diagram is used to represent the **states and transitions of objects or components in a system**. It helps to understand the behavior of the system over time, including the possible states and events that trigger transitions between them.

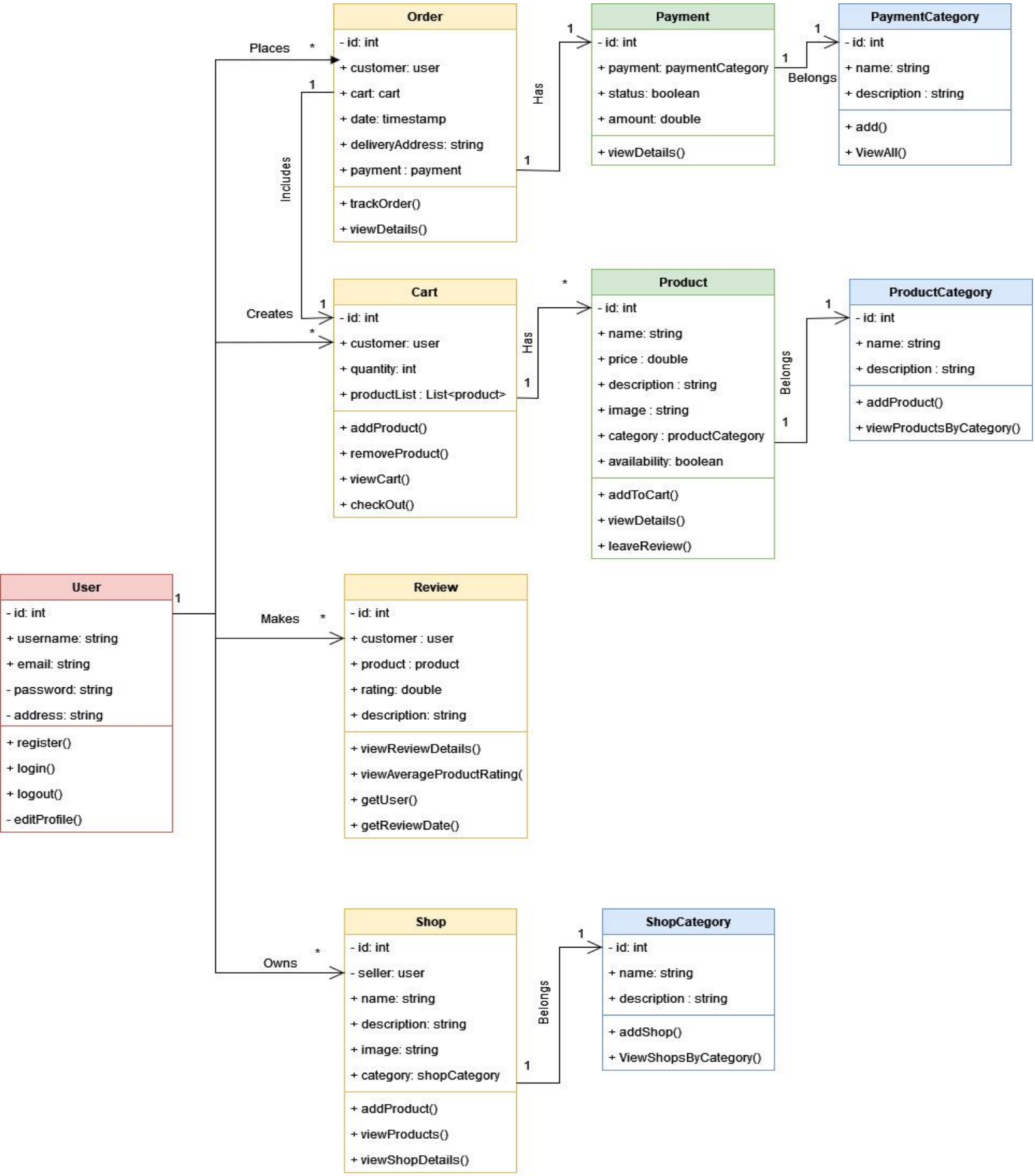
8. **Component Diagram (v.1):** This diagram is used to represent the components and the **relationships** between them in a system. It helps to understand the structure of the system and the dependencies between the components.

9. **Deployment Diagram (v.1):** This diagram is used to represent the **physical deployment** of the system, including the hardware and software components and their relationships. It helps to understand the environment in which the system will run and the dependencies between the components.

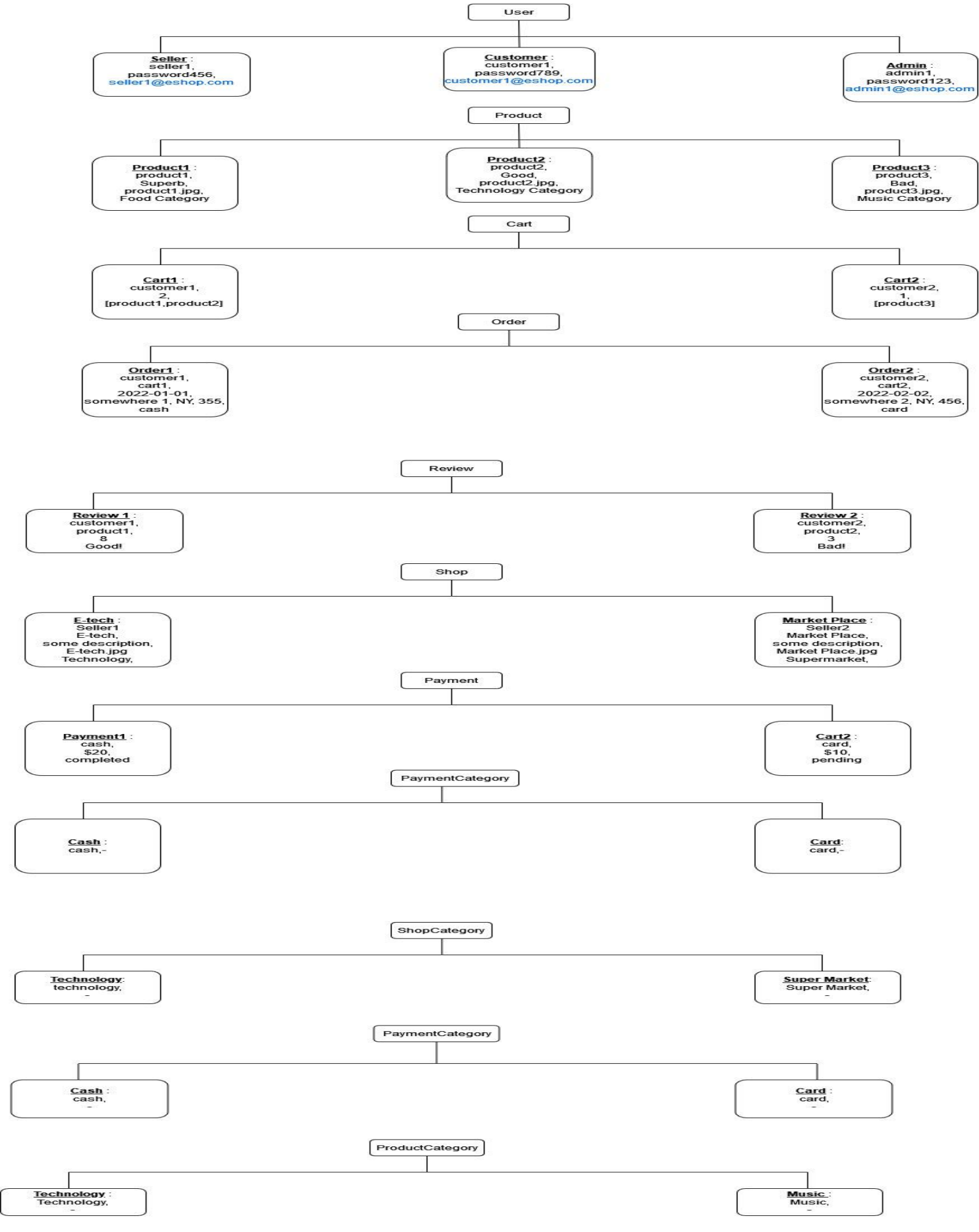
4.1.1 Use Case Diagram (2nd Version)



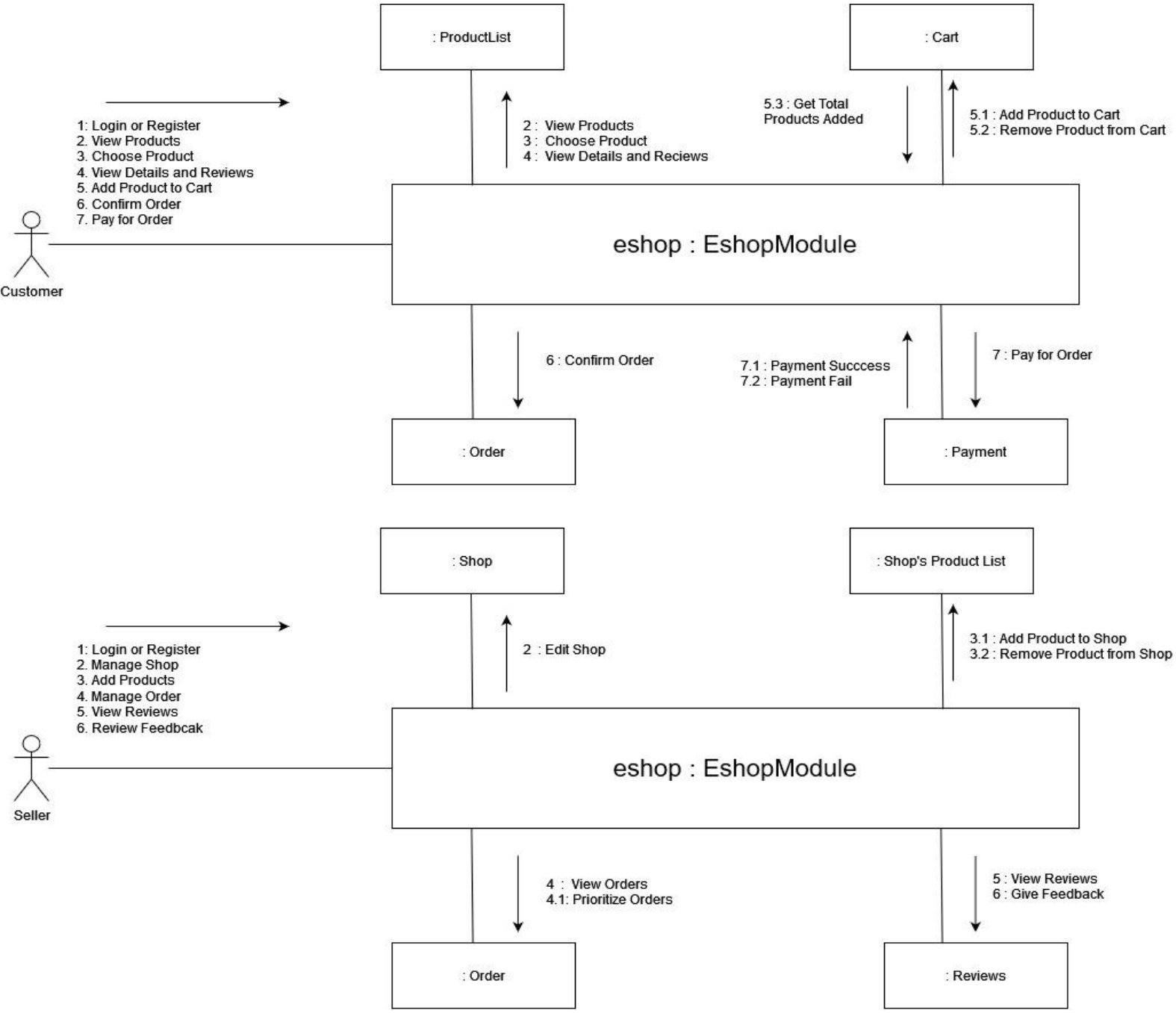
4.1.2 Class Diagram (2nd Version)



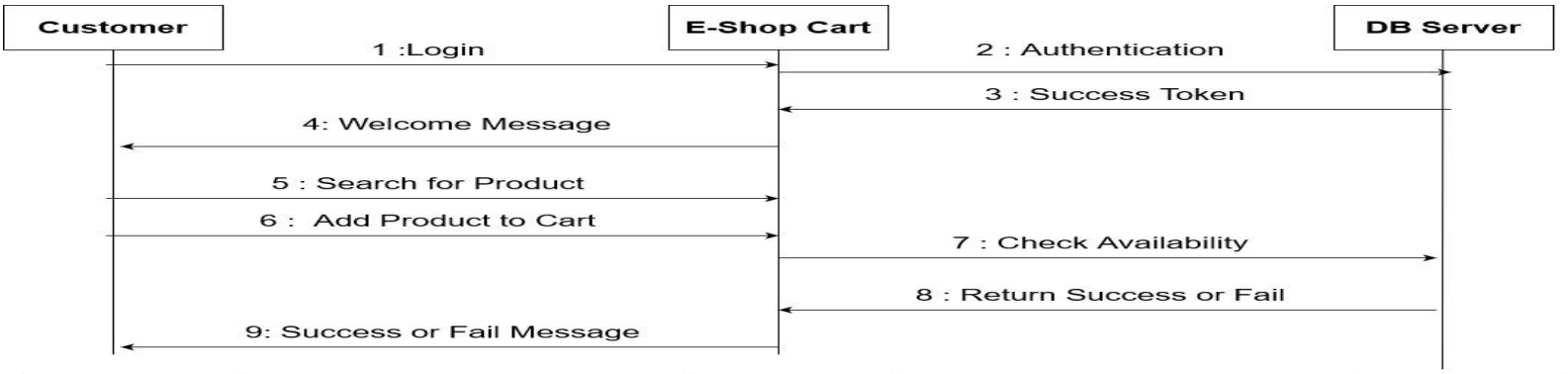
4.1.3 Object Diagram (1st Version)

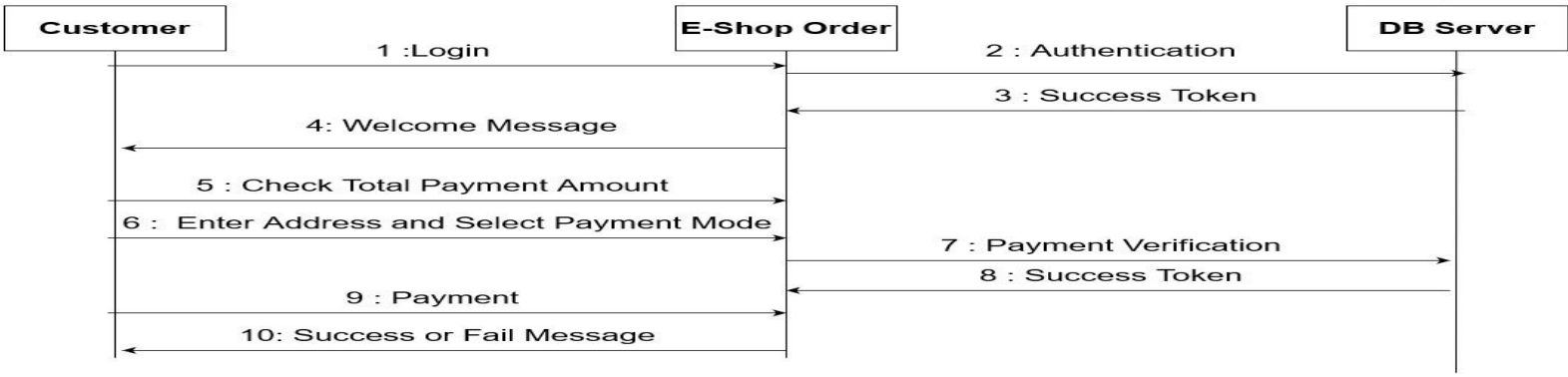


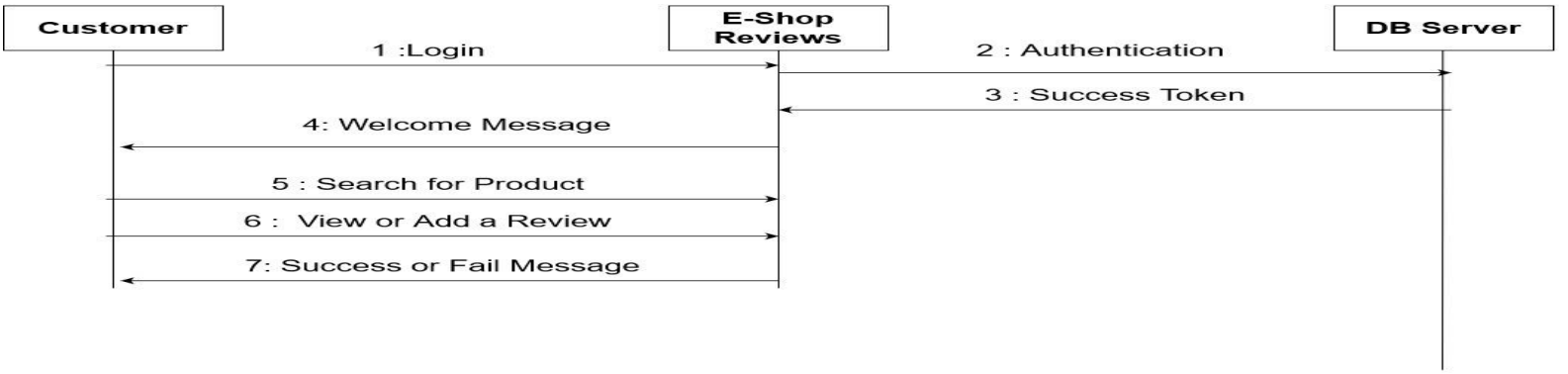
4.1.4 Collaboration or Communication Diagram (1st Version)

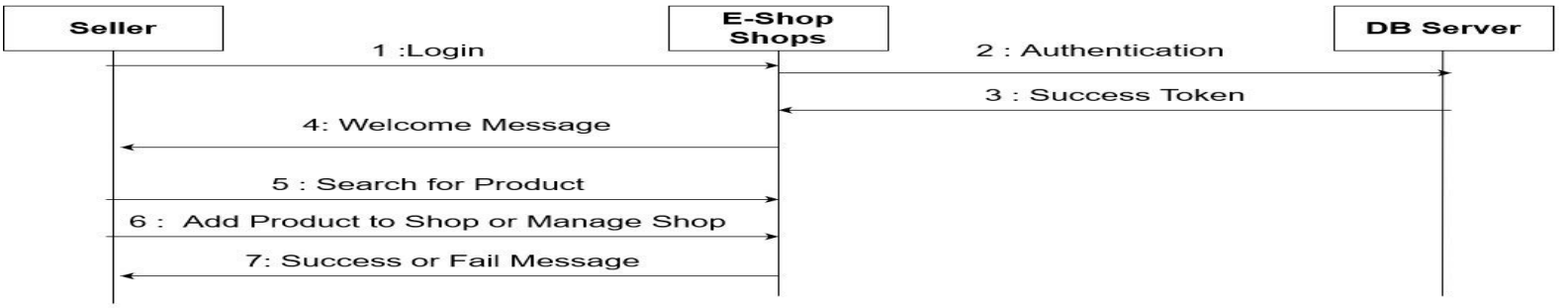


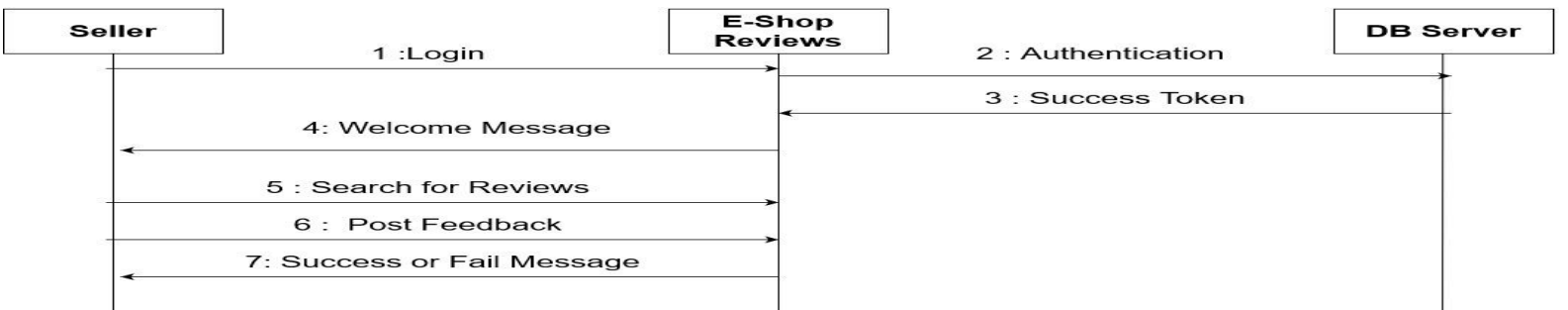
4.1.5 Sequence Diagram (1st Version)



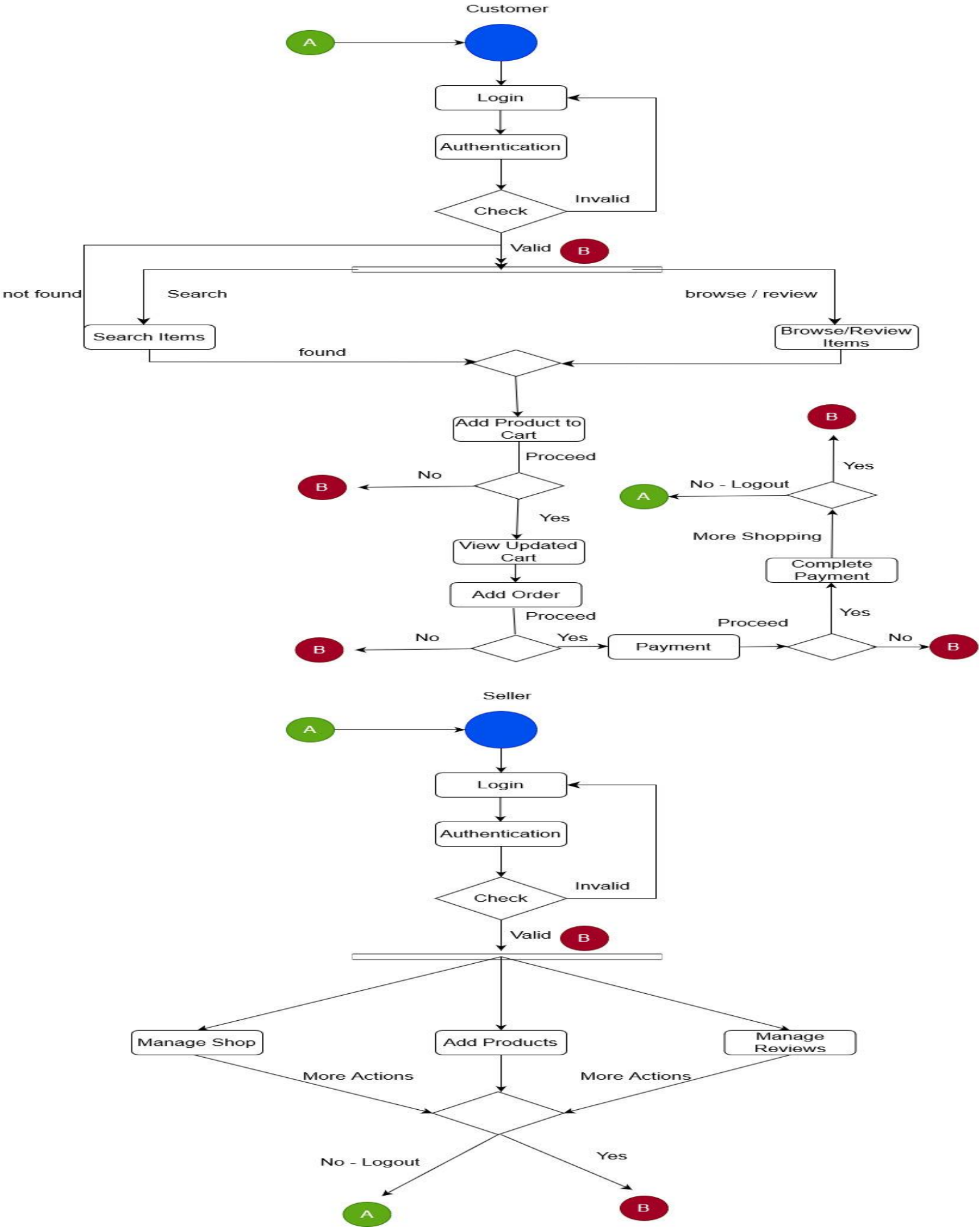




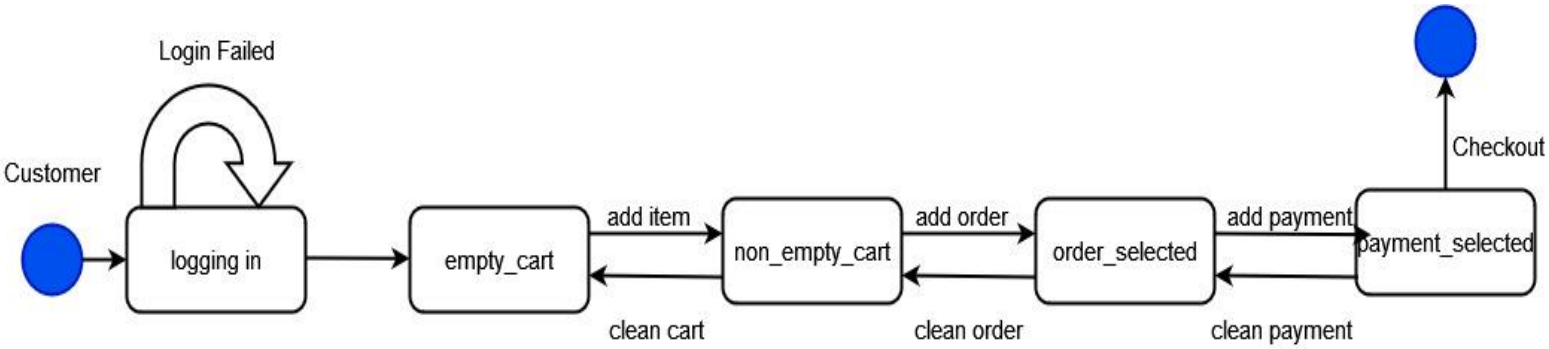


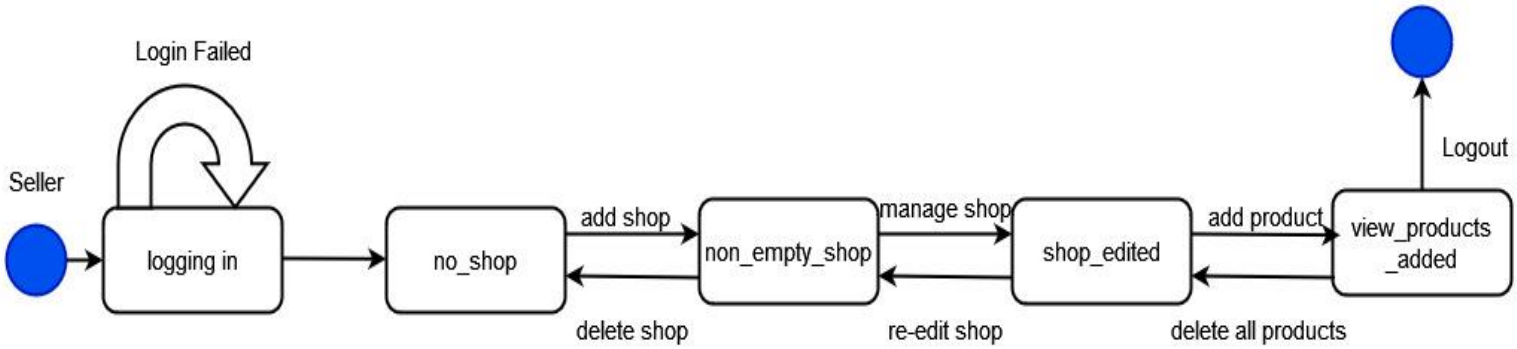


4.1.6 Activity Diagram (1st Version)

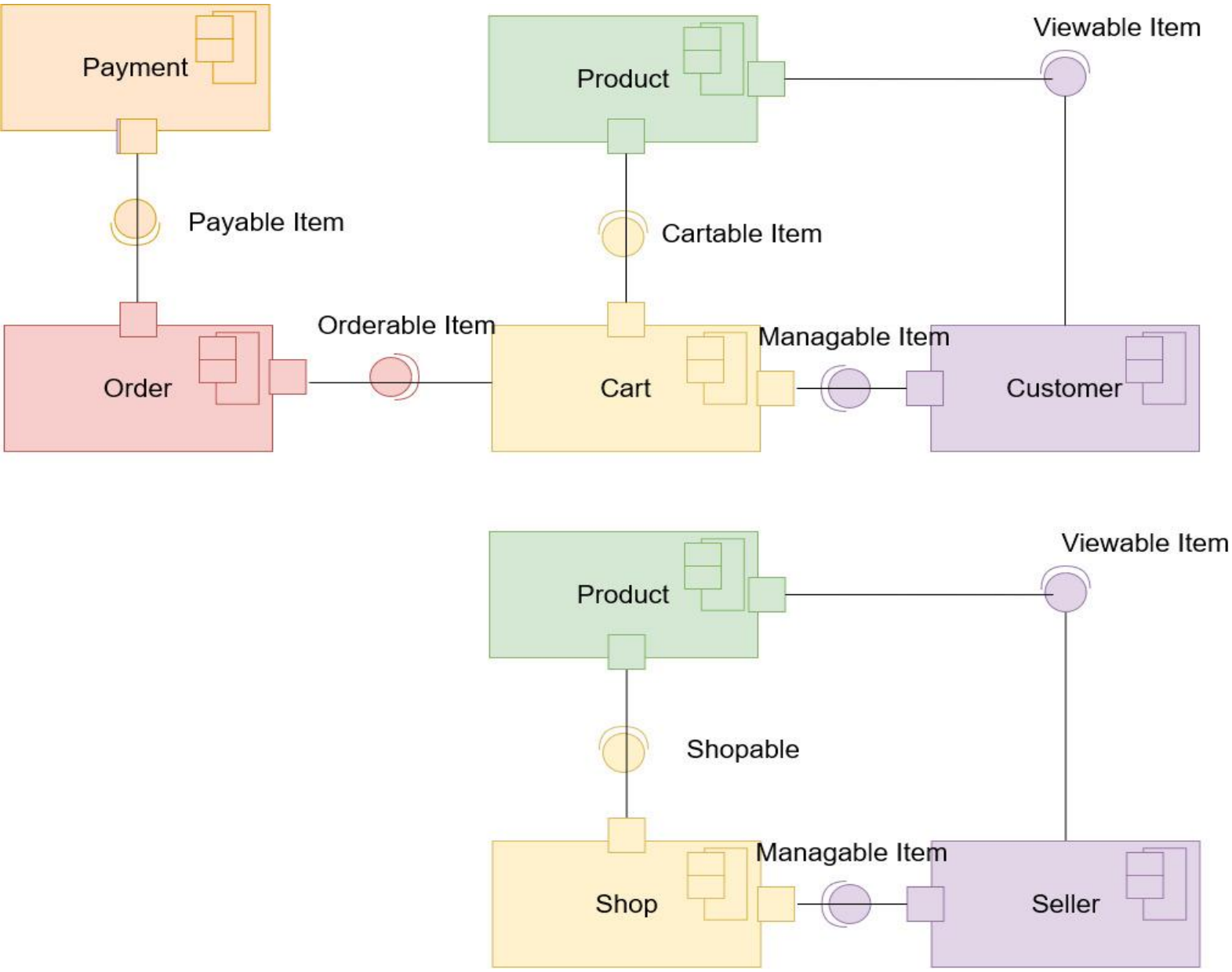


4.1.7 State Chart Diagram (1st Version)

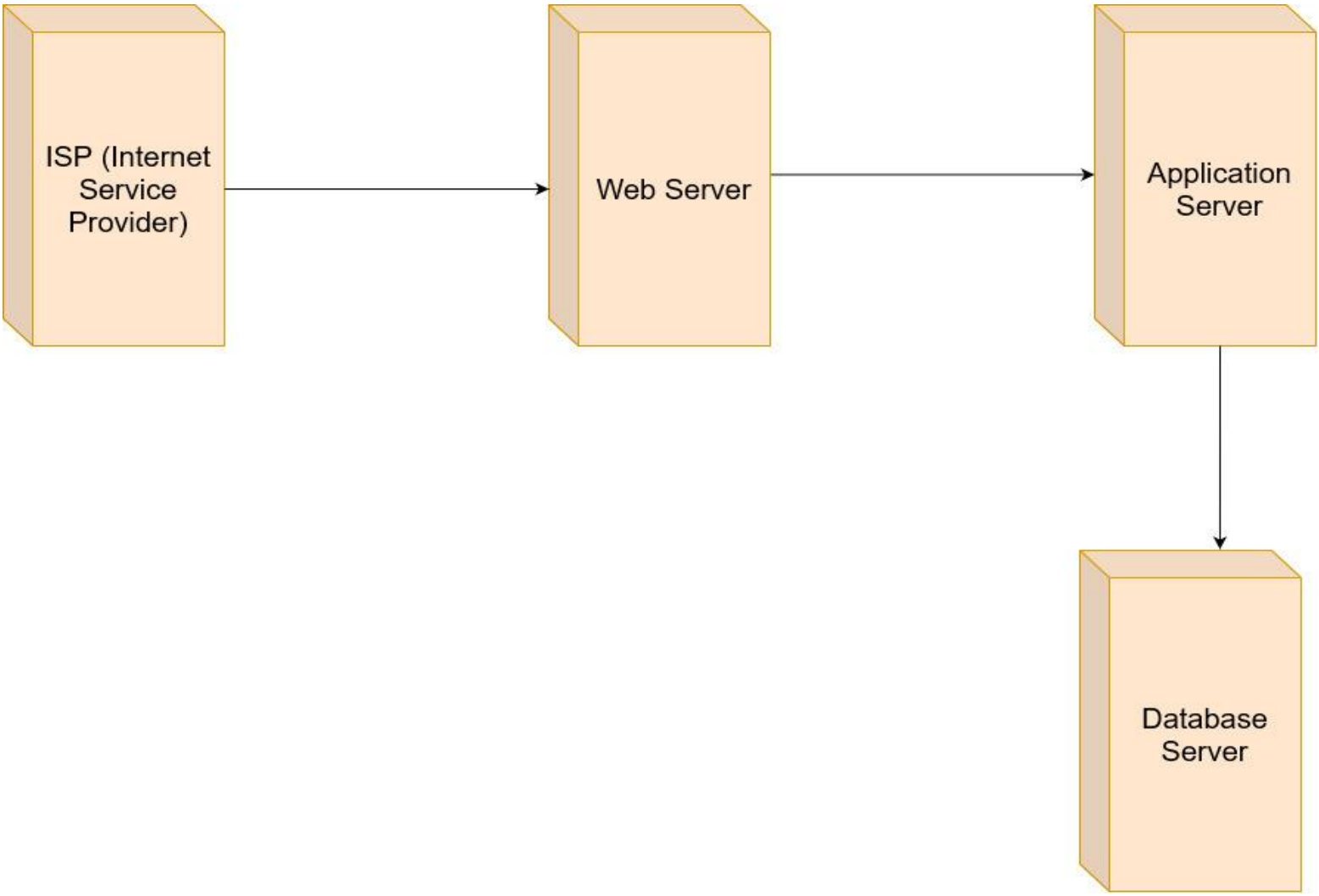




4.1.8 Component Diagram (1st Version)



4.1.9 Deployment Diagram (1st Version)



5.1 Construction Phase

The **Construction Phase** is the third phase in the Rational Unified Process (RUP) software development methodology. It involves the actual development and implementation of the software system, based on the detailed architecture defined in the previous Elaboration Phase. During the Construction Phase, the development team focuses on writing code, testing and integrating components, and building the system incrementally. The development team will also continuously refine and improve the design as the system is being built. The goal of the Construction Phase is to deliver a working version of the system, ready for testing and validation by stakeholders. To achieve this, the development team will use tools and techniques such as version control, build automation, continuous integration and testing, and project management to ensure that the system is built to a high standard, on time and within budget. By the end of the Construction Phase, the development team should have a functioning system that can be demonstrated to stakeholders and stakeholders can provide feedback for further refinement.

During the Construction Phase, the development team will also carry out a range of activities to ensure the quality and reliability of the system, such as code reviews, testing, and debugging. The development team will also use design patterns, best practices, and established software engineering methodologies to ensure that the system is built to be scalable, maintainable, and secure. In addition, the development team will also use agile methodologies, such as Scrum or Kanban, to manage and track progress during the Construction Phase. This will help to ensure that the system is being built according to schedule, and that any issues or problems are identified and addressed in a timely manner. The Construction Phase is also an opportunity for the development team to refine and improve the design of the system, based on feedback from stakeholders and real-world usage. This can lead to a better understanding of the system requirements, and can result in a more robust, flexible and user-friendly system. Overall, the Construction Phase is a critical stage in the software development process, as it involves the actual creation of the software system. Successful completion of this phase sets the foundation for the next phase of the RUP process, the Transition Phase, where the system is deployed and transitioned into production.

The following are some **steps** that we should do in the Construction phase of the RUP methodology for our e-shop application:

1. **Use Case Diagram (v.3):** This diagram is used to represent the functional **requirements** of the system and the **interactions** between the system and its actors. It helps to identify the main functions of the system and the relationships between the actors and the system. In this **updated version**, we noticed that it was important to add **role** and **admin** classes and objects.

2. **Class Diagram (v.3):** This diagram is used to represent the **structure** of the system, including classes, their attributes, and their relationships. It provides a clear understanding of the objects and the relationships between them in the system. In this **updated version**, we noticed that it was important to add **role** and **admin** classes and objects.

3. **Object Diagram (v.2):** This diagram is used to represent the **instances** of classes and the relationships between them at a specific point in time. It helps to understand the behavior of the system at runtime.

4. **Collaboration or Communication Diagram (v.2):** This diagram is used to represent the flow of **messages** between objects or components in a system. It helps to understand how the objects interact and communicate with each other.

5. **Sequence Diagram (v.2):** This diagram is used to represent the **interaction between objects or components over time**, including the order of messages and the conditions under which they are sent. It helps to understand the flow of events in the system and the dependencies between the objects.

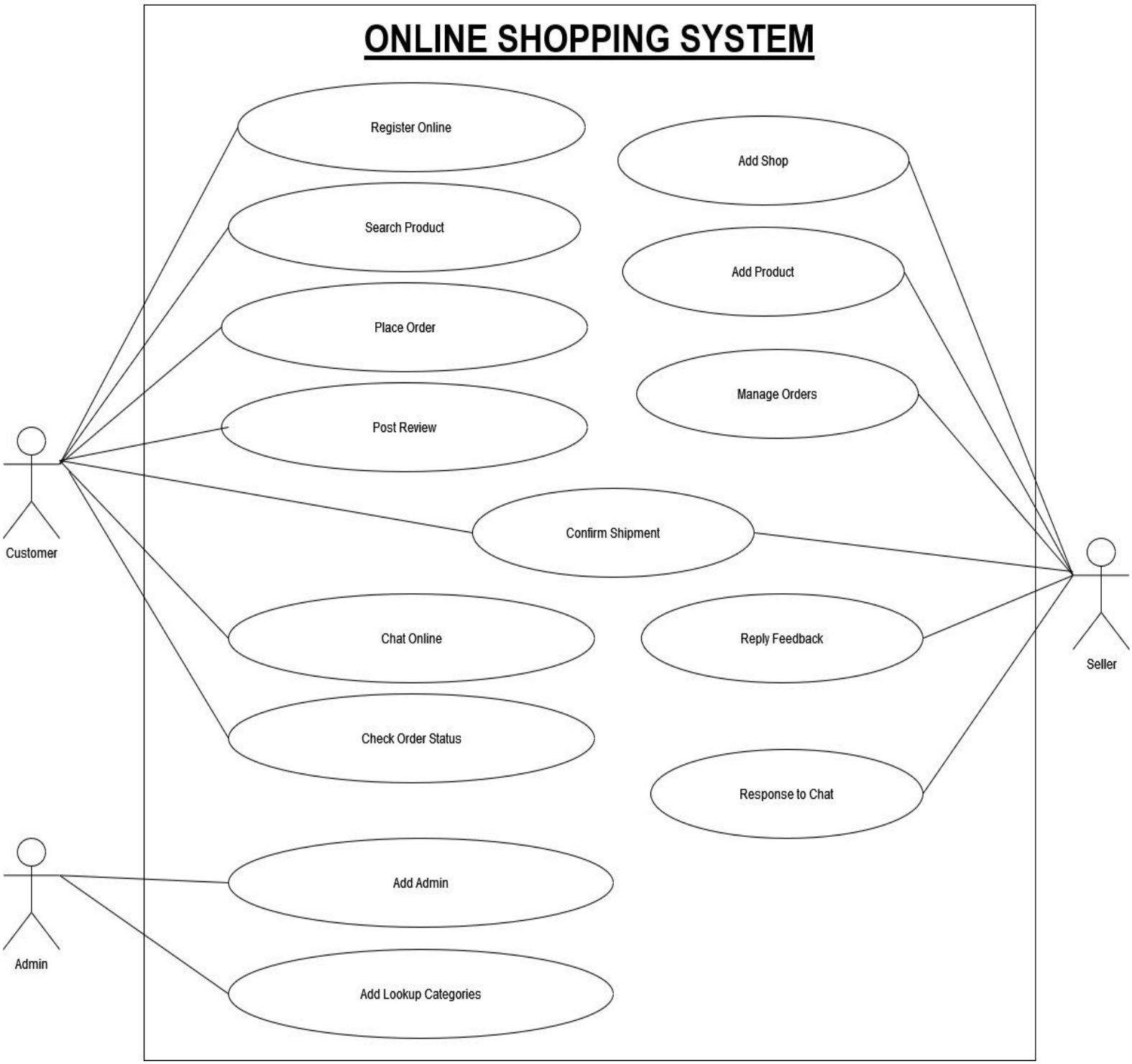
6. **Activity Diagram (v.2):** This diagram is used to represent the **flow of activities** within a system. It helps to understand the logic of a process and the relationships between the steps involved.

7. **State Chart Diagram (v.2):** This diagram is used to represent the **states and transitions of objects or components in a system**. It helps to understand the behavior of the system over time, including the possible states and events that trigger transitions between them.

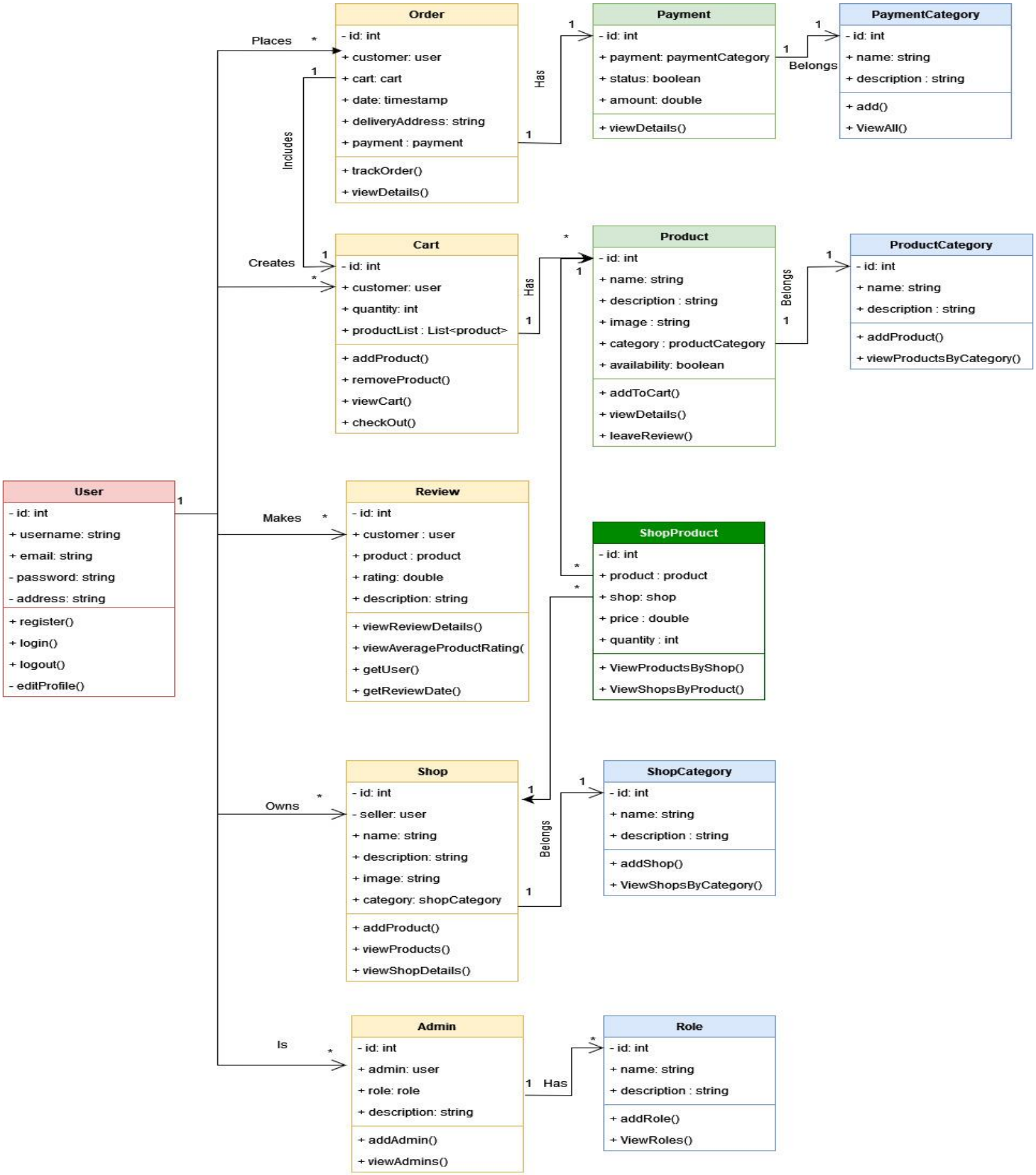
8. **Component Diagram (v.2):** This diagram is used to represent the components and the **relationships** between them in a system. It helps to understand the structure of the system and the dependencies between the components.

9. **Deployment Diagram (v.2):** This diagram is used to represent the **physical deployment** of the system, including the hardware and software components and their relationships. It helps to understand the environment in which the system will run and the dependencies between the components.

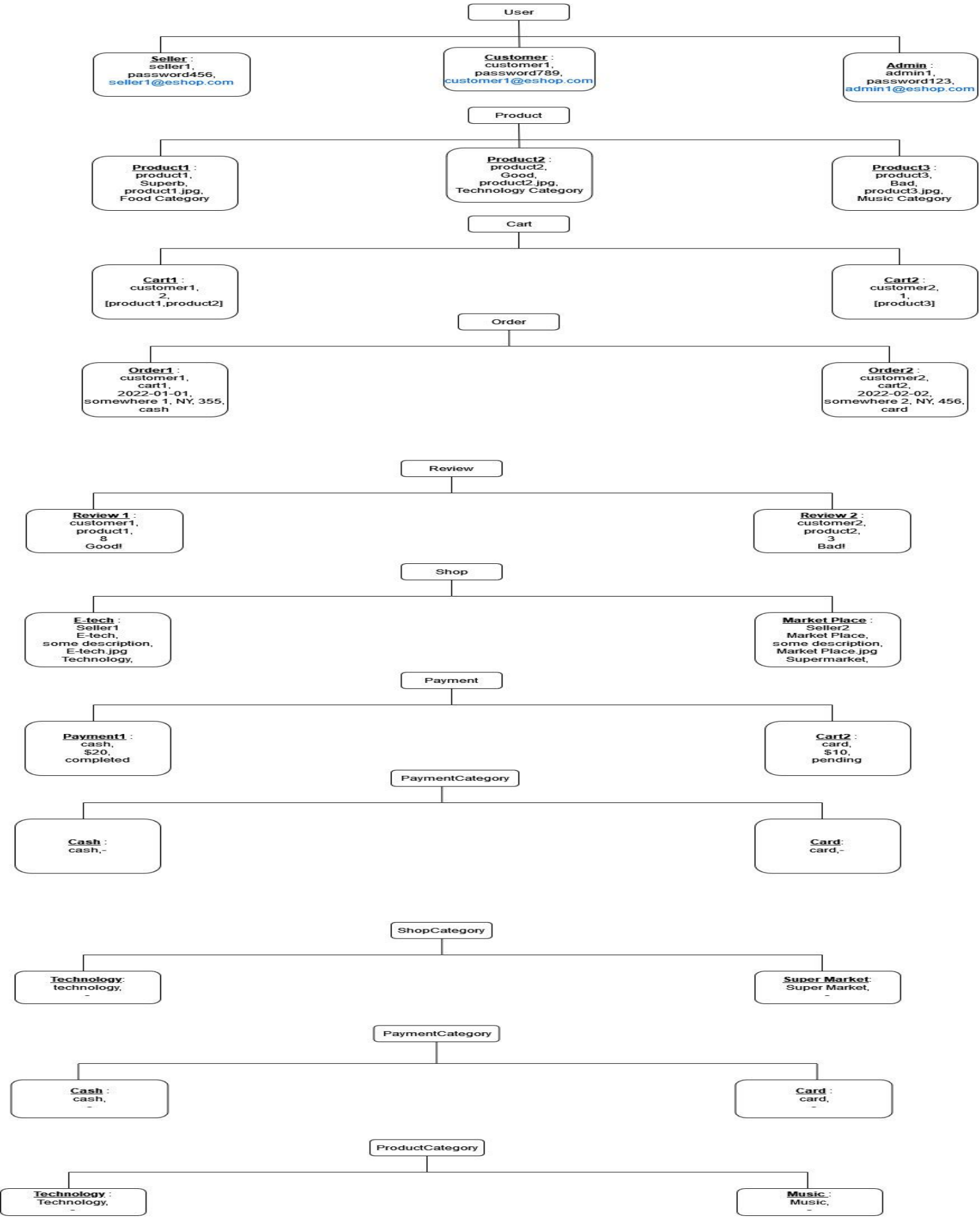
5.1.1 Use Case Diagram (3rd Version)

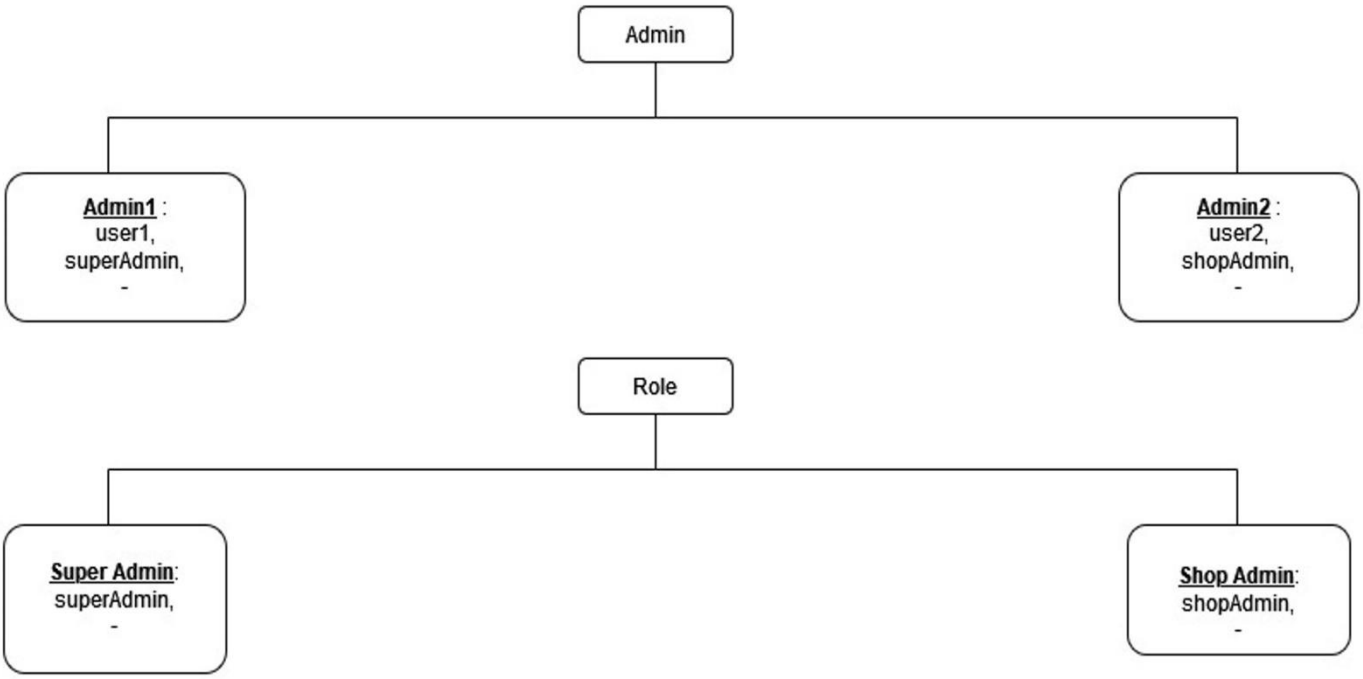


5.1.2 Class Diagram (3rd Version)

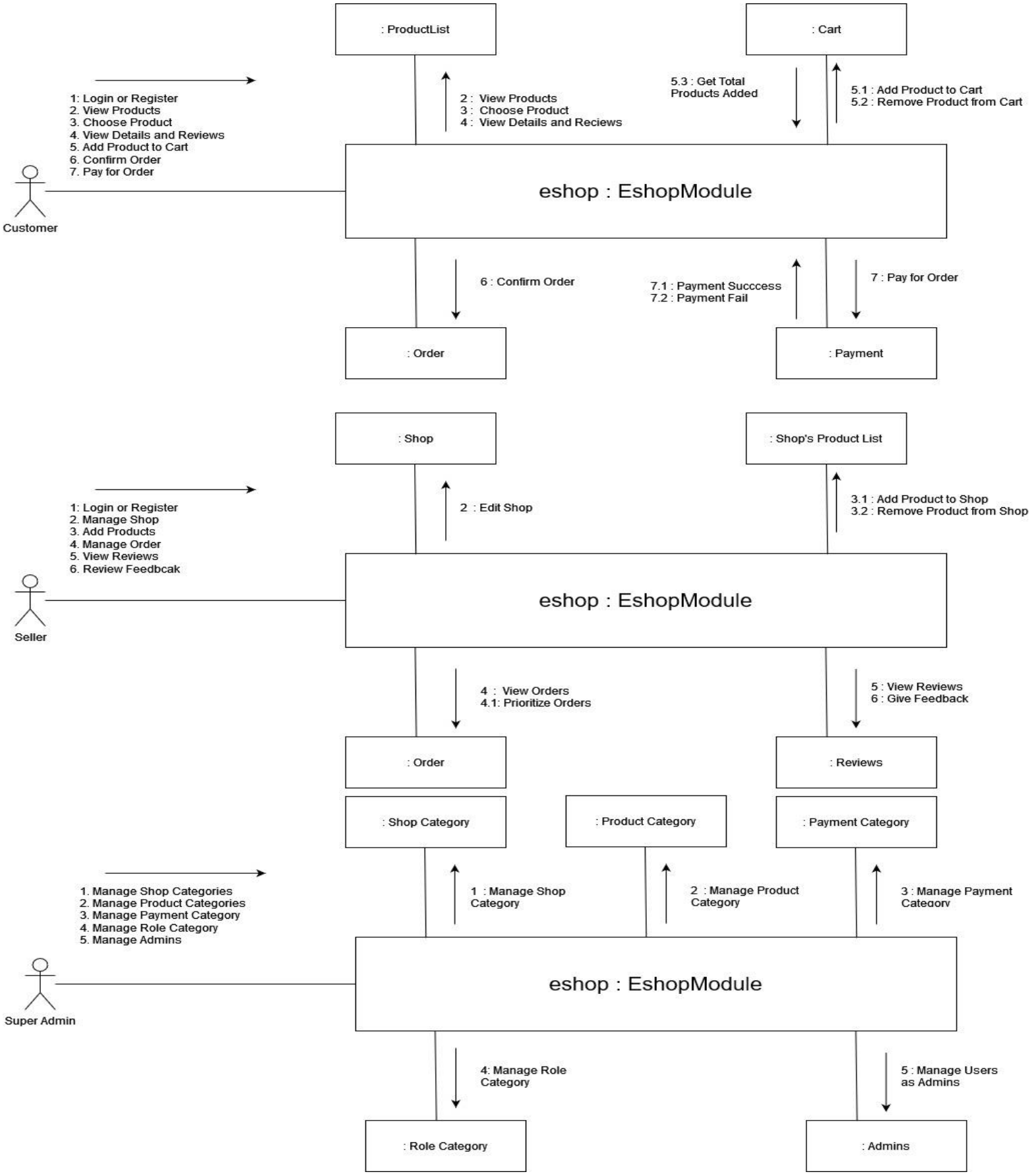


5.1.3 Object Diagram (2nd Version)

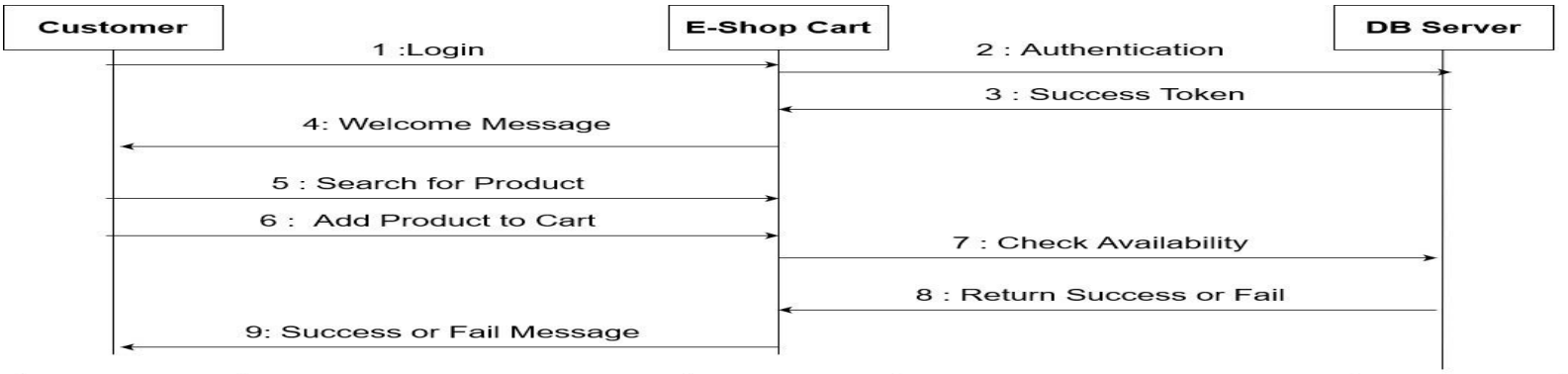


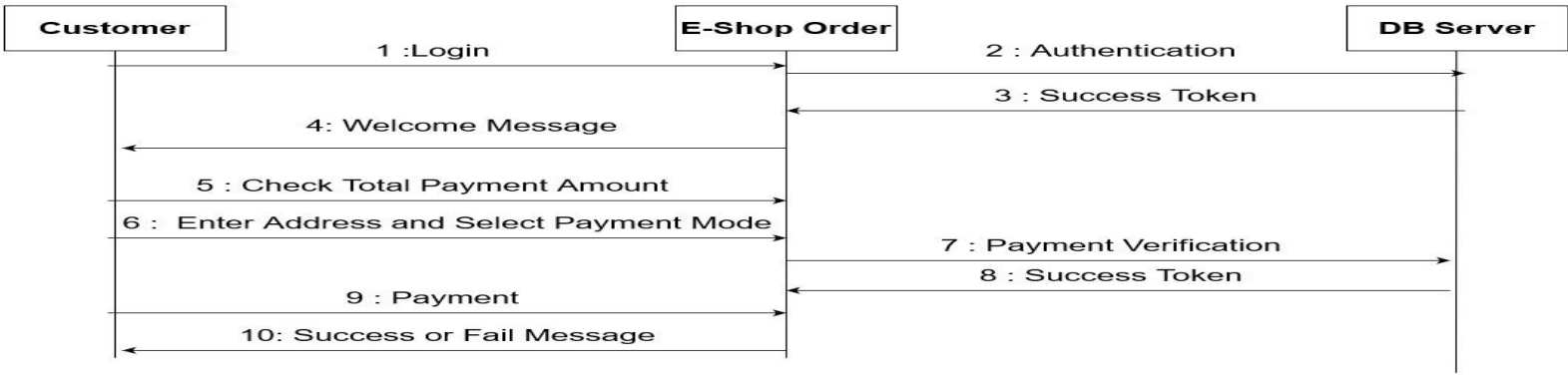

Admin
Admin1 :
user1,
superAdmin,
-
Admin2 :
user2,
shopAdmin,
-
Role
Super Admin:
superAdmin,
-
Shop Admin:
shopAdmin,
-

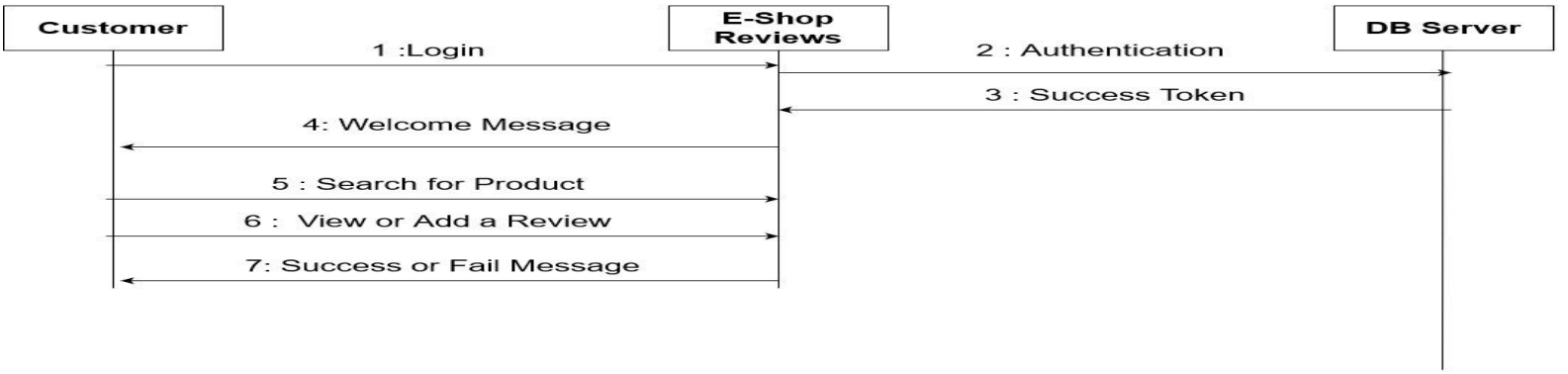
5.1.4 Collaboration or Communication Diagram (2nd Version)

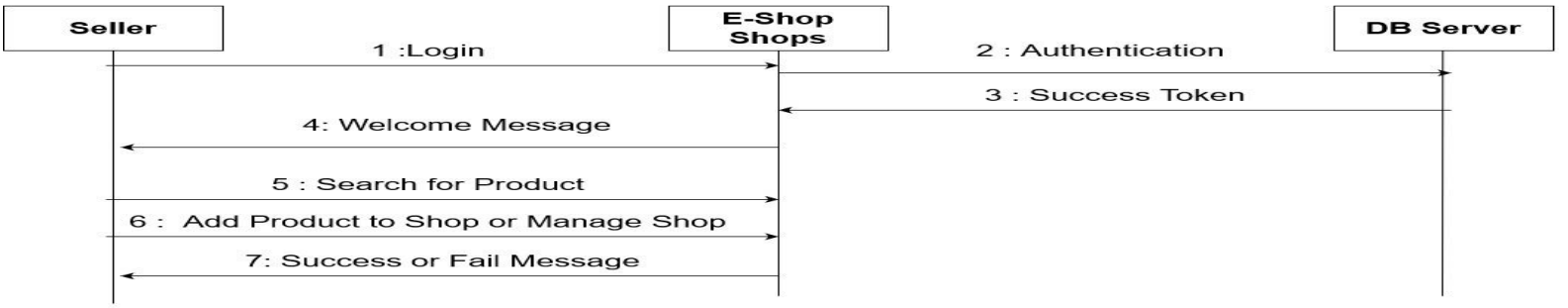


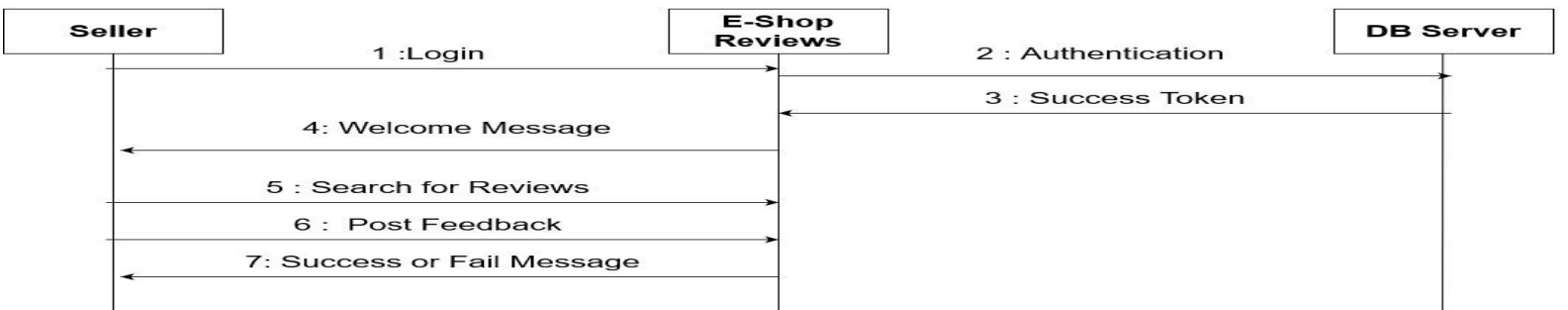
5.1.5 Sequence Diagram (2nd Version)

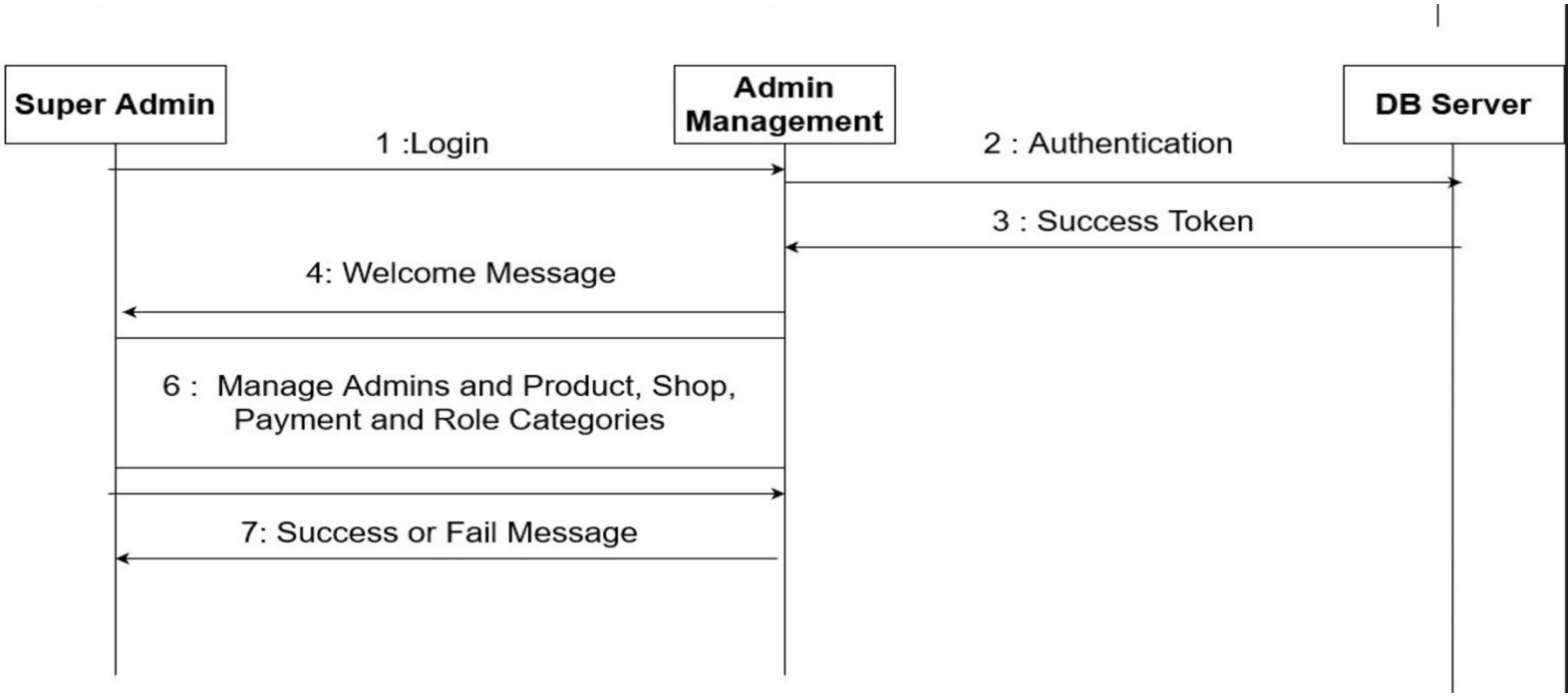




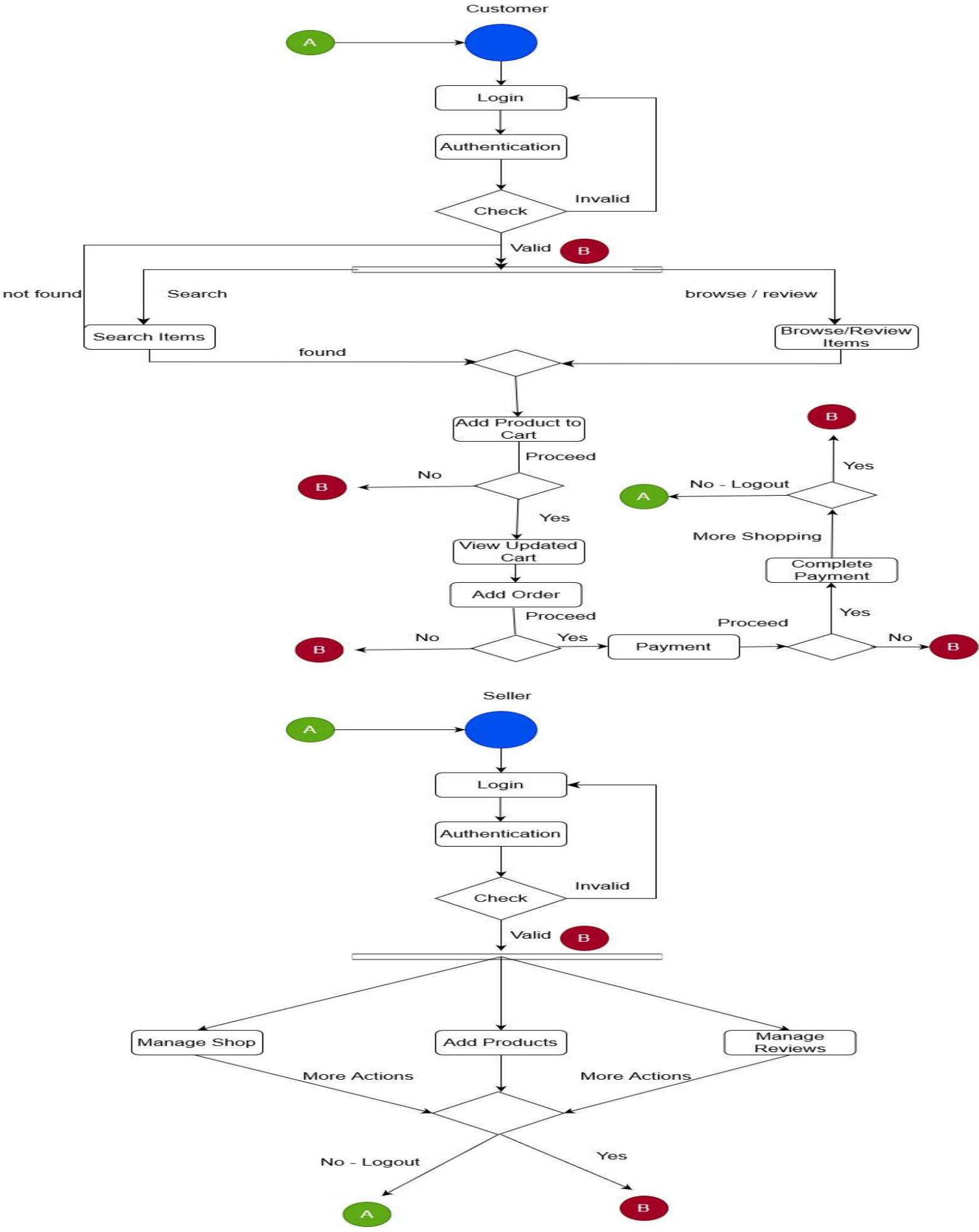


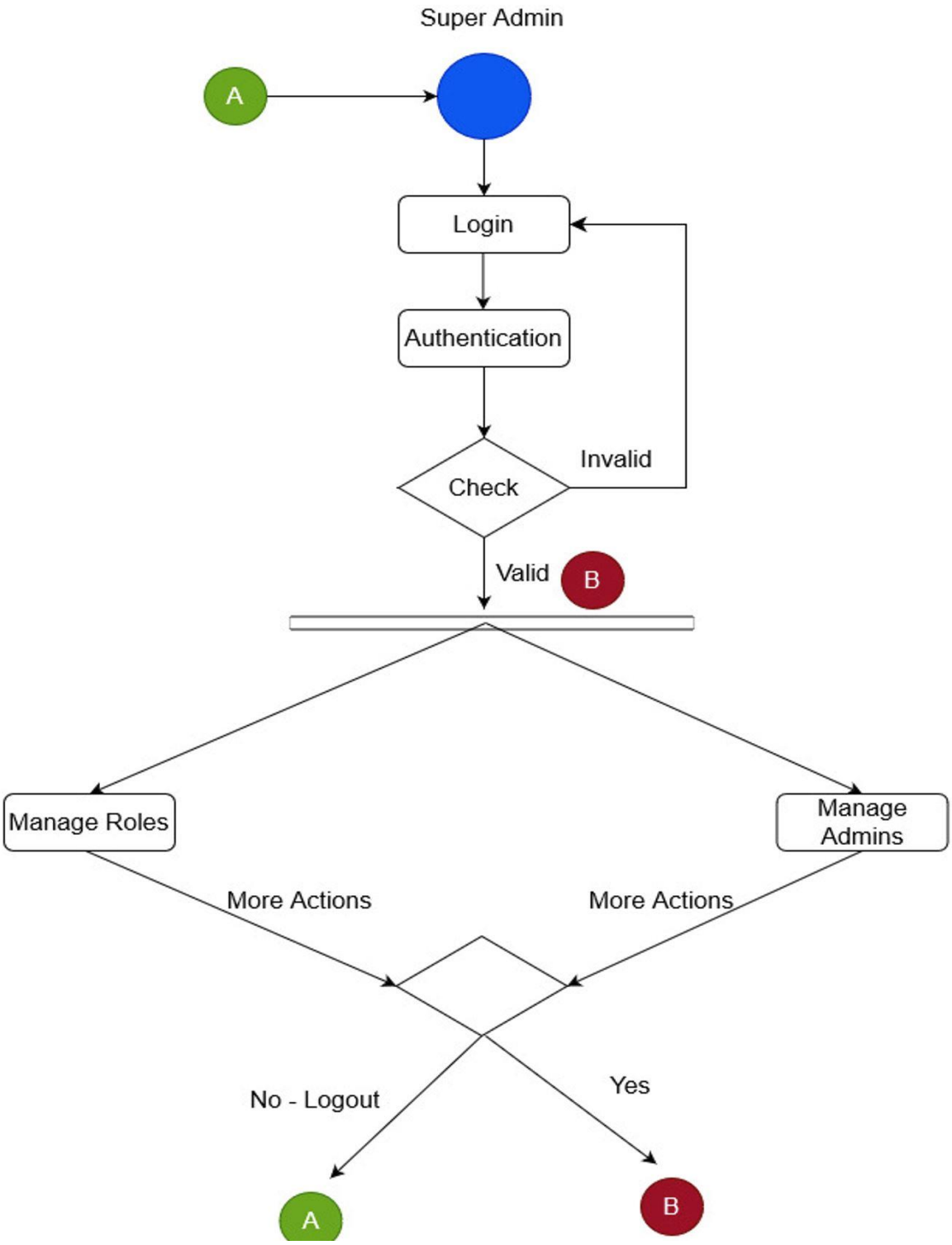




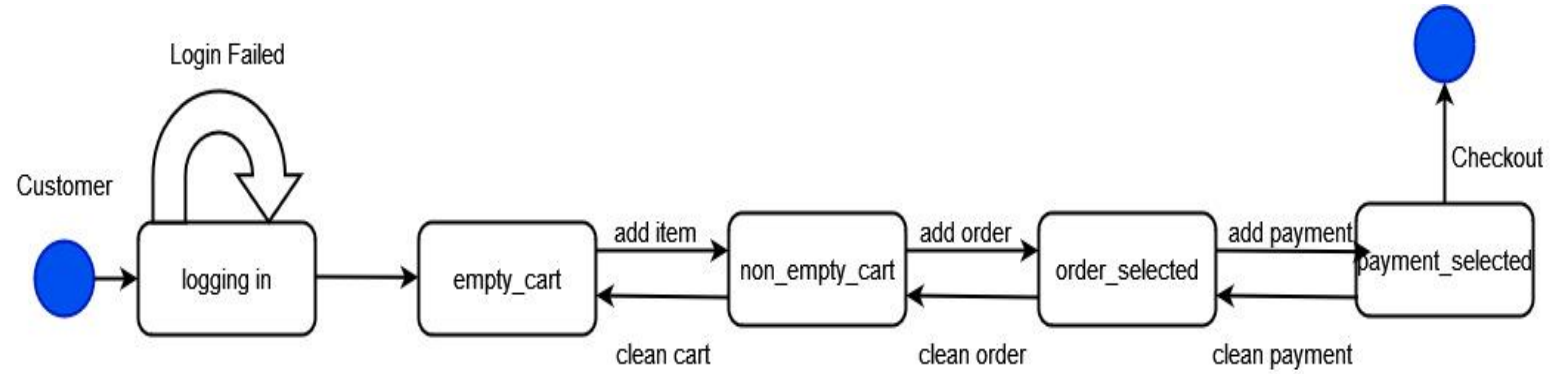

Super Admin
Admin Management
DB Server
1 :Login
2 : Authentication
3 : Success Token
4: Welcome Message
6 : Manage Admins and Product, Shop, Payment and Role Categories
7: Success or Fail Message

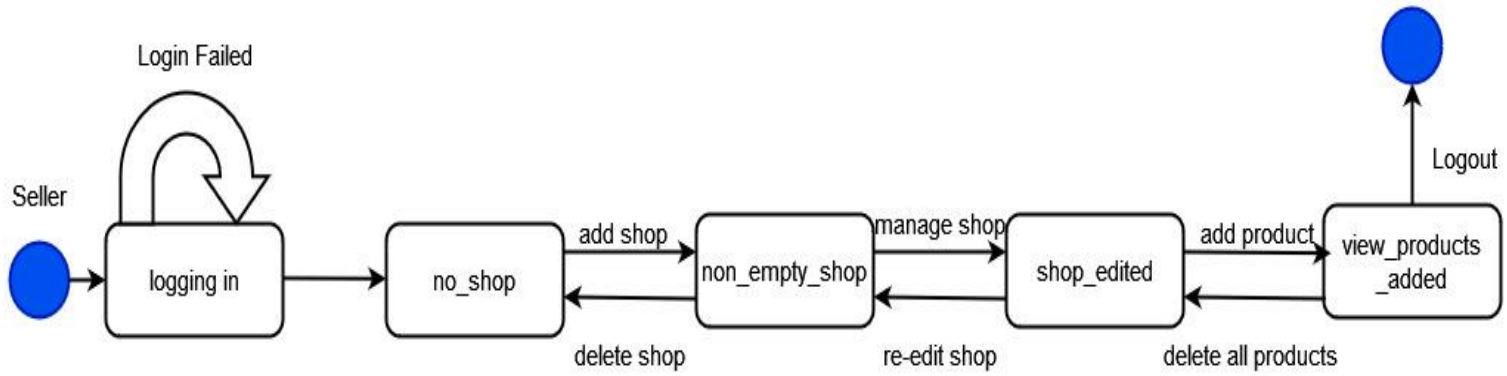
5.1.6 Activity Diagram (2nd Version)

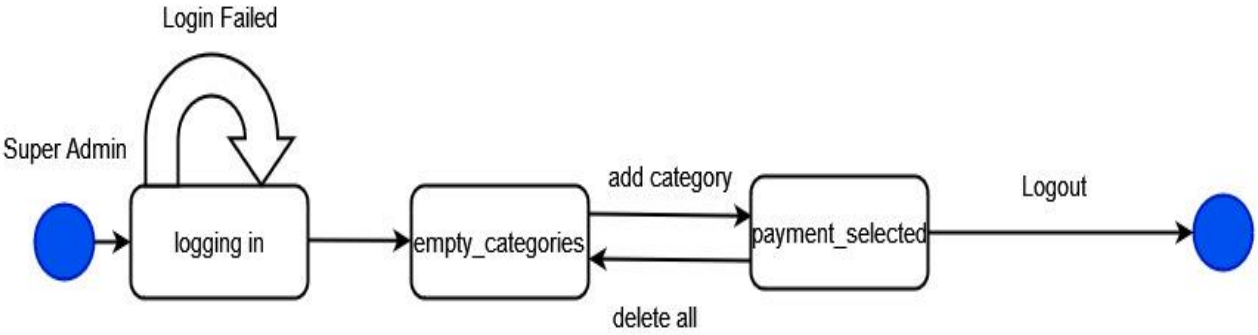



Super Admin
A
Login
Authentication
Check
Invalid
Valid
B
Manage Roles
Manage Admins
More Actions
More Actions
No - Logout
Yes
A
B

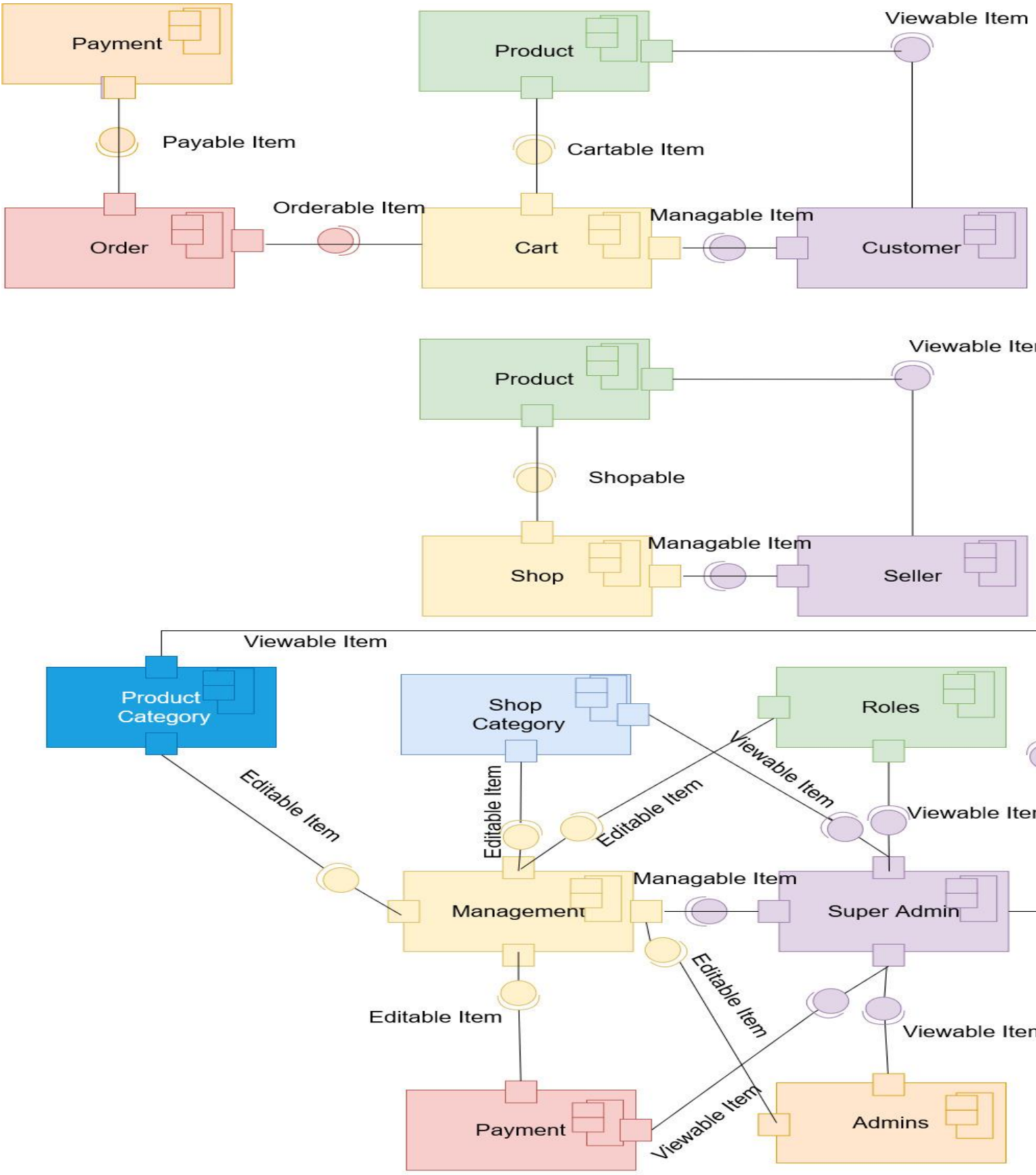
5.1.7 State Chart Diagram (2nd Version)







5.1.8 Component Diagram (2nd Version)



5.1.9 Deployment Diagram (2nd Version)