**Bazaar**

Design Document, version 1

October 18, 2019

Group Members:

Kundan Rupakheti

Rajan Khadka

Kiran Thapa

Francisco Caceres

Group Number: 16

Table of Contents

[1. Overview Class Diagram 4](#_Toc22326025)

[2. Detailed Class Diagrams 5](#_Toc22326026)

[2.1 Data Classes 5](#_Toc22326027)

[2.1.1 User 5](#_Toc22326028)

[2.1.2 Customer 6](#_Toc22326029)

[2.1.3 Seller 7](#_Toc22326030)

[2.1.4 Administrator 8](#_Toc22326031)

[2.1.5 Product 9](#_Toc22326032)

[2.1.5 Search 10](#_Toc22326033)

[2.1.6 Shopping Cart 11](#_Toc22326034)

[2.1.7 Order 12](#_Toc22326035)

[2.1.8 Payment 13](#_Toc22326036)

[2.1.9 Order Details 14](#_Toc22326037)

[2.1.19 Complain 15](#_Toc22326038)

[3. Sequence Diagrams 16](#_Toc22326039)

[3.1 View Cart 16](#_Toc22326040)

[3.2 Make Complaint 17](#_Toc22326041)

[3.3 Check Out 18](#_Toc22326042)

[3.4 Add to Cart 19](#_Toc22326043)

[3.5 Create Account 20](#_Toc22326044)

[Appendix A. Database Design 21](#_Toc22326045)

[Appendix B. Task & Role Assignments 22](#_Toc22326046)

# 1. Overview Class Diagram

A screenshot of a cell phone

Description automatically generated

# Detailed Class Diagrams

## 2.1 Data Classes

### 2.1.1 User

|  |
| --- |
| User |
| -userId: string  -password: string  -loginStatus: string  -accountType: string |
| +\_User(string userId)  +verifyLogin(): bool |

* \_User(string userId) : Default Constructor. Establishes username (as userId ) and other values to null.
* verifyLogin(userId,: string, password: string): Retrieves the username and password from the user. Verifies if the user is logged in with loginStatus. accountType informs program whether the account is of customer or seller. After username, password, and account type are retrieved, the class returns a value based on the Boolean type.

### 2.1.2 Customer

|  |
| --- |
| Customer |
| -customerName: string  -address: string  -email: string  -userName: string  -password: string  -customerId: int  -shippingAddress: string  -accountType: string |
| +\_Customer(int customerId)  +createAccount()  +login()  +updateProfile()  +logout()  +accountInfo()  +verifyAccount(): string |

* \_Customer( int customerId): Default constructor. Establishes Customer with an ID
* createAccount ( customerName: string, address: string, email: string, userName: string, password: string, customerId: int, shippingAddress: string, accountType: string ): Constructor. It generates the account from the attributes of customerName, address, email, username, password, customerId, shippingAddress and accountType and saves it to database of our system.
* Login( userName: string, password: string, ) : The login method takes the userName and password and authenticates if the user is in the system or not.
* updateProfile(customerName: string, address: string, email: string, userName: string, password: string, customerId: int, shippingAddress: string, accountType: string): Updates the customer information if they want to change it. It would be userName, password, customerId and shippingAddress. )
* Logout() : redirects the customer to the homepage of the system once the method is run.
* verifyAccount() : This method checks whether the account is valid or invalid based on what the user created.

### 2.1.3 Seller

|  |
| --- |
| Seller |
| -SellerId: String  -AccountBalance: float  -totalAddedProduct: int |
| +\_Seller(string Sellerid, string password)  +addProduct():void  +productListed():void  +removeProduct(): void  +modifyProduct: void |

* \_Seller (string SellerId, string password) : Constructor. Used to set the default Seller account object. Uses the seller information to generate an Id for them.
* addProduct() : this method adds the product in the inventory of the system by the seller
* productListed (addProduct(): void) : It retrieve the product added on the inventory and list out all the product.
* removeProduct(totalAddedProduct: int) : This method modifies the product. It takes the parameter totalAddedProduct.
* modifyProduct() : It modifies the product based on the attributes.

### 2.1.4 Administrator

|  |
| --- |
| Administrator |
| -adminName: string  -email: string |
| +\_Admin(string adminName, string email)  +viewComplaints(adminName: string, email:string): void  + verifySellerAccount(string userId, sting email): bool |

* \_Admin (string adminName, string email): Constructor. Allows administrator to have control over functions like make complains and verify accounts.
* viewComplaints(string adminName, string email) : this method takes the attributes adminName and email to login as a Administrator and viewComplaints made by customer associated with the seller account.
* verifySellerAccount(string sellerID, sting email) : verifies seller details

### 2.1.5 Product

|  |
| --- |
| Product |
| -productName: string  -productId: int  -productUnitCost: float  -productType: string  -sellerId: string  -dateAdded: Date  -productRentPrice: float |
| +\_Product (string productName, int productid, string sellerId) |

* \_Product (string productName, int productid, string sellerId): Constructor. Used to construct Product class. Stores the information about products.

### 2.1.5 Search

|  |
| --- |
| Search |
| -productName: string  -productId: int  -productType: string  -dateAdded: date |
| +\_Search (string productName, int productId)  +searchProductByName(): void  +searchProductByType(): void  +searchProductByDateAdded(): void  +searchProductByPrice():void |

* \_Search (string productName, int productId): Constructor. Creates search class to search products in the inventory.
* searchProductByName(): Default method of searching products.
* searchProductByType(): Uses filter to search product by type.
* searchProductByDateAdded(): Uses filter to search product by date.
* searchProductByPrice(): Uses filter to search products by price.

### 2.1.6 Shopping Cart

|  |
| --- |
| Shopping Cart |
| -cartId: int  -productID: int  -quantity: int  -dateAdded: int  -customer:string |
| +\_Cart (int cartId, int productid, string customer, int dateAdded  +addCartItem(int productid, int quantity) : void  +updateQuantity(): void  +viewCartDetails(int quantity, int dateAdded, int productid): void  +checkout (quantity:int, cartId:int): void |

* \_Cart (int cartId, int productid, string customer, int dateAdded) : Constructor. Creates a cart for the customer with their appropriate customer and product information.
* addCartItem(int productid, int quantity) : Adds the item to the shopping cart based on the cartId and the productid.
* updateQuantity() : Updates the quantity in the shopping cart.
* viewCartDetails(int quantity, int dateAdded, int productid) : Displays cart details including: quantity of the current products in the shopping cart has, respective productid’s, and the respective date when each product was added to the shopping cart.
* checkout (quantity:int, cartId:int) : Directs customer to checkout the products in the shopping cart.

### 2.1.7 Order

|  |
| --- |
| Order |
| -orderId: int  -dateCreated: string  -date Shipped: string  -customerName: string  -customerId: string  -status:string  -shipping:string |
| +\_Order(string customerName, string customerId, string shipping, string status)  +placeOrder() : void |

* \_Order(string customerName, string customerId, string shipping, string status) : Constructor. Uses parameters to set member variables. Coupled with the orderDetails class, Order will succinctly keep all customer, product, and order details.
* placeOrder() : Places the order based on the added quantity on the cart.

### 2.1.8 Payment

|  |
| --- |
| Payment |
| -cardType: String  -cardName: string  -cardNumber: string  -expDate: date |
| +\_Payment(string cardType, string cardName, string cardNumber, date expDate)  +verifyPayment(): void |

* \_Payment(string cardType, string cardName, string cardNumber, date expDate) : Constructor. Passes payment details in parameters by value for later verification. Payment will then be checked against a credit-card verification system.
* verifyPayment() : Takes member variables to ensure verification is valid or not.

### 2.1.9 Order Details

|  |
| --- |
| Order Details |
| -orderId: int  -productId: int  -productName: string  -quantity:int  -productUnitCost: float  -productRentPrice: float  -rentDuration: int  -subTotal:float |
| +\_Order(float productUnitCost, float productRentPrice, int rentDuration, float subtotal)  +calcPrice(): float |

* \_Order(float productUnitCost, float productRentPrice, int rentDuration, float subtotal) : Constructor. Uses parameter variable to set member variables which will be displayed to customer on checkout.
* calcPrice() : Calculates the price based on the unit cost of the product, unit rent price and the time duration the order is processed. Thus, it calculates the total price of the product while renting.

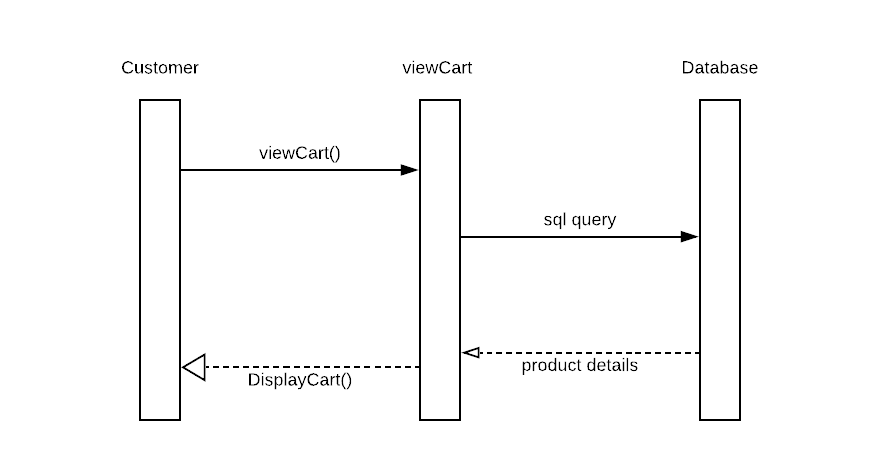
### 2.1.19 Complain

|  |
| --- |
| Complain |
| -customerId: string  -sellerId:string  -complainId: string |
| +\_Complain (string customerId, string sellerId, string complainId): Constructor. |

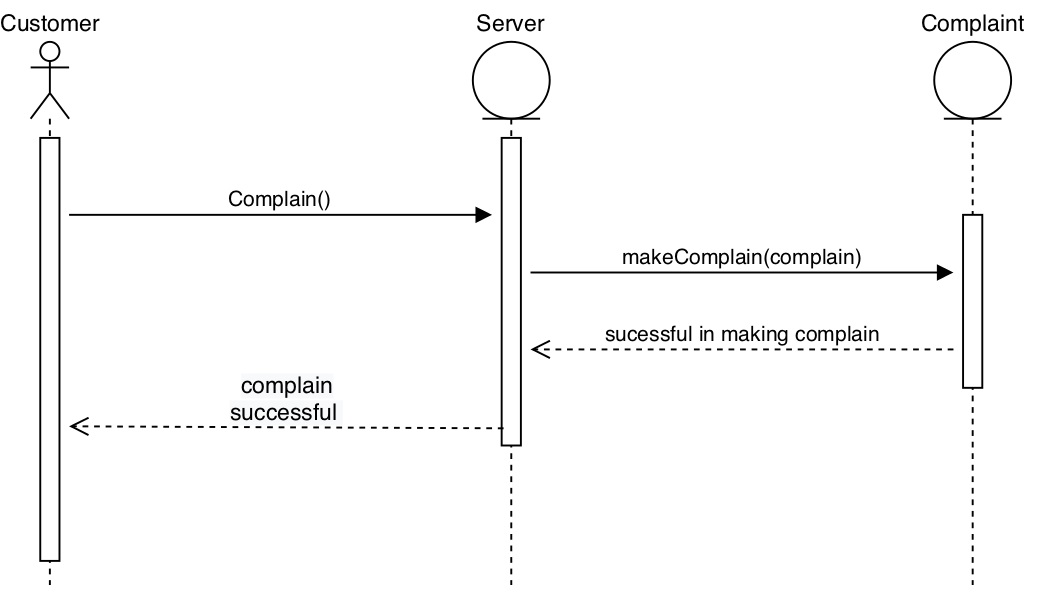
\_Complain (string customerId, string sellerId, string complainId): Constructor. Creates complain class to receive complaint from customers about the sellers.

# 3. Sequence Diagrams

## 3.1 View Cart



## 3.2 Make Complaint



## 3.3 Check Out

A close up of a piece of paper

Description automatically generated

## 3.4 Add to Cart

A screenshot of a cell phone

Description automatically generated

## 3.5 Create Account

A screenshot of a cell phone

Description automatically generated

# Appendix A. Database Design

\*denotes primary keys  
#denotes foreign keys

**Customer**

\* customerId: int

customerName: string

userName: string

password: string

address: string

accountType: string

**Seller**

\* sellerId: int

sellerUserName: string

sellerPassword: string

sellerAddress: string

accountType: string

accountBalance: float

**Product**

\* productId: int

productName: string

productUnitCost: float

sellerId: string

dateAdded: date

productRentPrice: float

productType: string

C**omplaint**

\* complainId: int

# customerId: int

# productId: int

complainType: string

complainDetails: string

**Order**

\* orderId: int

# customerId: int

# productId: int

dateOrdered: date

dateShipped: date

status: bool

**RentedProduct**

\* rentId: int

# productId: int

# customerID:int

# sellerID: int

rentDate: date

rentDuration: int

totalCost: float

creditCardNumber: int

creditCardName: string

creditCardExpirationDate: date

# Appendix B. Task & Role Assignments

Throughout the analysis and design process our group project we have furthered our ideas of how to implement and present our E-Commerce website and still have not changed our stance from our Initial Tasks and Role assignments from the Requirements document.

To reiterate:

* We will share all project responsibilities and duties.
* Design, maintain, test, evaluate requirements, and develop appropriately.