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.

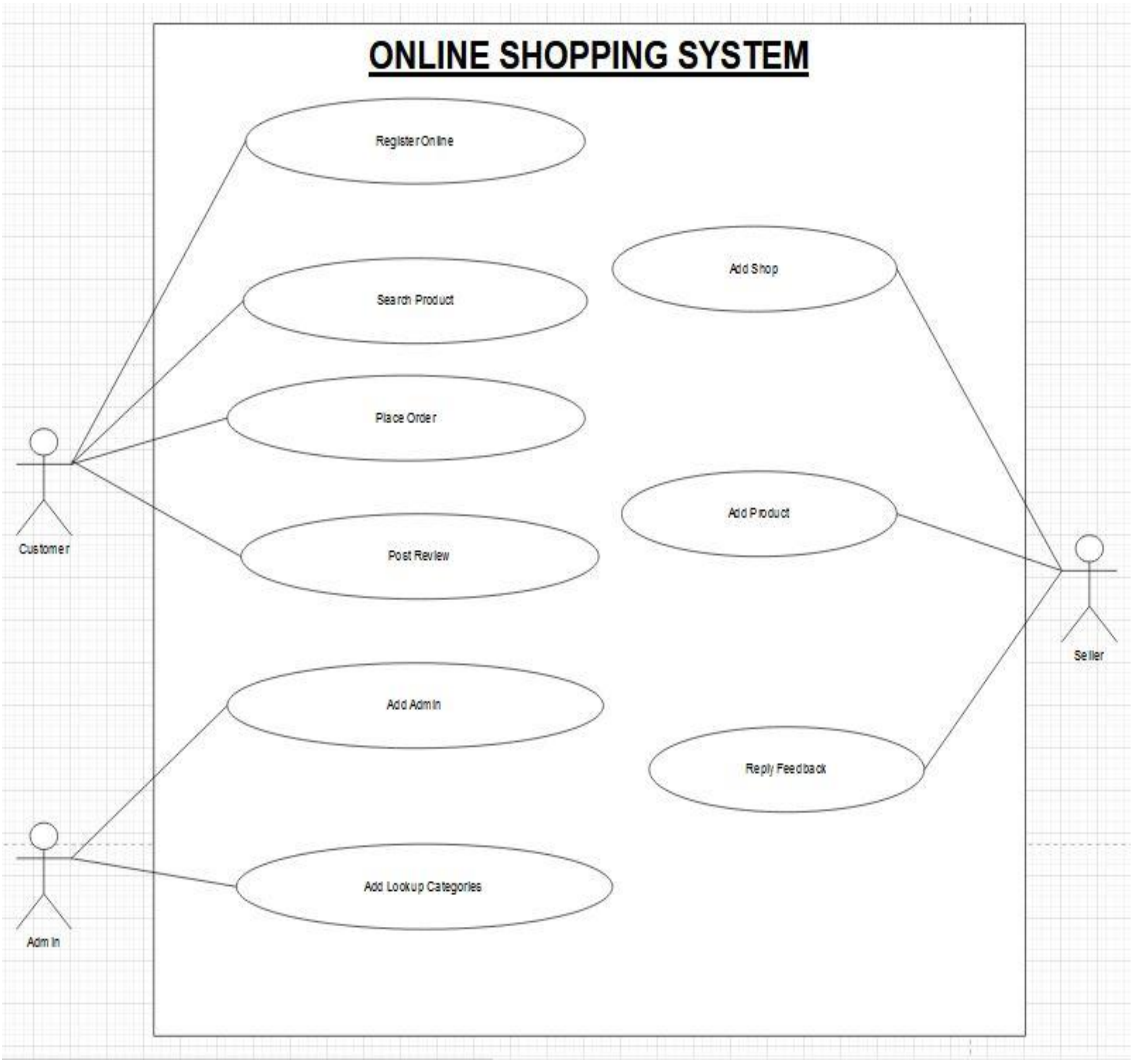
Objects:

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.

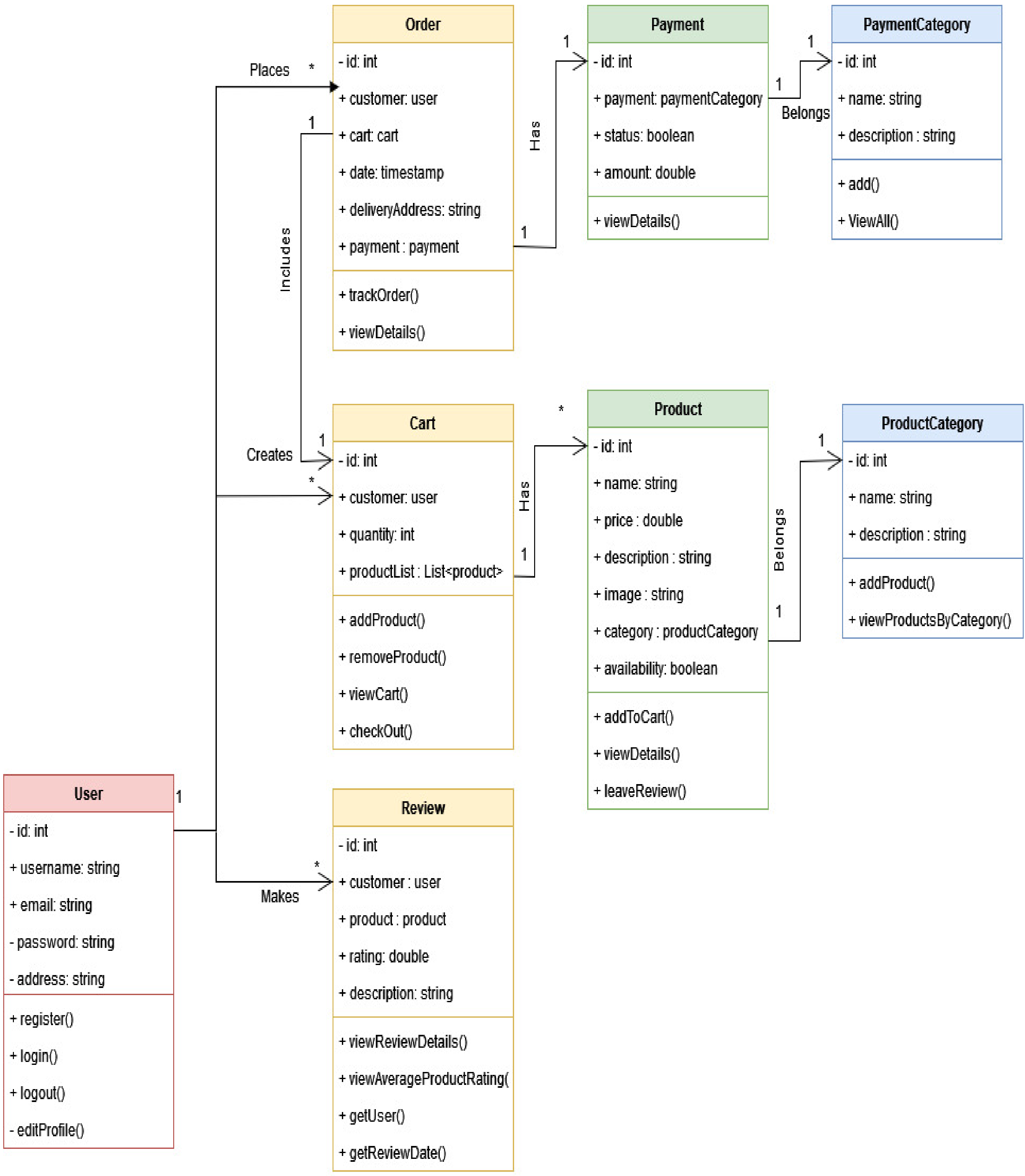
Relationships:

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.

3.2.1 Use Case Diagram (1st Version)



3.2.2 Class Diagram (1st Version)


Order
- id: int
+ customer: user
+ cart: cart
+ date: timestamp
+ deliveryAddress: string
+ payment : payment
+ trackOrder()
+ viewDetails()
Payment
- id: int
+ payment: paymentCategory
+ status: boolean
+ amount: double
+ viewDetails()
PaymentCategory
- id: int
+ name: string
+ description : string
+ add()
+ ViewAll()
Cart
- id: int
+ customer: user
+ quantity: int
+ productList : List<product>
+ addProduct()
+ removeProduct()
+ viewCart()
+ checkOut()
Product
- id: int
+ name: string
+ price : double
+ description : string
+ image : string
+ category : productCategory
+ availability: boolean
+ addToCart()
+ viewDetails()
+ leaveReview()
ProductCategory
- id: int
+ name: string
+ description : string
+ addProduct()
+ viewProductsByCategory()
User
- id: int
+ username: string
+ email: string
- password: string
- address: string
+ register()
+ login()
+ logout()
- editProfile()
Review
- id: int
+ customer : user
+ product : product
+ rating: double
+ description: string
+ viewReviewDetails()
+ viewAverageProductRating(
+ getUser()
+ getReviewDate()
Places
Includes
Creates
Makes
Has
Belongs
1
*