ElectroHub

Design Document, version 1

March 4, 2020

Group Members:
Dhiraj Basnet
Rohit Rauniyar
Sunil Shrestha
Bhabish Subedi

Group Number: 14

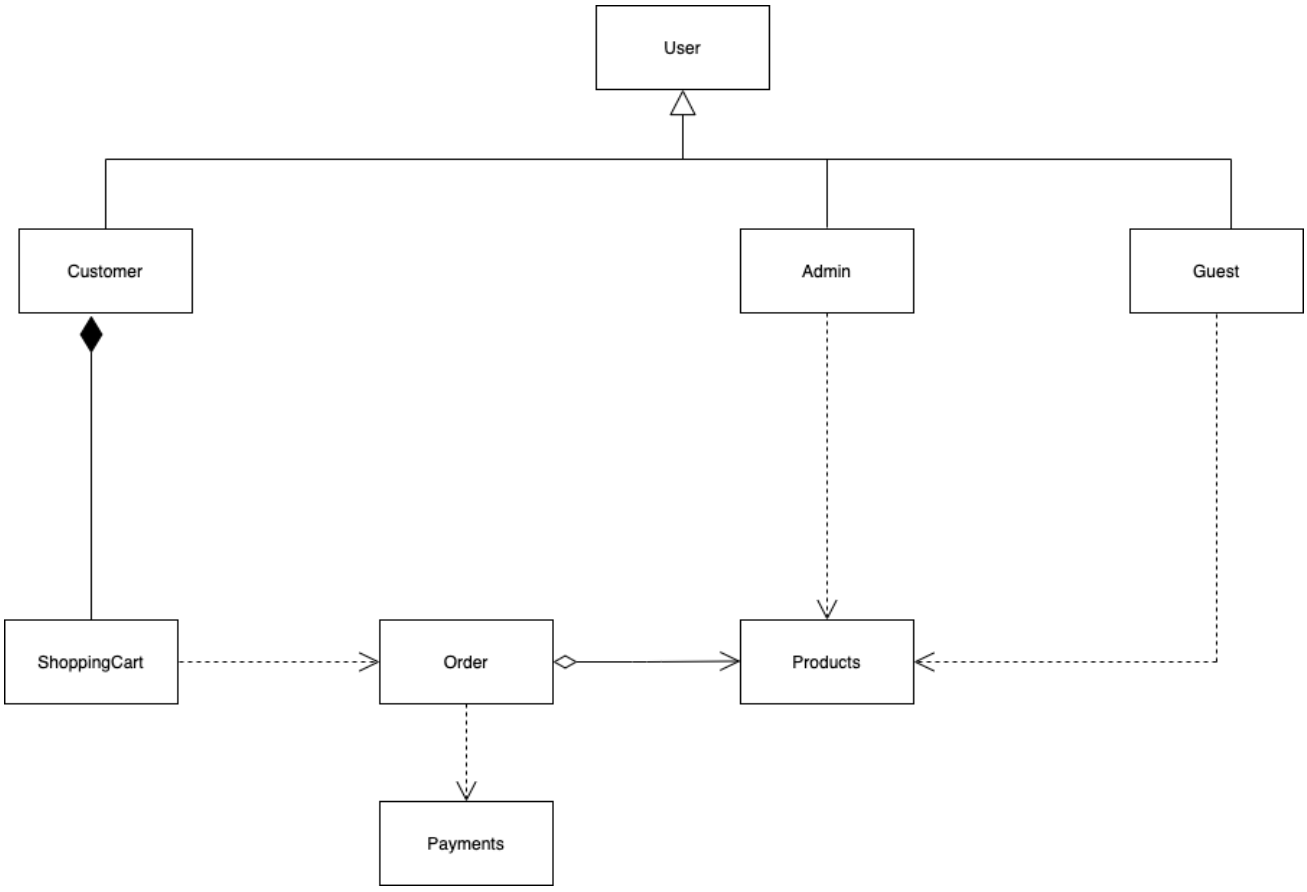
Lab Instructor:
Nicholas Dinep-Schneider

This page is intentionally left blank.

Table of Contents

This page is intentionally left blank.

1. Overview Class Diagram



2. Detailed Class Diagrams

2.1. User

User
-username: string -password: string -admin: bool -customer: bool
+User(string username) +checkAccess():bool

- User (string username): Default Constructor. Establishes username and other values to null.
- checkAccess(username: string, password: string): Sets the admin or customer value true depending on the username and password combination provided by the user. Determines privileges throughout the site.

2.2. Customer

Customer
-firstname: string -lastname: string -username: string -password: string -customerID: int -address: string -email: string -shippingAddress: string -phoneNumber: int
+Customer(int customerId) +login():bool +updateProfile() :void +logout(): void +viewProfile(): accountInfo[] +viewProducts(): product[]

- Customer(int customerId): Default constructor. Establishes Customer with an ID
- login(userName: string, password: string) : Checks the username and password provided by a user and returns true if the given information are valid else return false.
- updateProfile(firstName: string, lastName: string, address: string, email: string, username: string, password: string, customerID: int, shippingAddress: string): Allows user to update and save information in their profile such as name, email, address, and shipping address
- logout() : Logs the user out of the system and redirects to the homepage.
- viewProfile(): Returns the account information of the customer
- viewProducts(): Returns list of the products customer may interested in buying.

2.3. Guest

Guest
+Guest() +createAccount(string firstname, string lastname, string address, string email, string username, string password, string shippingAddress): void +verifyAccount(string username): bool +viewProducts(): product[]

- Guest(): Empty Constructor. Provide object to have ontrol over guest function.
- createAccount (firstname: string, lastname: string, address: string, email: string, username: string, password: string, shippingAddress: string): Creates account based on the user provided information.
- viewProducts(): Returns list of the products user may interested in buying.
- VerifyAccount(string username): Checks to see if the username already exists or not. Return true or false based on the result.

2.4. Admin

Admin
-adminFirstName: string -adminLastName: string -adminID: string -productDetails: Product
+Admin(string adminName, string email) +viewProducts(): product[] +addCategory(): void +addProducts(Product productDetails): void +deleteProducts(productID):void +modifyProducts(Product productDetails):void +updateInventory(Product productDetails):void +terminateUsers(userID):void

- Admin(): Empty Constructor. Provides object to have control over admin functions
- +viewProducts(): Returns
- addCategory(category: string): Allows admin to add category of items to the database
- addProducts(productName: string, productDetails: string, productUnitPrice: float, productID: int, productCount: integer): Allows admin to add products to the database
- modifyProducts(Product productDetails) : It modifies the product based on the attributes.
- deleteProducts(productID): Allows admin to delete the product object
- updateInventory(Product productDetails): Allows admin to update the number of items in the inventory.
- terminateUsers(userID): Allows the admin to terminate fraud and inactive users.

2.5. Product

Product
-productName: string -productID: int -productUnitCost: float -productCount: int -category: string
+Product (string productName, int productid, float productUnitCost, int productCount, string category) +filterProduct(string category): product[] +searchProduct(string productName): product[]

- Product (string productName, int productid, float productUnitCost, int productCount, string category): Constructor. Creates product objects. Stores the information about products.
- filterProduct(string category) : Returns a list of all the products filtered by category.
- searchProduct(string productName): Returns a list of all the products that matches a specific product name.

2.6. Shopping Cart

ShoppingCart
-cartID: int -productID: int -quantity: int -dateAdded: int -customerInfo: Customer
+ShoppingCart (int cartId, int productID, Customer customerInfo, int dateAdded) +addCartItem(int productID, int quantity) : void +removeItem(int productID): void +updateQuantity(): void +viewCartDetails(int quantity, int dateAdded, int productID): void +checkout (quantity:int, cartId:int): void

- ShoppingCart (int cartID, int productID, int quantity, int dateAdded, Customer customerInfo) : 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.
- RemoveItem(int productID): Remove the product of specific productID from the cart and the returns the updated cart.
- 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.

2.7. Order

Order
-orderID: int -dateCreated: date -dateShipped: string -customerInfo: Customer -status: string -totalPrice: float
+Order(Customer customerInfo, string orderID, string totalPrice) +placeOrder() : void +cancelOrder(orderID): bool +calculatePrice(): float +confirmOrder(): bool +addOrderToDatabase(int orderID):void +viewOrder(date dateCreated):void +buyAgain(int orderID):void

- Order(Customer customerInfo, string orderID, string totalPrice) : Constructor. Uses parameters to set member variables and creates order object that operates order functions.
- placeOrder() : Places the order based on the product added on the cart.
- cancelOrder(orderID): Allows user to cancel order by passing order ID.
- calculatePrice(): Returns total price of the order.
- confirmOrder(): Confirms whether the order is successfully placed or not. Return true or false
- addOrderToDatabase(): Add the orders placed to the database.
- viewOrder(date dateCreated): Display the total order placed as per the date provided.
- BuyAgain(int orderID): Allows the user to order the same order again from their order history.

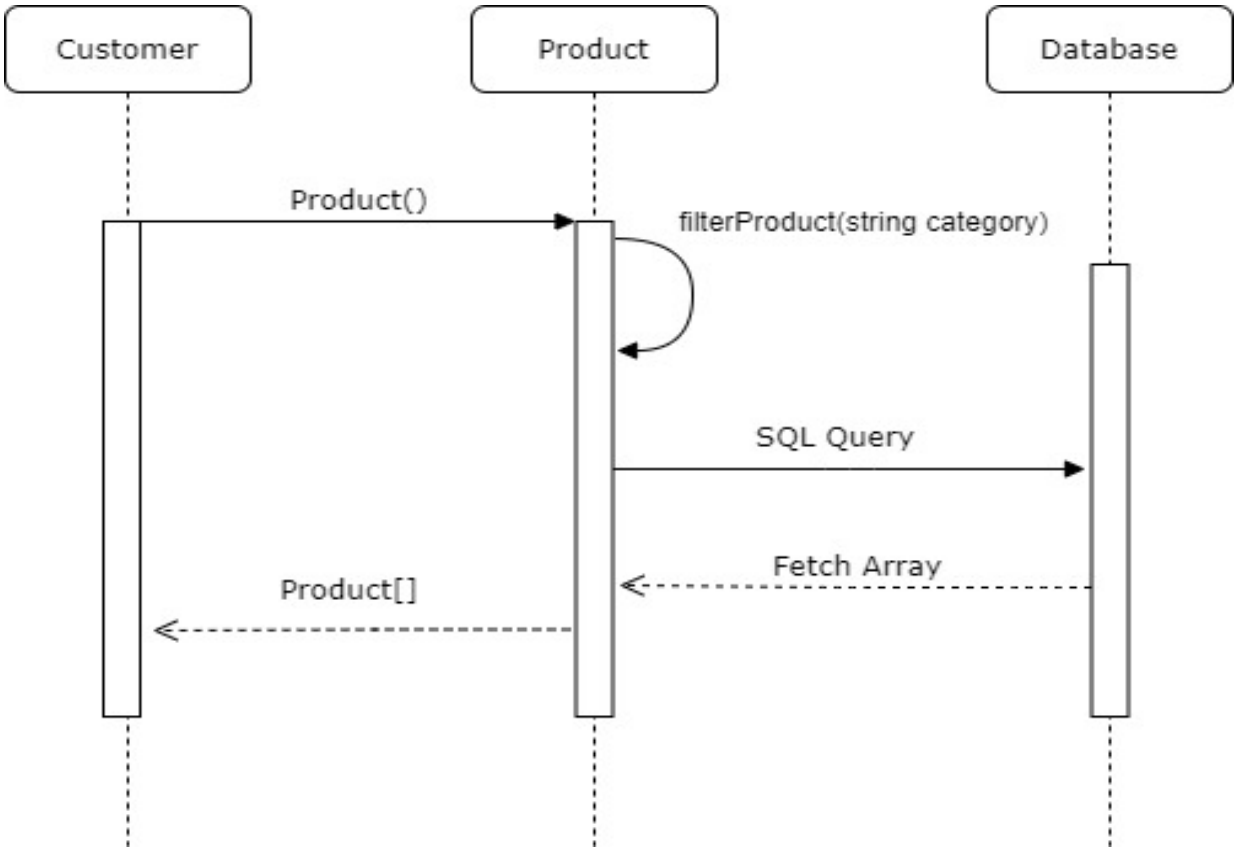
2.8. Payments

Payments
-paymentID: int -paymentDate: date -paymentAmount: float -cardType: string -cardName: string -cardNumber: string -expDate: date -cvv: int
+_Payment(string cardType, string cardName, string cardNumber, date expDate) +verifyPayment(): void

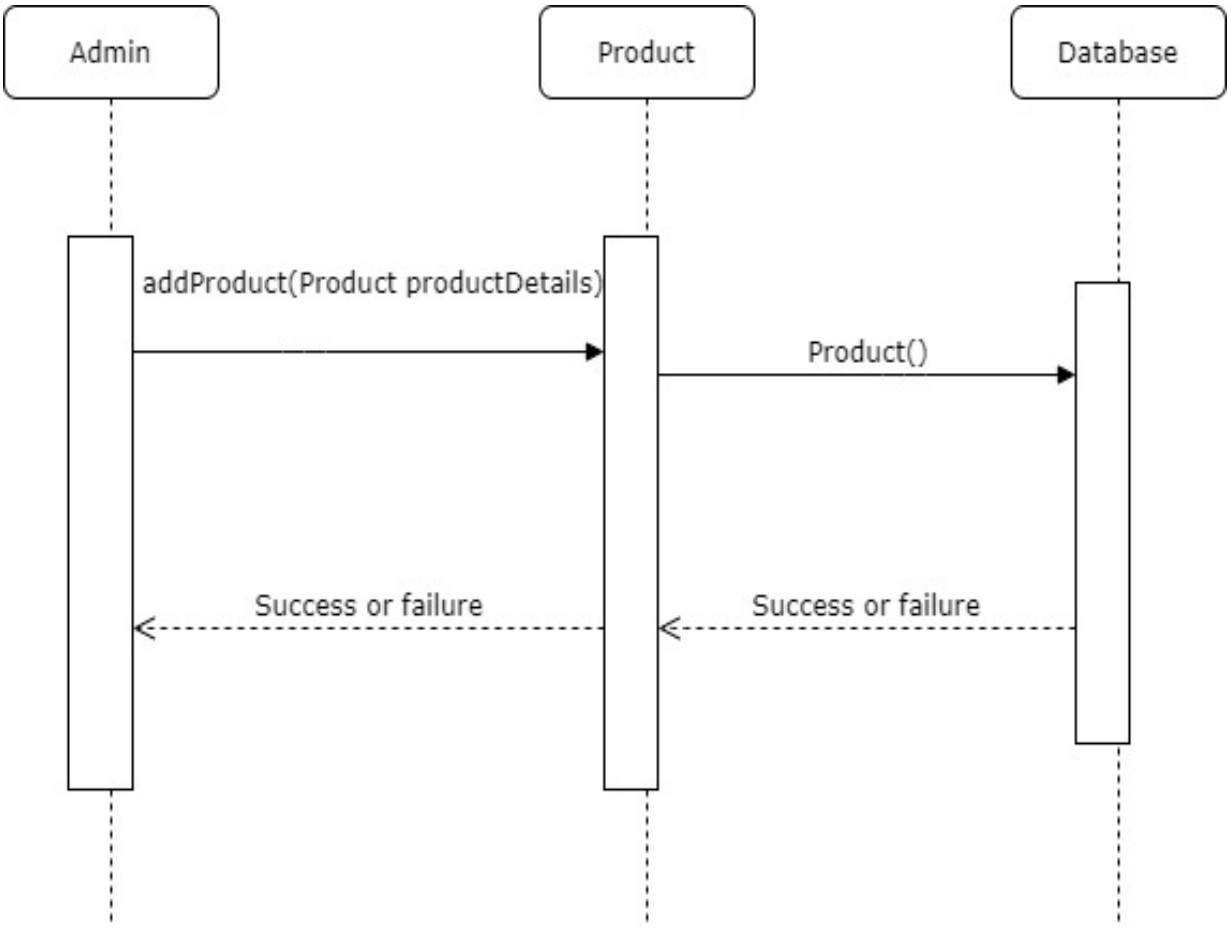
- Payment(string cardType, string cardName, string cardNumber, date expDate, int cvv) : Constructor. Passes payment details in parameters by value for later verification.
- verifyPayment() : Verifies that the credit card payment is valid.

3. Sequence Diagrams

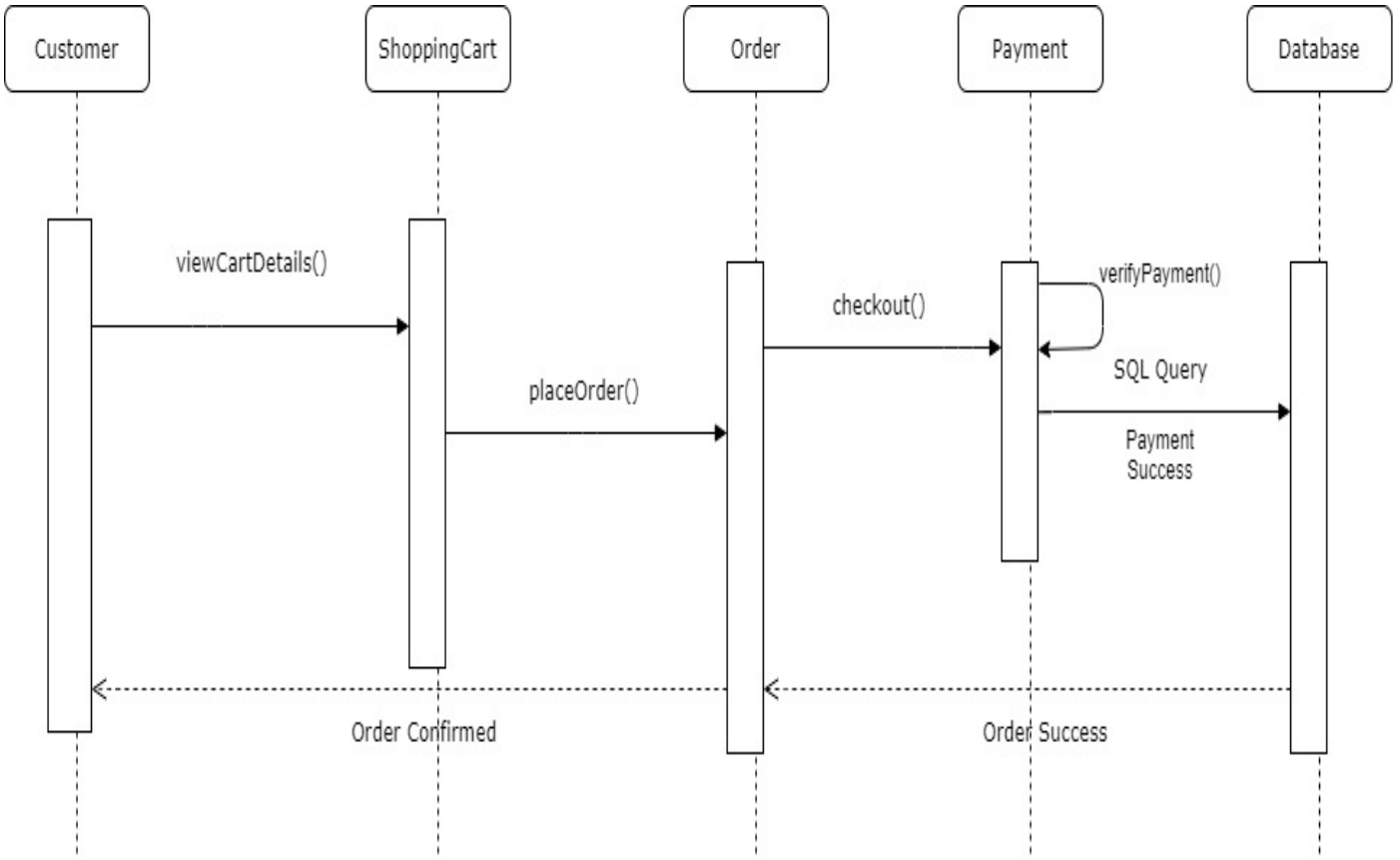
3.1. Filter Product



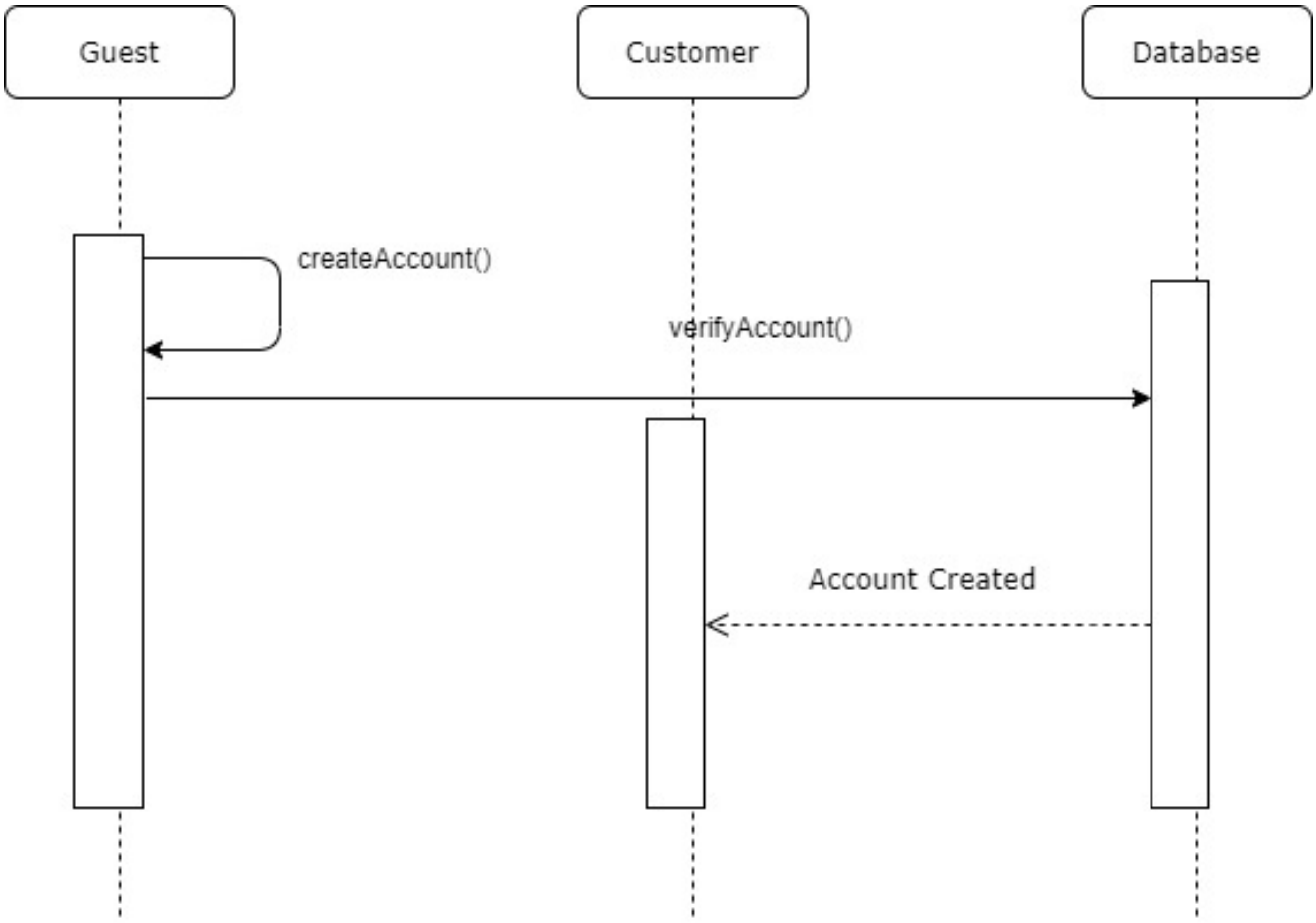
3.2. Add Product



3.3. Check Out


Customer
ShoppingCart
Order
Payment
Database
viewCartDetails()
placeOrder()
checkout()
verifyPayment()
SQL Query
Payment
Success
Order Confirmed
Order Success

3.4. Create Account



Appendix A: Database Design

*denotes primary keys
#denotes foreign keys

Customer

*customerID: int

FirstName: string

LastName: string

username: string

password: string

address: string

Email: varchar

shippingAddress: string

PhoneNumber: int

Product

*productID: int

productName: string

productUnitCost: float

ProductCount: int

category: string

Order

orderID: int

#customerID: int

#productID: int

dateCreated:date

dateShipped:date

status: string

totalPrice: float

Appendix B. Task & Role Assignments

We got to learn a lot from this groupwork design assignment. We all worked together for this design document.

1) Bhabish Subedi
    a) Frontend/Backend developer
    b) Responsible for collecting as well as analyzing the data and database design.
2) Dhiraj Basnet
    a) Frontend/Backend developer
    b) Generally did sequence diagrams as well as helped in class diagrams.
3) Sunil Shrestha
    a) Frontend/database developer
    b) Generally, did constructing data classes as well as system overview.
4) Rohit Rauniyar
    a) Frontend/Database developer
    b) Generally, worked hard for constructing sequence diagrams and as we as system overview.