Banking Application Use Case

&

Online Shopping Class Diagram

REPORT

Submitted to :-

Supta Philip Richard

Senior Lecturer

City University

Submitted By :-

Fajlul Karim Chowdhury

171442505

CSE 44th Batch

City University

Contents

1. Banking Application Use Cases

From below figure it will easy to understand A simple banking application use case.
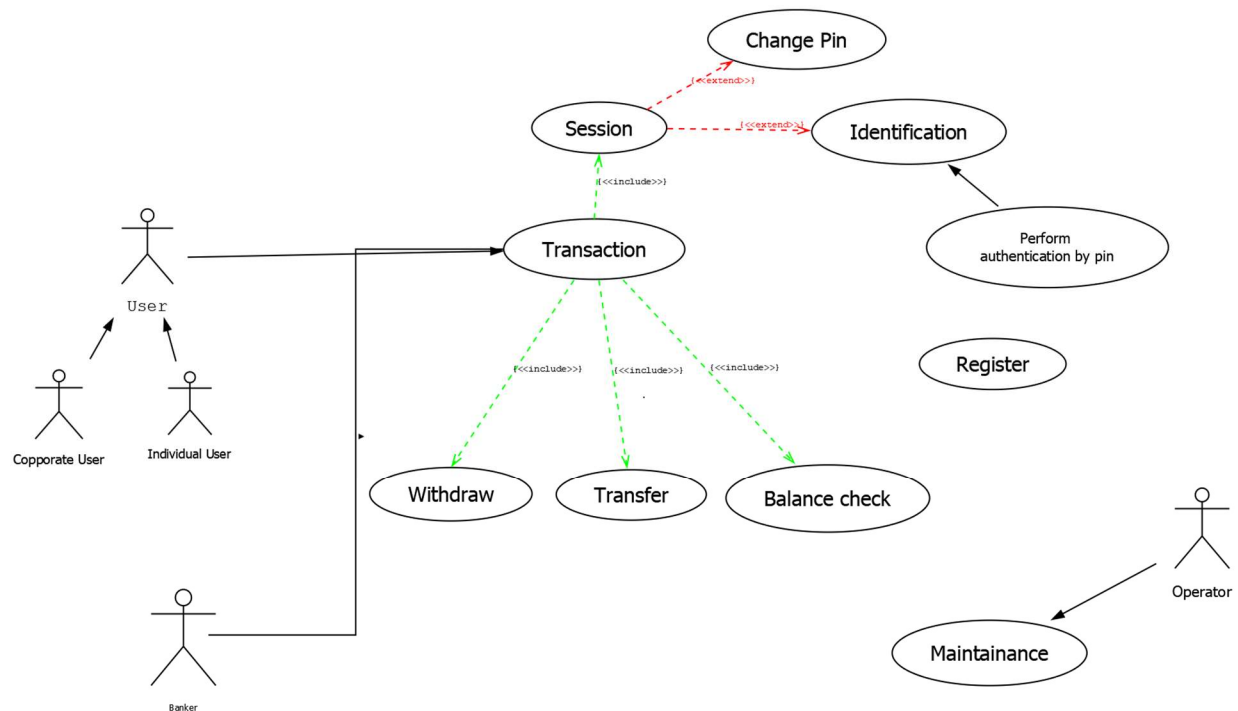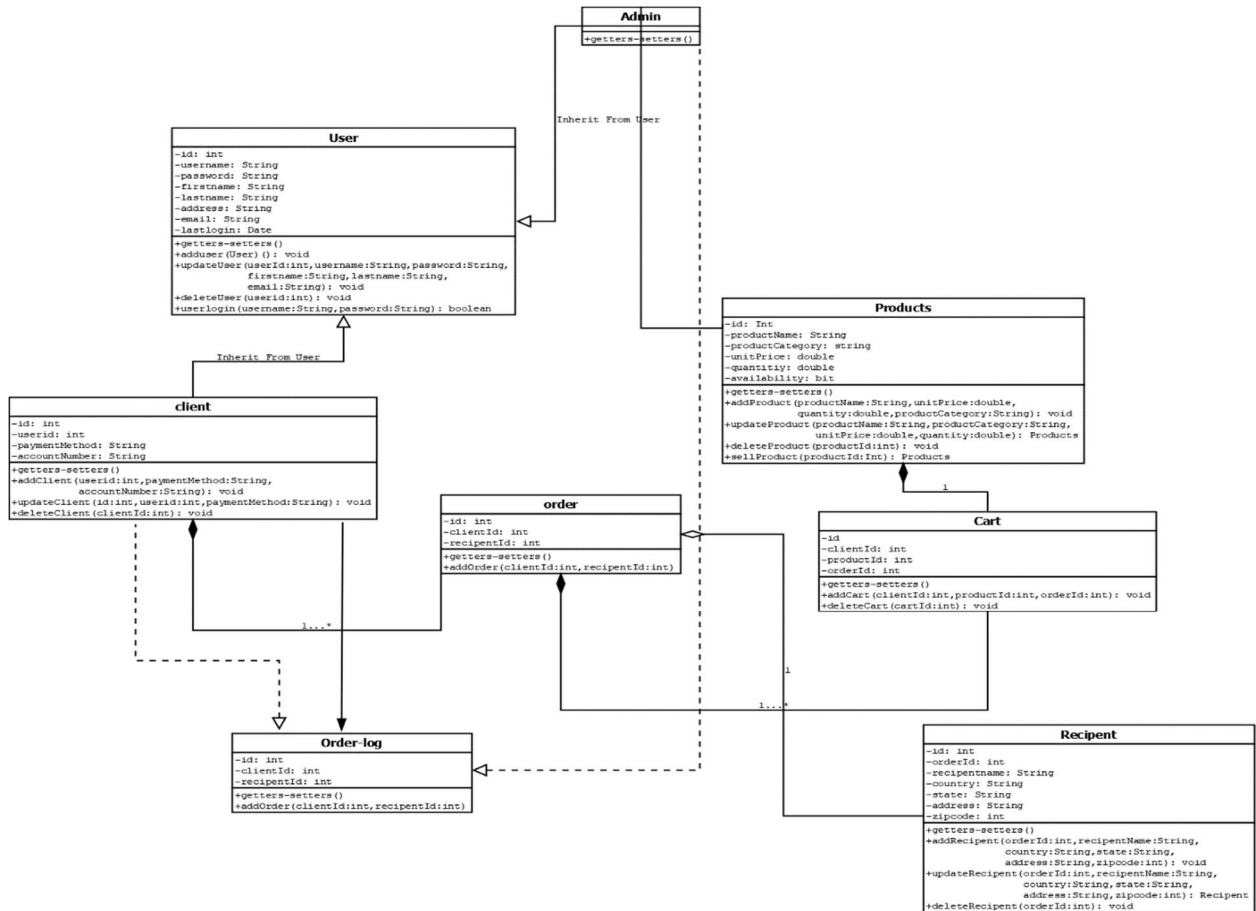

Figure 01 :- Banking Application Use Case Diagram

a. **Login :** Each Customer/ User must login before executing. The bank system verifies the user authentication to login.
b. **Session :** If the authentication is verified by the system then session is created with the application server between Customers, which enables user to perform other operation into system.
c. **Change pin:** A user able to change of his own password/pin. The password/pin changing is validated by the application server Protocols.
d. **Check Balance :** A customer/user can check his account balance in two type of operation.
    i. **Current Balance :** Customer can their current Balance.
    ii. **Statement :** Or they can get a statement with transaction Details for a limited duration.
e. **Withdraw :** Customers is able to withdraw their money in cash if the transaction is validated from bank.

f. **Fund Transfer :** Fund Transfer is one of the most important part of this application customer have to give all the required information's about the transaction then these

Admin
+getters-setters()

Inherit From User

User
-id: int
-username: String
-password: String
-firstname: String
-lastname: String
-address: String
-email: String
-lastlogin: Date
+getters-setters()
+adduser(User)(): void
+updateUser(userId:int,username:String,password:String,
        firstname:String,lastname:String,
        email:String): void
+deleteUser(userid:int): void
+userlogin(username:String,password:String): boolean

Inherit From User

client
-id: int
-userid: int
-paymentMethod: String
-accountNumber: String
+getters-setters()
+addClient(userid:int,paymentMethod:String,
        accountNumber:String): void
+updateClient(id:int,userid:int,paymentMethod:String): void
+deleteClient(clientId:int): void

Products
-id: Int
-productName: String
-productCategory: string
-unitPrice: double
-quantity: double
-availability: bit
+getters-setters()
+addProduct(productName:String,unitPrice:double,
        quantity:double,productCategory:String): void
+updateProduct(productName:String,productCategory:String,
        unitPrice:double,quantity:double): Products
+deleteProduct(productId:int): void
+sellProduct(productId:Int): Products

order
-id: int
-clientId: int
-recipentId: int
+getters-setters()
+addOrder(clientId:int,recipentId:int)

Cart
-id
-clientId: int
-productId: int
-orderId: int
+getters-setters()
+addCart(clientId:int,productId:int,orderId:int): void
+deleteCart(cartId:int): void

Order-log
-id: int
-clientId: int
-recipentId: int
+getters-setters()
+addOrder(clientId:int,recipentId:int)

Recipent
-id: int
-orderId: int
-recipentname: String
-country: String
-state: String
-address: String
-zipcode: int
+getters-setters()
+addRecipent(orderId:int,recipentName:String,
        country:String,state:String,
        address:String,zipcode:int): void
+updateRecipent(orderId:int,recipentName:String,
        country:String,state:String,
        address:String,zipcode:int): Recipent
+deleteRecipent(orderId:int): void

information's are validated by the bank system.

2. Online Shopping Class Diagram

a. **User:** The user class is the super class for both (Admin class) & (Customer Class). All the basic information which is common between an Admin and Customer are described into this class.

b. **Customer :** This class inherits all the properties from the (User) Class. Customers are basically allowed to purchase and mange their profile only.

c. **Admin :** This class also inherits all the properties from the (User) Class. This User is permitted to manage all users, Products and Orders into the system.

d. **Order :** Order Class basically are tightly coupled with the (Customer Class). The (Order) Class and The (Customer Class) have a composition relation which ensures that every order must have a customer before the Order is created.

e. **Cart :** The cart class simply maps the relation between each product to specific Order. Cart Class is tightly coupled with both (Order) and (Products) Class. Both relation is in many to many order.

f. **Products :** This Class Is an independent class. It have a simple association relation with Admin Class, So the admin is able to manage all the products.

Summary

In this Report Simply described about the Use case and Class diagram of A Bank application &
An Online Shopping system with figure.