

Ecommerce – Case Study

Dao:

CustomerDAO.py

```
from util.DBConnUtil import DBConnection

class CustomerDAO(DBConnection):
    def registerCustomer(self):
        try:
            connection = self.getConnection()

            name = input("Enter your Name: ")
            email = input("Enter your Email: ")
            password = input("Enter your Password: ")

            data = (name, email, password)
            cursor = connection.cursor()
            insert_query = "INSERT INTO customer (name, email, password)
VALUES (%s, %s, %s);"

            cursor.execute(insert_query, data)
            customerId = cursor.lastrowid

            connection.commit()
            connection.close()

            return customerId

        except Exception as e:
            print("In Customer DAO", str(e))
            return -1

    def checkCustomerId(self, customerId):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            select_query = "select * from customer where customer_id = %s"
            cursor.execute(select_query, (customerId,))

            rows = cursor.fetchone()

            connection.commit()
            connection.close()
            return rows

        except Exception as e:
            print("In Customer DAO", str(e))
            return None
```

ProductDAO.py

```
import mysql
from util.DBConnUtil import DBConnection
from exception.ProductNotFoundException import ProductNotFoundException

class ProductDAO(DBConnection):
    def createProduct(self, data):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            insert_query = "INSERT INTO products (name, price, description, stockQuantity) VALUES (%s, %s, %s, %s);"

            cursor.execute(insert_query, data)

            id = cursor.lastrowid

            connection.commit()
            connection.close()

            return id

        except Exception as e:
            print("In Product DAO - Create", str(e))
            return -1

    def deleteProduct(self, productId):
        try:
            connection = self.getConnection()
            data = (productId,)

            cursor = connection.cursor()
            delete_query = "delete from products where product_id = %s;"

            cursor.execute(delete_query, data)

            if cursor.rowcount == 0:
                raise ProductNotFoundException(productId)
            else:
                print("Product deleted successfully!")

            connection.commit()
            connection.close()

        except mysql.connector.Error as e:
            print("In Product DAO - Delete", str(e))

    def getAllProducts(self):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            select_query = "select * from products"
            cursor.execute(select_query)

            rows = cursor.fetchall()
            return rows
```

```

        except Exception as e:
            print("In Product DAO - Retrieve", str(e))
            return None

    def updateProductQuantity(self, productId, quantity):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            data = (quantity, productId)
            update_query = "update Products set stockQuantity =
stockQuantity - %s where product_id = %s"

            cursor.execute(update_query, data)
            connection.commit()
            connection.close()

        except Exception as e:
            print("In Product DAO - Update Quantity", str(e))
            return None

    def checkProductId(self, productId):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            select_query = "select * from products where product_id = %s"
            cursor.execute(select_query, (productId,))

            rows = cursor.fetchall()

            connection.commit()
            connection.close()

            if len(rows) == 0:
                raise ProductNotFoundException(productId)

        except ProductNotFoundException as e:
            print(e)

        except Exception as e:
            print("In Product DAO - Check Product Id", str(e))

```

CartDAO.py

```
from util.DBConnUtil import DBConnection
from dao.ProductDAO import ProductDAO
from dao.CustomerDAO import CustomerDAO
from exception.ProductNotFoundException import ProductNotFoundException

class CartDAO(DBConnection):

    def addToCart(self, customerId, productId, quantity):
        try:
            connection = self.getConnection()

            products = ProductDAO().getAllProducts()

            flag = "False"
            for p in products:
                if p[0] == productId:
                    if quantity > p[4]:
                        print("\nSorry! Quantity exceeds the Product
Quantity")
                        flag = "Exceeds"
                    else:
                        flag = "True"
                        break

            if flag == "True":
                cursor = connection.cursor()
                ProductDAO().updateProductQuantity(productId, quantity)

                row = CartDAO().checkProductInCart(productId, customerId)

                if row is None:
                    data = (customerId, productId, quantity)
                    insert_query = "insert into cart
(customer_id,product_id,quantity) values (%s,%s,%s)"

                    cursor.execute(insert_query, data)
                else:
                    data = (quantity, customerId, productId)
                    update_query = ("update cart set quantity = quantity +
%s where customer_id = %s and "
                                "product_id = %s")

                    cursor.execute(update_query, data)

                print("Product Added to your cart!!!")

                connection.commit()
                connection.close()

            elif flag == "False":
                raise ProductNotFoundException(productId)

        except ProductNotFoundException as e:
            print(e)

        except Exception as e:
            print("In Cart DAO", str(e))
```

```

def checkProductInCart(self, productId, customerId):
    try:
        connection = self.getConnection()
        data = (productId, customerId)

        cursor = connection.cursor()
        select_query = "select * from cart where product_id = %s and
customer_id = %s"

        cursor.execute(select_query, data)
        row = cursor.fetchone()

        connection.commit()
        connection.close()

        return row

    except Exception as e:
        print("In Cart DAO - Product Check", str(e))
        return None

def getAllFromCart(self, customerId):
    try:
        connection = self.getConnection()
        data = (customerId,)

        cursor = connection.cursor()
        select_query = ("select cart.product_id,name,price,quantity
from cart join products on cart.product_id = "
                        "products.product_id where customer_id = %s")
        cursor.execute(select_query, data)

        rows = cursor.fetchall()

        connection.commit()
        connection.close()

        return rows

    except Exception as e:
        print("In Cart DAO - Get Cart", str(e))
        return None

def removeFromCart(self, customerId):
    try:
        connection = self.getConnection()

        cursor = connection.cursor()
        delete_query = "delete from cart where customer_id = %s"
        cursor.execute(delete_query, (customerId,))

        connection.commit()
        connection.close()
    except Exception as e:
        print("In Cart DAO - Delete Cart", str(e))
        return None

```

OrderDAO.py

```
from util.DBConnUtil import DBConnection
from dao.CartDAO import CartDAO
from dao.OrderItemDAO import OrderItemDAO
import datetime

class OrderDAO(DBConnection):

    def placeOrder(self, data):
        try:
            connection = self.getConnection()

            customerId, street, city, state, pincode = data

            cartItems = CartDAO().getAllFromCart(customerId)
            total_price = 0

            # Printing Cart Items for reference

            print("\nYour Cart")
            print("*****\n")

            headers = ["Product Id", "Name", "Price", "Quantity"]
            header_row = "|".join(f"{header:<27}" for header in headers)
            print(header_row)
            print("-" * len(header_row))

            # Print data rows
            for row in cartItems:
                data_row = "|".join(f"{str(item):<27}" for item in row)
                total_price += row[2] * row[3]
                print(data_row)

            # Order Date
            order_date = datetime.date.today()

            # Insert Order
            cursor = connection.cursor()
            data = (customerId, order_date, total_price, street, city, state, pincode)
            insert_query = ("insert into orders "
                             "(customer_id, order_date, total_price, street, city, state, pincode) values "
                             "('%s', '%s', '%s', '%s', '%s', '%s', '%s');")
            cursor.execute(insert_query, data)

            # Retrieving Order Id
            orderId = cursor.lastrowid
            connection.commit()
            connection.close()

            # Inserting Order items

            OrderItemDAO().insertOrderItems(orderId, cartItems)

            # Removing Cart Items after Ordering

            CartDAO().removeFromCart(customerId)
            return [orderId, total_price]
```

```

        except Exception as e:
            print("In Order DAO - Place Order", str(e))
            return -1

    def getOrdersByCustomer(self, customerId):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()
            select_query = "select order_id,order_date,total_price from
orders where customer_id = %s"
            cursor.execute(select_query, (customerId,))

            rows = cursor.fetchall()
            connection.commit()
            connection.close()

            return rows

        except Exception as e:
            print("In Order DAO - Get Orders", str(e))

```

OrderItemDAO.py

```

from util.DBConnUtil import DBConnection

class OrderItemDAO(DBConnection):

    def insertOrderItems(self, orderId, cartItems):
        try:
            connection = self.getConnection()

            cursor = connection.cursor()

            for item in cartItems:
                data = (orderId, item[0], item[3])
                insert_query = "insert into order_items
(order_id,product_id,quantity) values (%s,%s,%s)"

                cursor.execute(insert_query, data)

            connection.commit()
            connection.close()
        except Exception as e:
            print("In Order Item DAO - Insert Items", str(e))

```

Entity:

Customer.py:

```
class Customer:
    def __init__(self):
        customerId = 0
        name = ''
        email = ''
        password = ''

    def setCustomerId(self, customerId):
        self.customerId = customerId

    def setName(self, name):
        self.name = name

    def setEmail(self, email):
        self.email = email

    def setPassword(self, password):
        self.password = password

    def getCustomerId(self):
        return self.customerId

    def getName(self):
        return self.name

    def getEmail(self):
        return self.email

    def getPassword(self):
        return self.password
```

Product.py:

```
class Product:
    def __init__(self):
        productId = 0
        name = ""
        price = 0
        description = ""
        stockQuantity = 0

    def setProductId(self, productId):
        self.productId = productId

    def setName(self, name):
        self.name = name

    def setPrice(self, price):
        self.price = price

    def setDescription(self, description):
        self.description = description
```



```
def setStockQuantity(self, stockQuantity):
    self.stockQuantity = stockQuantity

def getProductId(self):
    return self.productId

def getName(self):
    return self.name

def getPrice(self):
    return self.price

def getDescription(self):
    return self.description

def getStockQuantity(self):
    return self.stockQuantity
```

Cart.py

```
class Cart:
    def __init__(self):
        cardId = 0
        customerId = 0
        productId = 0
        Quantity = 0

    def setCardId(self, cardId):
        self.cardId = cardId

    def setCustomerId(self, customerId):
        self.customerId = customerId

    def setProductId(self, productId):
        self.productId = productId

    def setQuantity(self, quantity):
        self.quantity = quantity

    def getCardId(self):
        return self.cardId

    def getCustomerId(self):
        return self.customerId

    def getProductId(self):
        return self.productId

    def getQuantity(self):
        return self.quantity
```

```
class Order:
    def __init__(self):
        order_id = 0
        customer_id = 0
        order_date = ''
        total_price = 0
        street = ''
        city = ''
        state = ''
        pincode = 0

    def setOrderId(self, order_id):
        self.order_id = order_id

    def setCustomerId(self, customer_id):
        self.customer_id = customer_id

    def setOrdeDate(self, order_date):
        self.order_date = order_date

    def setTotalPrice(self, total_price):
        self.total_price = total_price

    def setStreet(self, street):
        self.street = street

    def setCity(self, city):
        self.city = city

    def setState(self, state):
        self.state = state

    def setPincode(self, pincode):
        self.pincode = pincode

    def getOrderId(self):
        return self.order_id

    def getCustomerId(self):
        return self.customer_id

    def getTotalPrice(self):
        return self.total_price

    def getOrderDate(self):
        return self.order_date

    def getStreet(self):
        return self.street

    def getCity(self):
        return self.city

    def getState(self):
        return self.state
```

```
def getPincode(self):  
    return self.prcode
```

OrderItem.py

```
class Order_Item:  
    def __init__(self):  
        order_item_id = 0  
        order_id = 0  
        product_id = 0  
        quantity = 0  
  
    def setOrderItemId(self, order_item_id):  
        self.order_item_id = order_item_id  
  
    def setOrderId(self, order_id):  
        self.order_id = order_id  
  
    def setProductId(self, product_id):  
        self.product_id = product_id  
  
    def setQuantity(self, quantity):  
        self.quantity = quantity  
  
    def getOrderItemId(self):  
        return self.order_item_id  
  
    def getOrderId(self):  
        return self.order_id  
  
    def getProductId(self):  
        return self.product_id  
  
    def getQuantity(self):  
        return self.quantity
```

Exception:

CustomerNotFoundException.py

```
class CustomerNotFoundException(Exception):
    def __init__(self, customerId):
        super().__init__(f'\n***** No Customer is found with the customer
id {customerId} *****')
```

ProductNotFoundException.py

```
class ProductNotFoundException(Exception):
    def __init__(self, productId):
        super().__init__('No products available with this product id')
```

OrderNotFoundException.py

```
class OrderNotFoundException(Exception):
    def __init__(self, productId):
        super().__init__(f'Order Id {productId} is not found in the
database')
```

Util:

DBPropertyUtil.py

```
class PropertyUtil:
    host = "localhost"
    database = "ecommerce"
    username = "root"
    password = "Saibharathi@19"

    def getConnectionString(self):
        return {"host": PropertyUtil.host, "database":
PropertyUtil.database, "username": PropertyUtil.username,
                "password": PropertyUtil.password}
```

DBConnUtil.py

```
from mysql.connector import connect
from util.DBPropertyUtil import PropertyUtil

class DBConnection:
    def getConnection(self):
        try:
            data = PropertyUtil().getConnectionString()
            connection = connect(**data)
            return connection
        except Exception as e:
            print("Sorry! Couldn't connect to the database")
            return None
```

Main:

MainModule.py

```
from dao.CustomerDAO import CustomerDAO
from dao.ProductDAO import ProductDAO
from dao.CartDAO import CartDAO
from dao.OrderDAO import OrderDAO
from exception.CustomerNotFoundException import CustomerNotFoundException
from exception.ProductNotFoundException import ProductNotFoundException

def main():
    print("Welcome to Ecommerce Application!!!")

    try:
        while True:
            print("\nSelect your Preference:")
            print("Press\n1. Register Customer\n2. Create Product\n3. Delete Product\n4. Add To Cart\n5. View Cart\n6. "
                  "Place Order\n7. View Customer Orders\n8. Stop\n")

            choice = int(input("Enter your choice: "))

            if choice == 1:
                c = CustomerDAO()
                cust_id = c.registerCustomer()

                if cust_id != -1:
                    print("\nCustomer created Successfully!!!\n")
                    print(f"Your Customer Id: {cust_id}")
                else:
                    print("\nSorry! Could not complete your request! Please Try Again!")

            elif choice == 2:
                name = input("Enter the Product Name: ")
                price = float(input("Enter the price of the Product: "))
                description = input("Enter the Product Description: ")
                stockQuantity = int(input("Enter the Stock Quantity of the Product: "))

                data = (name, price, description, stockQuantity)
                p = ProductDAO()
                product_id = p.createProduct(data)

                if product_id != -1:
                    print("\nProduct created Successfully!!!\n")
                    print(f"Product Id: {product_id}")
                else:
                    print("\nSorry! Could not complete your request! Please Try Again!")

            elif choice == 3:
                product_id = int(input("Enter the Product Id to delete: "))
                p = ProductDAO()
                p.deleteProduct(product_id)
```

```

        elif choice == 4:
            customerId = int(input("Enter your Customer Id: "))
            flag = CustomerDAO().checkCustomerId(customerId)

            if flag is not None:
                products = ProductDAO().getAllProducts()

                if products is None:
                    print("Sorry!No Products Available")
                else:
                    headers = ["Product Id", "Name", "Price",
"Description", "Product Quantity", ]

                    header_row = "|".join(f"{header:<27}" for header in
headers)

                    print(header_row)
                    print("-" * len(header_row))

                    # Print data rows
                    for row in products:
                        data_row = "|".join(f"{str(item):<27}" for item
in row)

                        print(data_row)

                    productId = int(input("\nEnter the Product Id: "))
                    quantity = int(input("Enter the Quantity you want:
"))

                    c = CartDAO()
                    c.addToCart(customerId,productId,quantity)

            else:
                raise CustomerNotFoundException(customerId)

        elif choice == 5:
            customerId = int(input("Enter your Customer Id: "))
            flag = CustomerDAO().checkCustomerId(customerId)
            if flag is not None:
                c = CartDAO()
                rows = c.getAllFromCart(customerId)

                if len(rows) > 0:
                    print("\nYour Cart")
                    print("*****\n")

                    headers = ["Product Id", "Name", "Price",
"Quantity"]

                    header_row = "|".join(f"{header:<27}" for header in
headers)

                    print(header_row)
                    print("-" * len(header_row))

                    # Print data rows
                    for row in rows:
                        data_row = "|".join(f"{str(item):<27}" for item
in row)

                        print(data_row)
                    else:
                        print("\nYou have no products in your cart!!!")
                else:
                    raise CustomerNotFoundException(customerId)

```

```

elif choice == 6:
    customerId = int(input("Enter your Customer Id: "))
    flag = CustomerDAO().checkCustomerId(customerId)

    if flag is not None:
        cartItems = CartDAO().getAllFromCart(customerId)

        if len(cartItems) != 0:
            street = input("Enter your Street Name: ")
            city = input("Enter your City Name: ")
            state = input("Enter your state Name: ")
            pincode = input("Enter the pincode: ")

            data = (customerId, street, city, state, pincode)
            o = OrderDAO()
            order = o.placeOrder(data)

            if order[0] != -1:
                print("\nOrder created Successfully!!!\n")
                print(f"Your Order Id: {order[0]}")
                print(f"Total Price: {order[1]}")
            else:
                print("\nSorry! Could not place your order!
Please Try Again!")
        else:
            print("\nAdd products to your cart to place an
Order!!!!")
    else:
        raise CustomerNotFoundException(customerId)

elif choice == 7:
    customerId = int(input("Enter your Customer Id: "))
    flag = CustomerDAO().checkCustomerId(customerId)

    if flag is not None:
        o = OrderDAO()
        rows = o.getOrdersByCustomer(customerId)

        if len(rows) > 0:
            print("\nYour Orders:\n")
            headers = ["Order Id", "Order Date", "Total Price"]

            header_row = "|".join(f"{header:<20}" for header in
headers)

            print(header_row)
            print("-" * len(header_row))

            # Print data rows
            for row in rows:
                data_row = "|".join(f"{str(item):<20}" for item
in row)

                print(data_row)
            else:
                print("\nYou haven't placed any order so far!")
        else:
            raise CustomerNotFoundException(customerId)
    else:
        print("\n***** Thank You using the
application!!! *****")
        break

```



```
except CustomerNotFoundException as e:  
    print(e)  
  
if __name__ == "__main__":  
    main()
```

Outputs:

Register Customer

```
Welcome to Ecommerce Application!!!  
  
Select your Preference:  
Press  
1. Register Customer  
2. Create Product  
3. Delete Product  
4. Add To Cart  
5. View Cart  
6. Place Order  
7. View Customer Order  
8. Stop  
  
Enter your choice: 1  
Enter your Customer Name: Saibharathi  
Enter your Email: saibha@gmail.com  
Enter your Password: saib  
  
Customer created Successfully!!!  
  
Your Customer Id: 1
```

Create Product

```
Select your Preference:
Press
1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 2
Enter the Product Name: Laptop
Enter the price of the Product: 60000
Enter the Product Description: High Performance Laptop
Enter the Stock Quantity of the Product: 12

Product created Successfully!!!

Product Id: 1
```

	product_id	name	price	description	stockQuantity
▶	1	Laptop	60000.00	High Performance Laptop	12
	2	HeadPhone	2000.00	Wireless	30
*	NULL	NULL	NULL	NULL	NULL

Delete Product

```
Select your Preference:
Press
1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 3
Enter the Product Id to delete: 2
Product deleted successfully!
```

	product_id	name	price	description	stockQuantity
▶	1	Laptop	60000.00	High Performance Laptop	12
✱	NULL	NULL	NULL	NULL	NULL

Add To cart

```
Select your Preference:
Press
1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 4
Enter your Customer Id: 1
Product Id      |Name      |Price      |Description      |Product Quantity
-----
1                |Laptop    |60000.00    |High Performance Laptop |12
3                |Mobile    |20000.00    |8GB RAM          |20
4                |HeadPhones|2200.00     |Wireless         |20

Enter the Product Id: 3
Enter the Quantity you want: 2
Product Added to your cart!!!
```

	cart_id	customer_id	product_id	quantity
▶	1	1	3	2
•	NULL	NULL	NULL	NULL

View Cart

```
Select your Preference:
Press
1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 5
Enter your Customer Id: 1

Your Cart
*****

Product Id      |Name      |Price      |Quantity
-----
3               |Mobile    |20000.00   |2
```

Place Order

```
Press
1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 6
Enter your Customer Id: 1
Enter your Street Name: South Street
Enter your City Name: Sirkazhi
Enter your state Name: TamilNadu
Enter the pincode: 609111

Your Cart
*****

Product Id      |Name      |Price      |Quantity
-----
3               |Mobile    |20000.00   |2

Order created Successfully!!!

Your Order Id: 1
Total Price: 40000.00
```

	order_id	customer_id	order_date	total_price	street	city	state	pincode
▶	1	1	2024-03-08	40000.00	South Street	Sirkazhi	TamilNadu	609111
•	NULL	NULL	NULL	NULL	NULL	NULL	NULL	NULL

View Customer Orders

Select your Preference:

Press

1. Register Customer
2. Create Product
3. Delete Product
4. Add To Cart
5. View Cart
6. Place Order
7. View Customer Order
8. Stop

Enter your choice: 7

Enter your Customer Id: 1

Your Orders:

Order Id	Order Date	Total Price

1	2024-03-08	40000.00

Pytest:

Test_module.py

```
import pytest
from exception.ProductNotFoundException import ProductNotFoundException
from dao.ProductDAO import ProductDAO
from dao.CartDAO import CartDAO
from dao.OrderDAO import OrderDAO

def test_product_not_found_exception():
    with pytest.raises(ProductNotFoundException) as info:
        ProductDAO().checkProductId(9)
    assert str(info.value) == 'No products available with this product id'

def test_create_product():
    data = ("Eraser", 5, "Smooth", 20)
    product_dao = ProductDAO()

    product_id = product_dao.createProduct(data)

    assert product_id > 0

def test_add_to_cart(capfd):
    cart_dao = CartDAO()
    cart_dao.addToCart(1,1,2)

    captured = capfd.readouterr()

    assert "Product Added to your cart!!!" in captured.out

def test_place_order():
    order_dao = OrderDAO()
    data = (1,"South","Chennai","Tamil Nadu",609100)
    order = order_dao.placeOrder(data)
```

```
assert order[0] > 0
```

```
C:\Users\saisa\PycharmProjects\Sainivetha_Ecommerce\.venv\Scripts\python.exe "C:/Program Files/JetBrains/PyCharm Community Edition 2023.3.4
Testing started at 08:04 am ...
Launching pytest with arguments C:\Users\saisa\PycharmProjects\Sainivetha_Ecommerce\Test\test_module.py --no-header --no-summary -q in C:\U

===== test session starts =====
collecting ... collected 4 items

test_module.py::test_product_not_found_exception PASSED          [ 25%]
test_module.py::test_create_product PASSED                       [ 50%]
test_module.py::test_add_to_cart PASSED                           [ 75%]
test_module.py::test_place_order PASSED                           [100%]
Your Cart
*****

Product Id      |Name      |Price      |Quantity
-----
1               |Laptop    |60000.00   |2

===== 4 passed in 1.03s =====

Process finished with exit code 0
```