Online Book Store Application - Software Design Document

Welcome to our online book store!

Architectural Overview

- The Online Book Store Application follows a client-server architecture, with a frontend implemented using React for the client-side interface and a backend implemented using Spring Boot for server-side logic and data management.

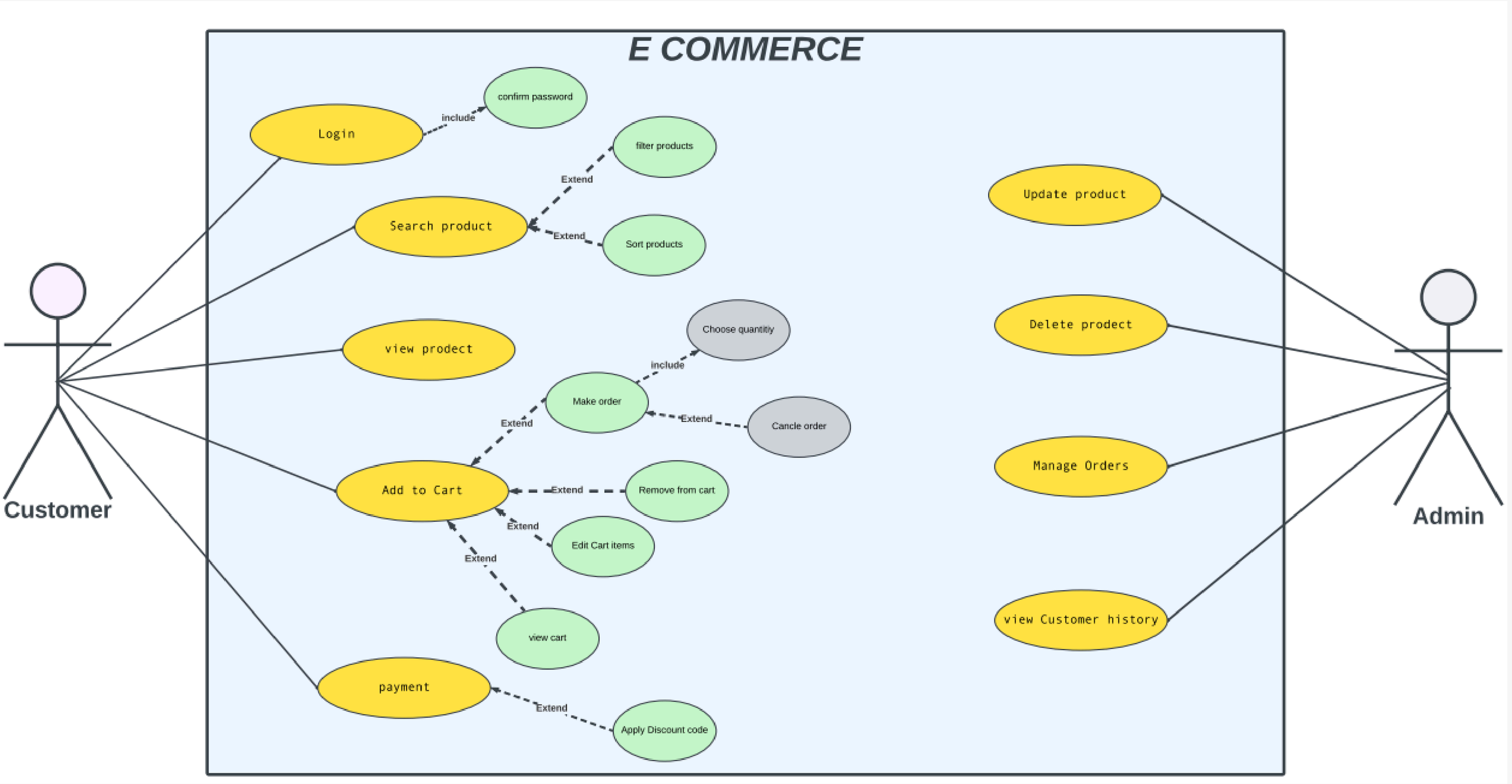
- Communication between the frontend and backend will be facilitated via RESTful API endpoints.

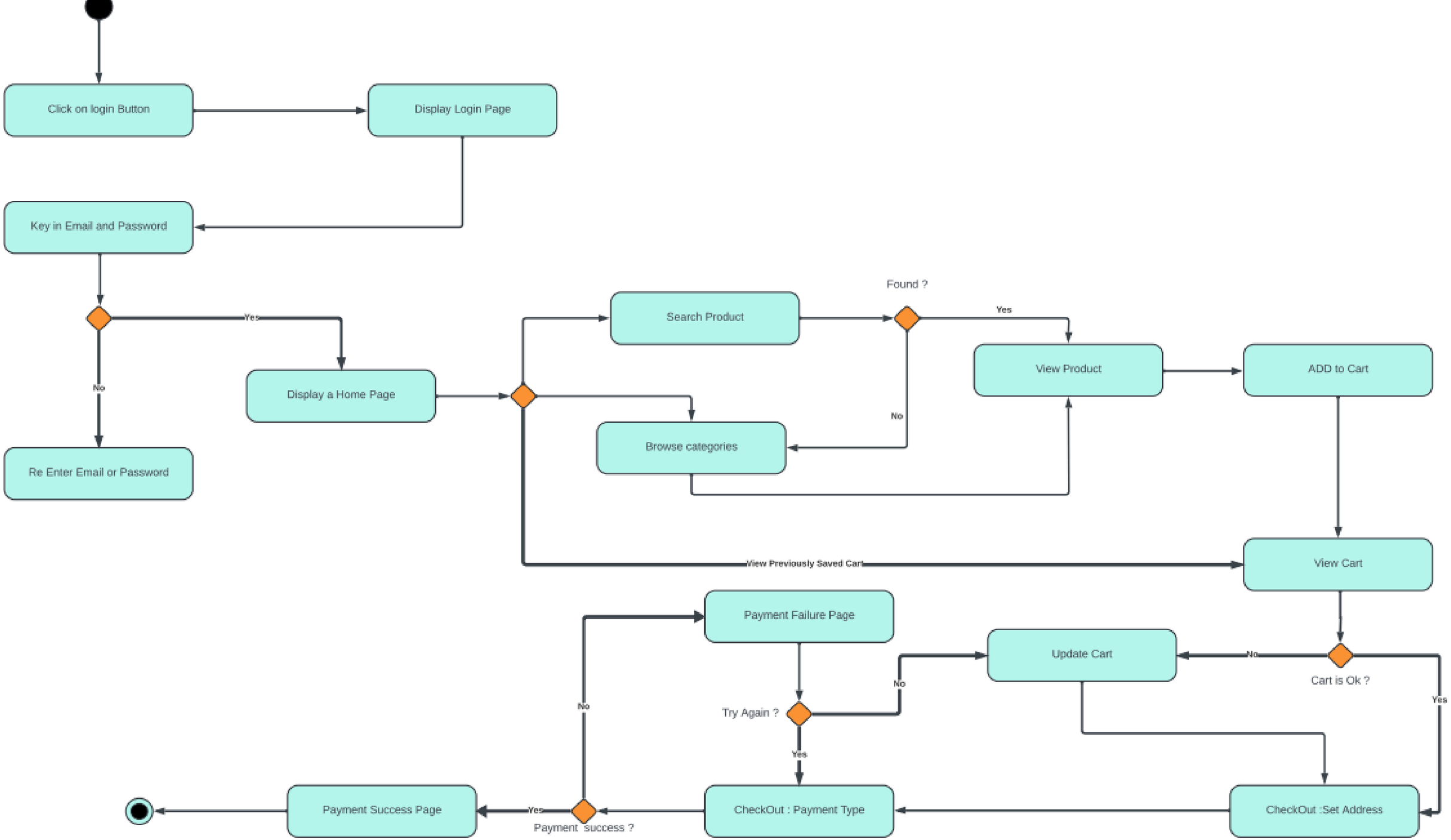
Frontend Design

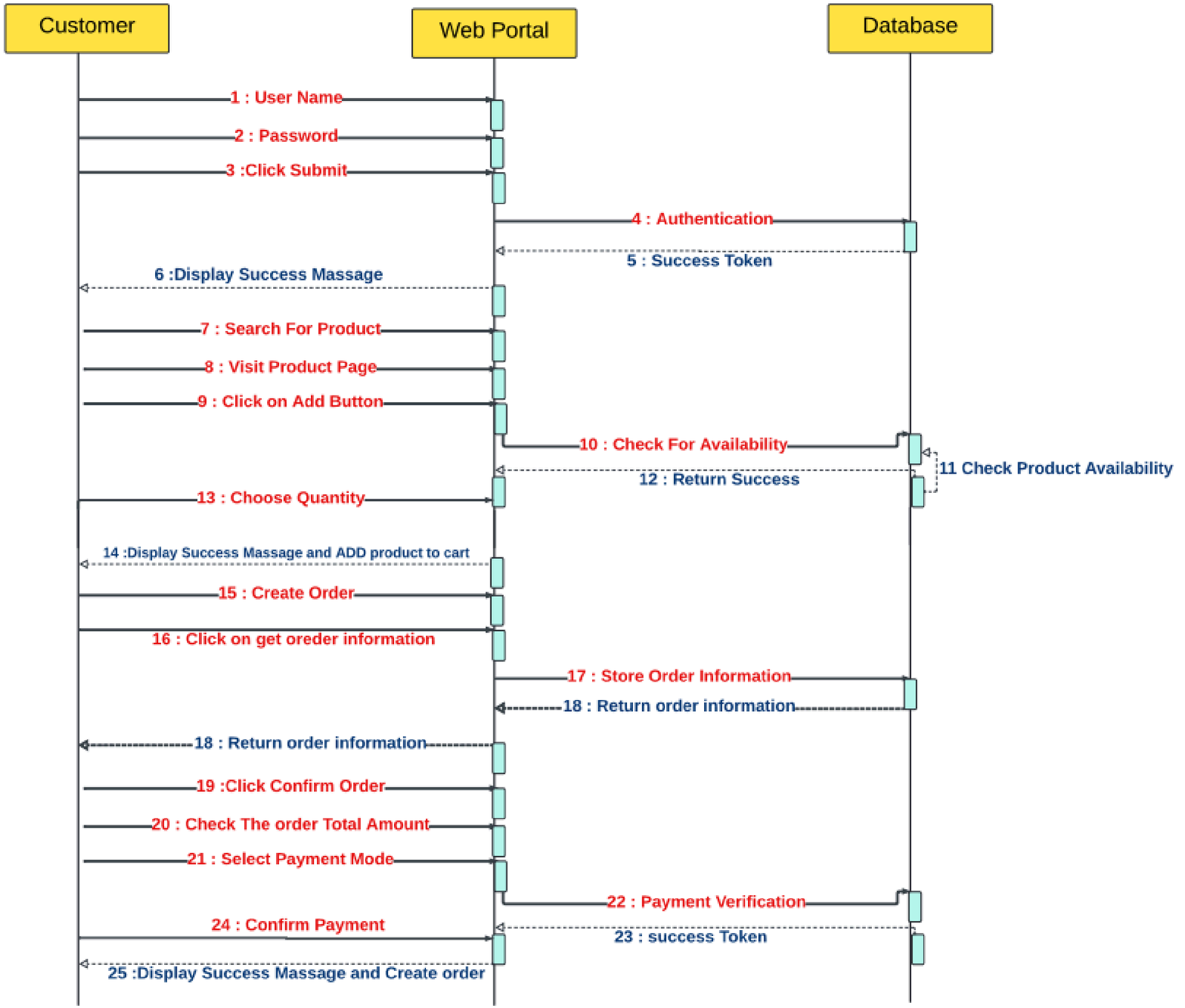
- Technology Stack: The frontend will be developed using React, a JavaScript library for building user interfaces.
- Component Structure: The application will be organized into reusable components for UI elements such as navigation bar, book listings, shopping cart, etc.
- Routing: React Router will be used for client-side routing to navigate between different pages of the application.
- State Management: Redux will be used for managing application state, including user authentication, shopping cart contents, and other global data.

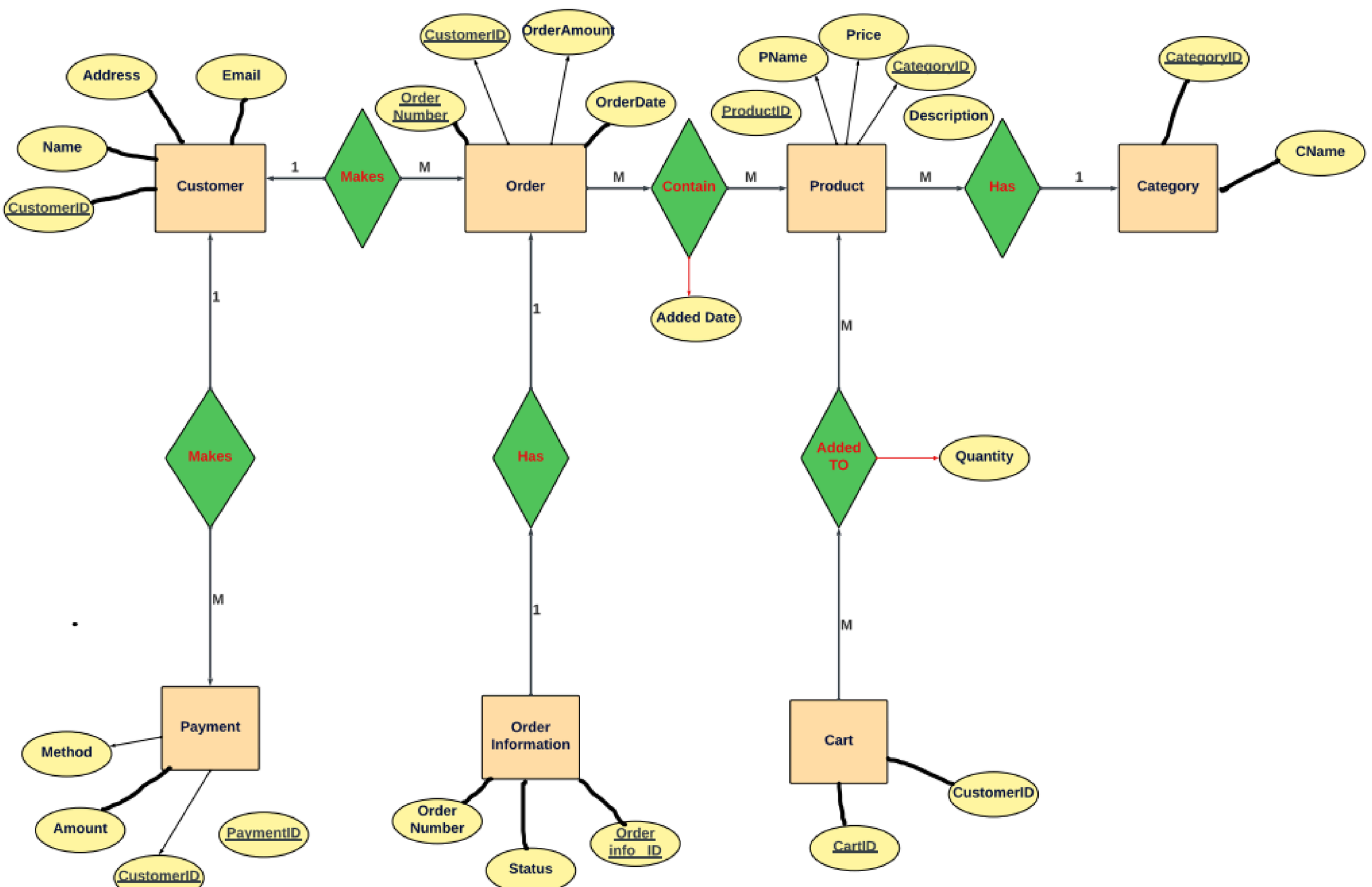
Backend Design

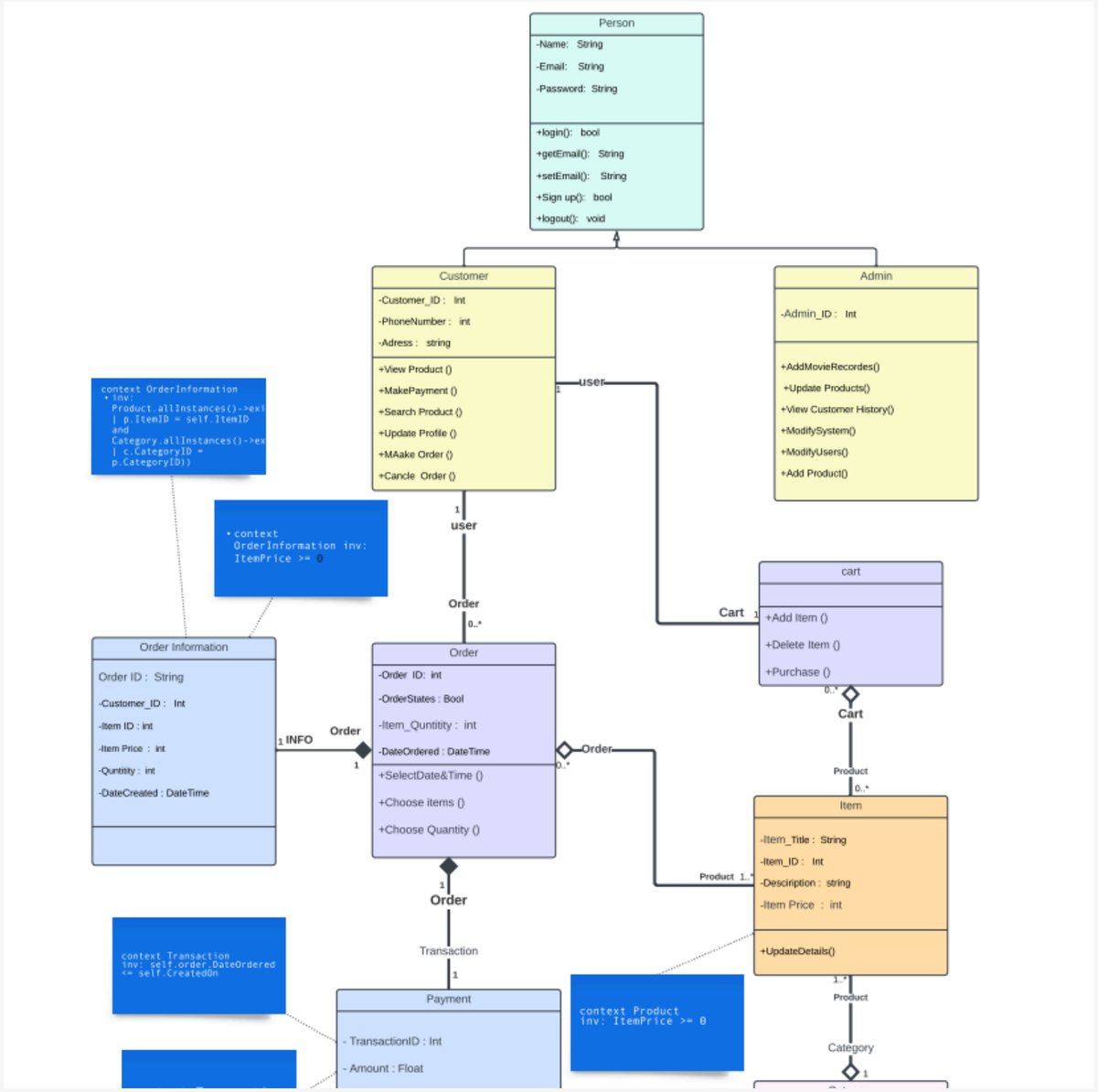
- Technology Stack: The backend will be developed using Spring Boot, a Java-based framework for building web applications.
- RESTful API Design: The backend will expose RESTful endpoints to handle CRUD (Create, Read, Update, Delete) operations for resources such as books, users, and orders.
- Data Persistence: Data will be stored in a relational database (e.g., MySQL) using Spring Data JPA for object-relational mapping.
- Security: Authentication and authorization will be implemented using Spring Security, with JSON Web Tokens (JWT) for token-based authentication.
- Validation: Input validation will be enforced using Hibernate Validator to ensure data integrity and security.

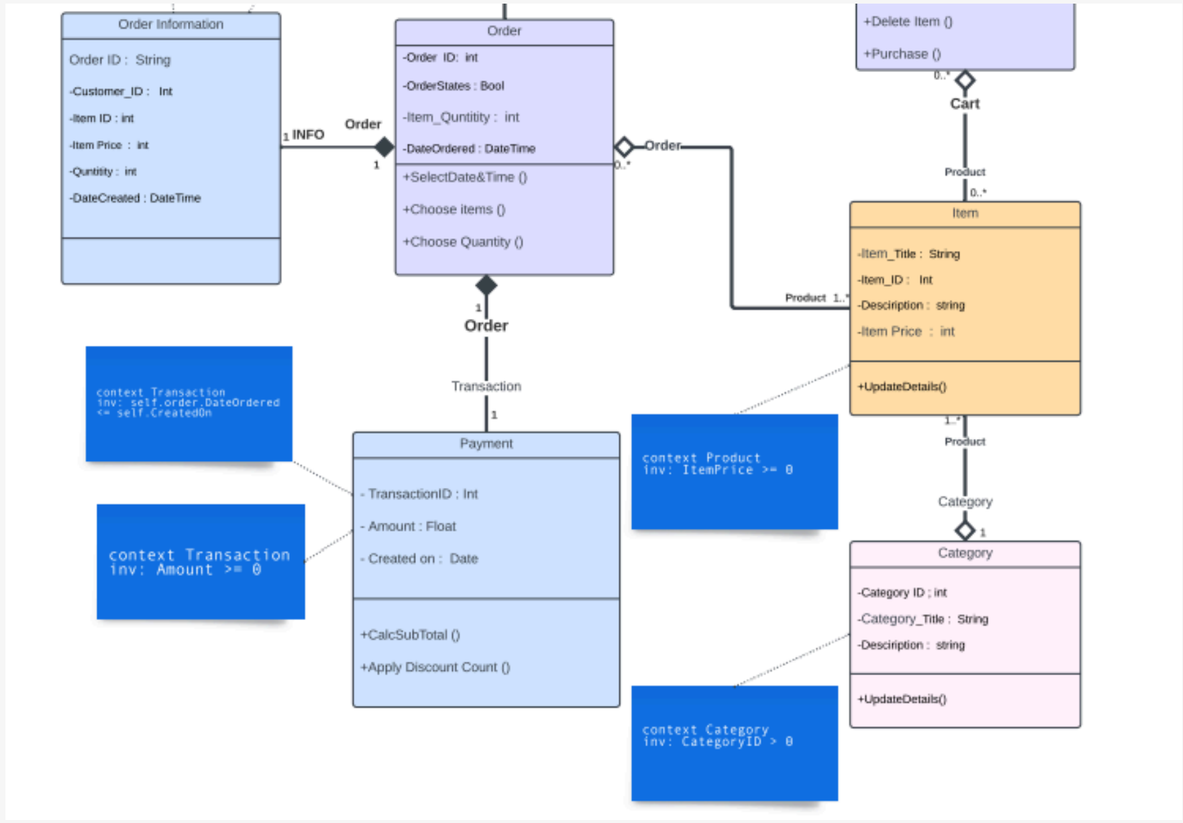

E COMMERCE
Login
confirm password
include
Search product
filter products
Extend
Sort products
Extend
view prodect
Add to Cart
Choose quantity
include
Make order
Extend
Cancle order
Extend
Extend
Remove from cart
Extend
Edit Cart items
Extend
view cart
payment
Extend
Apply Discount code
Update product
Delete prodect
Manage Orders
view Customer history
Customer
Admin


Click on login Button
Display Login Page
Key in Email and Password
Yes
No
Display a Home Page
Re Enter Email or Password
Search Product
Found ?
Yes
No
Browse categories
View Product
ADD to Cart
View Previously Saved Cart
View Cart
Payment Failure Page
Update Cart
No
Cart is Ok ?
Yes
No
Try Again ?
Yes
Payment Success Page
Yes
Payment success ?
CheckOut : Payment Type
CheckOut :Set Address


Customer
Web Portal
Database
1 : User Name
2 : Password
3 :Click Submit
4 : Authentication
5 : Success Token
6 :Display Success Massage
7 : Search For Product
8 : Visit Product Page
9 : Click on Add Button
10 : Check For Availability
11 Check Product Availability
12 : Return Success
13 : Choose Quantity
14 :Display Success Massage and ADD product to cart
15 : Create Order
16 : Click on get oreder information
17 : Store Order Information
18 : Return order information
18 : Return order information
19 :Click Confirm Order
20 : Check The order Total Amount
21 : Select Payment Mode
22 : Payment Verification
23 : success Token
24 : Confirm Payment
25 :Display Success Massage and Create order


Address
Email
Name
CustomerID
Customer
1
Makes
M
CustomerID
OrderAmount
Order Number
OrderDate
Order
M
Contain
M
Added Date
PName
Price
ProductID
CategoryID
Description
Product
M
Has
1
CategoryID
Category
CName
1
Makes
M
Payment
Method
Amount
PaymentID
CustomerID
1
Has
1
Order Information
Order Number
Status
Order info ID
M
Added TO
Quantity
M
Cart
CustomerID
CartID


Person
-Name: String
-Email: String
-Password: String
+login(): bool
+getEmail(): String
+setEmail(): String
+Sign up(): bool
+logout(): void
Customer
-Customer_ID : Int
-PhoneNumber : int
-Adress : string
+View Product ()
+MakePayment ()
+Search Product ()
+Update Profile ()
+MAake Order ()
+Cancle Order ()
Admin
-Admin_ID : Int
+AddMovieRecordes()
+Update Products()
+View Customer History()
+ModifySystem()
+ModifyUsers()
+Add Product()
user
context OrderInformation inv: ItemPrice >= 0
cart
+Add Item ()
+Delete Item ()
+Purchase ()
Cart
Order Information
Order ID : String
-Customer_ID : Int
-Item ID : int
-Item Price : int
-Quntity : int
-DateCreated : DateTime
INFO
Order
-Order ID: int
-OrderStates : Bool
-Item_Quntity : int
-DateOrdered : DateTime
+SelectDate&Time ()
+Choose items ()
+Choose Quantity ()
Product
Item
-Item_Title : String
-Item_ID : Int
-Description : string
-Item Price : int
+UpdateDetails()
Transaction
context Transaction inv: self.order.DateOrdered <= self.CreatedOn
Payment
- TransactionID : Int
- Amount : Float
context Product inv: ItemPrice >= 0
Category


Order Information
Order ID : String
-Customer_ID : Int
-Item ID : int
-Item Price : int
-Quntity : int
-DateCreated : DateTime
INFO
Order
-Order ID: int
-OrderStates : Bool
-Item_Quntity : int
-DateOrdered : DateTime
+SelectDate&Time ()
+Choose items ()
+Choose Quantity ()
+Delete Item ()
+Purchase ()
Cart
Product
Item
-Item_Title : String
-Item_ID : Int
-Description : string
-Item Price : int
+UpdateDetails()
context Transaction inv: self.order.DateOrdered <= self.CreatedOn
Transaction
Payment
- TransactionID : Int
- Amount : Float
- Created on : Date
+CalcSubTotal ()
+Apply Discount Count ()
context Transaction inv: Amount >= 0
context Product inv: ItemPrice >= 0
Category
-Category ID : int
-Category_Title : String
-Description : string
+UpdateDetails()
context Category inv: CategoryID > 0

Shop now for best books!