Ecommerce Application

Tarun Tamang

Classes

Cart

```
from entity.customers import Customers
from entity.products import Products


class Cart(Customers,Products):
    def __init__(self):
        super().__init__()
        self.cart_id = 0
        self.quantity = ''

    def get_cart_id(self):
        return self.cart_id

    def get_quantity(self):
        return self.quantity

    def set_card_id(self,cart_id):
        self.cart_id=cart_id

    def set_quantity(self,quantity):
        self.quantity=quantity
```

Orders

```
from entity.customers import Customers


class Orders(Customers):
    def __init__(self):
        super().__init__()
        self.order_id = 0
        self.order_date = ''
        self.shipping_address=''
        self.total_price=0.0

    def get_order_id(self):
        return self.order_id

    def get_order_date(self):
        return self.order_date

    def get_shipping_address(self):
        return self.shipping_address

    def get_total_price(self):
```

```
        return self.total_price

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

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

    def set_shipping_address(self,shipping_address):
        self.shipping_address=shipping_address

    def set_total_price(self,total_price):
        self.total_price=total_price
```

Customers

```
from util.DBconnection import DBConnection


class Customers(DBConnection):
    def __init__(self):
        super().__init__()
        self.customer_id = 0
        self.name = ''
        self.email = ''
        self.password = ''

    def get_customer_id(self):
        return self.customer_id

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

    def get_name(self):
        return self.name

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

    def get_email(self):
        return self.email

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

    def get_password(self):
        return self.password

    def set_password(self,password):
        self.password = password
```

Products

```
from util.DBconnection import DBConnection


class Products(DBConnection):
    def __init__(self):
        super().__init__()
        self.product_id = 0
        self.name = ''
        self.price = 0.0
        self.description = ''
        self.stockQuantity = 0

    def get_product_id(self):
        return self.product_id

    def set_product_id(self,product_id):
        self.product_id = product_id

    def get_name(self):
        return self.name

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

    def get_description(self):
        return self.description

    def set_description(self, description):
        self.description = description

    def get_stockQuantity(self):
        return self.stockQuantity

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

OrderItems

```
from entity.orders import Orders
from entity.products import Products


class OrderItems(Orders,Products):
    def __init__(self):
        super().__init__()
        self.order_item_id = 0
        self.quantity = 0
```

```
    def get_order_item_id(self):
        return self.order_item_id

    def set_order_item_id(self,order_item_id):
        self.order_item_id=order_item_id

    def get_quantity(self):
        return self.quantity

    def set_quantity(self,quantity):
        self.quantity=quantity
```

Daos

Order Processor Repository

```
from dao.productsdao import ProductsDao
from dao.customersdao import CustomersDao
from dao.cartdao import CartDao
from dao.ordersdao import OrdersDao
from exception.Customer_not_FoundException import CustomerNotFoundException


class OrderProcessorRepository(CartDao,OrdersDao):
    def create_product(self) -> bool:
        p = ProductsDao()
        p.add_product()
        p.select_product()
        pass

    def create_customer(self) -> bool:
        c = CustomersDao()
        c.add_customer()
        c.select_customer()
        pass

    def delete_product(self) -> bool:
        p = ProductsDao()
        p.delete_product()
        p.select_product()
        pass

    def delete_customer(self) -> bool:
        c = CustomersDao()
        c.delete_customer()
        c.select_customer()

        pass

    def add_to_cart(self) -> bool:
        c = CartDao()
        c.add_cart()
```

```
        c.select_cart()
        pass

    def remove_from_cart(self) -> bool:
        c = CartDao()
        c.delete_cart()
        c.select_cart()
        pass

    def get_all_from_cart(self) -> bool:
        c = CartDao()
        c.select_cart()
        pass

    def place_order(self) -> bool:
        o = OrdersDao()
        o.add_order()
        o.select_order()
        return True

    def viewCustomerOrder(self, customer_id) -> bool:
        try:
            self.open()
            # customer_id = int(input("Enter customer ID: "))
            self.stmt.execute(f'''SELECT COUNT(*) FROM Orders WHERE
customer_id = {customer_id}''')
            count = self.stmt.fetchone()[0]
            if count == 0:
                return CustomerNotFoundException(customer_id)
            else:
                self.stmt.execute(f'''SELECT * FROM Orders WHERE
customer_id = {customer_id} ''')
                records = self.stmt.fetchall()
                self.close()
                return records
        except CustomerNotFoundException as e:
            return e
        except Exception as e:
            return e
```

ProductsDao

```
from entity.products import Products

class ProductsDao( Products):
    def __init__(self):
        super().__init__()


    def perform_product_actions(self):
        while True:
            print("(Products) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
            ch=int(input("Enter Choice: "))
            if ch==1:
```

```
                self.create_product_table()
            elif ch==2:
                print(self.add_product())
            elif ch==3:
                print(self.update_product())
            elif ch==4:
                print(self.delete_product())
            elif ch==5:
                self.select_product()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_product_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Product(
            product_id int primary key,
            name varchar(50) not null,
            description varchar(50),
            price double(10,2) not null ,
            stockQuantity int not null
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Products Table created Successfully")
        except Exception as e:
            print(e)


    def add_product(self):
        try:
            self.open()
            self.product_id=int(input("Product ID: "))
            self.name=input("Product Name : ")
            self.description = input("Description : ")
            self.price = input("Price : ")
            self.stockQuantity = int(input("Quantity in stock : "))
            data =
[(self.product_id,self.name,self.description,self.price,self.stockQuantity)
]
            insert_str = '''INSERT into
Product(product_id,name,description,price,stockQuantity)
            values(%s,%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_product(self):
        try:
            self.open()
            self.product_id = int(input("Product ID: "))
            self.name = input("Product Name : ")
            self.description = input("Description : ")
```

```
            self.price = input("Price : ")
            self.stockQuantity = int(input("Quantity in stock : "))
            data = [(self.product_id, self.name, self.description,
self.price, self.stockQuantity)]
            update_str = '''Update Product set
product_id=%s,name=%s,description=%s,price=%s,
                        stockQuantity=%s '''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_product(self):
        try:
            self.open()
            productId = input("Enter Product Id to be Deleted : ")
            delete_str = f'''Delete from Product where product_id =
{productId}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_product(self):
        try:
            select_str = '''select * from Product'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Product Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrdersDao

```
from entity.orders import Orders


class OrdersDao(Orders):
    def __init__(self):
        super().__init__()


    def perform_orders_actions(self):
        while True:
            print("(Orders) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")
```

```
            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_orders_table()
            elif ch==2:
                print(self.add_order())
            elif ch==3:
                print(self.update_order())
            elif ch==4:
                print(self.delete_order())
            elif ch==5:
                self.select_order()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_orders_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Orders(
            order_id int primary key,
            customer_id int,
            order_date Date,
            total_price float,
            shipping_address varchar(255),
            foreign key (customer_id) references Customers(customer_id)
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Cart Table created Successfully")
        except Exception as e:
            print(e)


    def add_order(self):
        try:
            self.open()
            self.order_id = int(input("Order ID: "))
            self.customer_id = int(input("Customer Id : "))
            self.order_date = input("Order Date : ")
            self.total_price = int(input("Total Price : "))
            self.shipping_address = input("Shipping Address : ")
            data =
[(self.order_id,self.customer_id,self.order_date,self.total_price,self.ship
ping_address)]
            insert_str = '''INSERT into
Orders(order_id,customer_id,order_date,total_price,shipping_address)
            values(%s,%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_order(self):
        try:
```

```
            self.open()
            self.order_id = int(input("Order ID: "))
            self.customer_id = int(input("Customer Id : "))
            self.order_date = input("Order Date : ")
            self.total_price = int(input("Total Price : "))
            self.shipping_address = input("Shipping Address : ")
            data = [(self.order_id, self.customer_id, self.order_date,
self.total_price, self.shipping_address)]
            update_str = '''Update Orders set
order_id=%s,customer_id=%s,order_date=%s,
            total_price=%s,shipping_address=%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_order(self):
        try:
            self.open()
            order_id = input("Enter Order Id to be Deleted : ")
            delete_str = f'''Delete from Orders where order_id =
{order_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_order(self):
        try:
            select_str = '''select * from Orders'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Orders Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

OrderItemsDao

```
from entity.orderitems import OrderItems
# from entity.orders import Orders
# from entity.products import Products


class OrderItemsDao(OrderItems):
    def __init__(self):
        super().__init__()
```

```
    def perform_orderitems_actions(self):
        while True:
            print("(OrdersItems) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE
5.SELECT 0.EXIT")

            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_orderitems_table()
            elif ch==2:
                print(self.add_orderitem())
            elif ch==3:
                print(self.update_orderitem())
            elif ch==4:
                print(self.delete_orderitem())
            elif ch==5:
                self.select_orderitem()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_orderitems_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS OrderItems(
            order_item_id int primary key,
            order_id int,
            product_id int,
            quantity int,
            foreign key (order_id) references Orders(order_id),
            foreign key (product_id) references Product(product_id)
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Cart Table created Successfully")
        except Exception as e:
            print(e)


    def add_orderitem(self):
        try:
            self.open()
            self.order_item_id = int(input("Order Item ID: "))
            self.order_id= int(input("Order Id : "))
            self.product_id = int(input("Product Id : "))
            self.quantity = input("Quantity : ")
            data =
[(self.order_item_id,self.order_id,self.product_id,self.quantity)]
            insert_str = '''INSERT into
OrderItems(order_item_id,order_id,product_id,quantity)
            values(%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e
```

```
    def update_orderitem(self):
        try:
            self.open()
            self.order_item_id = int(input("Order Item ID: "))
            self.order_id = int(input("Order Id : "))
            self.product_id = int(input("Product Id : "))
            self.quantity = input("Quantity : ")
            data = [(self.order_item_id, self.order_id, self.product_id,
self.quantity)]
            insert_str = '''Update OrderItems set
order_item_id=%s,order_id=%s,product_id=%s,quantity=%s'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_orderitem(self):
        try:
            self.open()
            order_item_id = input("Enter Order Item Id to be Deleted : ")
            delete_str = f'''Delete from OrderItems where order_item_id =
{order_item_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_orderitem(self):
        try:
            select_str = '''select * from OrderItems'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Orders Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

CustomerDao

```
from entity.customers import Customers


class CustomersDao(Customers):
    def __init__(self):
        super().__init__()
```

```
    def perform_customer_actions(self):
        while True:
            print("(Customers) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")

            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_customer_table()
            elif ch==2:
                print(self.add_customer())
            elif ch==3:
                print(self.update_customer())
            elif ch==4:
                print(self.delete_customer())
            elif ch==5:
                self.select_customer()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_customer_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Customers(
            customer_id int primary key,
            name varchar(50) not null,
            email varchar(50),
            password varchar(50) not null
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Customers Table created Successfully")
        except Exception as e:
            print(e)


    def add_customer(self):
        try:
            self.open()
            self.customer_id=int(input("Customer ID: "))
            self.name=input("Customer Name : ")
            self.email = input("Email : ")
            self.password = input("Password : ")
            data = [(self.customer_id,self.name,self.email,self.password)]
            insert_str = '''INSERT into
Customers(customer_id,name,email,password)
            values(%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_customer(self):
        try:
```

```
            self.open()
            self.customer_id = int(input("Customer ID: "))
            self.name = input("Customer Name : ")
            self.email = input("Email : ")
            self.password = input("Password : ")
            data = [(self.customer_id, self.name, self.email,
self.password)]
            update_str = '''Update Customers set
customer_id=%s,name=%s,email=%s,password=%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_customer(self):
        try:
            self.open()
            customer_id = input("Enter Customer Id to be Deleted : ")
            delete_str = f'''Delete from Customers where customer_id =
{customer_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_customer(self):
        try:
            select_str = '''select * from Customers'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Customers Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

Cart

```
from entity.cart import Cart


class CartDao(Cart):
    def __init__(self):
        super().__init__()


    def perform_cart_actions(self):
```

```
        while True:
            print("(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT
0.EXIT")

            ch=int(input("Enter Choice: "))
            if ch==1:
                self.create_cart_table()
            elif ch==2:
                print(self.add_cart())
            elif ch==3:
                print(self.update_cart())
            elif ch==4:
                print(self.delete_cart())
            elif ch==5:
                self.select_cart()
            elif ch==0:
                break
            else:
                print("Invalid Choice")


    def create_cart_table(self):
        try:
            create_str = '''CREATE TABLE IF NOT EXISTS Cart(
            cart_id int primary key,
            quantity varchar(50) not null,
            customer_id int,
            product_id int,
            foreign key (product_id) references Product(product_id),
            foreign key (customer_id) references Customers(customer_id)
            )'''
            self.open()
            self.stmt.execute(create_str)
            self.close()
            print("Cart Table created Successfully")
        except Exception as e:
            print(e)


    def add_cart(self):
        try:
            self.open()
            self.cart_id=int(input("Cart ID: "))
            self.quantity=input("Quantity : ")
            self.customer_id = int(input("Customer Id : "))
            self.product_id = int(input("Product Id : "))
            data =
[(self.cart_id,self.quantity,self.customer_id,self.product_id)]
            insert_str = '''INSERT into
Cart(cart_id,quantity,customer_id,product_id)
            values(%s,%s,%s,%s)'''
            self.stmt.executemany(insert_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def update_cart(self):
```

```
        try:
            self.open()
            self.cart_id = int(input("Cart ID: "))
            self.quantity = input("Quantity : ")
            self.customer_id = int(input("Customer Id : "))
            self.product_id = int(input("Product Id : "))
            data = [(self.cart_id, self.quantity, self.customer_id,
self.product_id)]
            update_str = '''Update cart set cart_id=%s ,
quantity=%s,customer_id=%s,product_id=%s'''
            self.stmt.executemany(update_str, data)
            self.conn.commit()
            self.close()
            return True

        except Exception as e:

            return e

    def delete_cart(self):
        try:
            self.open()
            cart_id = input("Enter Cart Id to be Deleted : ")
            delete_str = f'''Delete from Cart where cart_id = {cart_id}'''
            self.stmt.execute(delete_str)
            self.conn.commit()
            self.close()
            return True
        except Exception as e:
            return e

    def select_cart(self):
        try:
            select_str = '''select * from Cart'''
            self.open()
            self.stmt.execute(select_str)
            records = self.stmt.fetchall()
            self.close()
            print("Records in Cart Table : ")
            for i in records:
                print(i)
        except Exception as e:
            print(e)
```

Exceptions

```
class CustomerNotFoundException(Exception):
    def __init__(self, message="Customer not found"):
        self.message = message
        super().__init__(self.message)
```

```
class OrderNotFoundException(Exception):
    def __init__(self, message="Order not found"):
        self.message = message
        super().__init__(self.message)
```

```
class ProductNotFoundException(Exception):
    def __init__(self, message="Product not found"):
        self.message = message
        super().__init__(self.message)
```

Main

```
from util.DBconnection import DBConnection
from dao.cartdao import CartDao
from dao.ordersdao import OrdersDao
from dao.productsdao import ProductsDao
from dao.customersdao import CustomersDao
from dao.orderitemsdao import OrderItemsDao
from dao.OrderProcessorRepository import OrderProcessorRepository
from exception.Order_not_FoundException import OrderNotFoundException
from exception.Product_not_FoundException import ProductNotFoundException
from exception.Customer_not_FoundException import CustomerNotFoundException


class EcommerceApplicationMain:


    @staticmethod
    def main():
        dbconnection = DBConnection()

        try:
            dbconnection.open()
            print("Database Connected")
        except Exception as e:
            print(e)

        try:
            print("_" * 30)
            print("Ecommerce Application")
            print("_" * 30)
            print("Welcome to Ecommerce Application!")

            order_processor = OrderProcessorRepository()

            while True:
                print("1.Products 2.Customers 3.Orders 4.OrderItems 5.Cart
0.EXIT")
                ch = int(input("Enter choice: "))
                if ch == 1:
                    p = ProductsDao()
                    p.perform_product_actions()
                elif ch == 2:
                    c = CustomersDao()
                    c.perform_customer_actions()
                elif ch == 3:
                    e = OrdersDao()
                    e.perform_orders_actions()
                elif ch == 4:
                    o = OrderItemsDao()
```

```
                    o.perform_orderitems_actions()
                elif ch == 5:
                    u = CartDao()
                    u.perform_cart_actions()

                elif ch == 0:
                    break
                else:
                    print("Invalid choice")

            while True:
                print("=" * 10)
                print("---Main Menu---")
                print("=" * 10)
                print('''
                1.Create Product
                2.Create Customer
                3.Delete Product
                4.Delete Customer
                5.Add to Cart
                6.Remove from Cart
                7.Get All Products
                8.Place Order
                9.View Customer Order
                0.EXIT''')
                ch = int(input("Enter choice: "))
                if ch == 1:
                    print(f'Product Created :
{order_processor.create_product()}')
                elif ch == 2:
                    print(f'Customer Created :
{order_processor.create_customer()}')
                elif ch == 3:
                    print(f'Product Deleted :
{order_processor.delete_product()}')
                elif ch == 4:
                    print(f'Customer Deleted :
{order_processor.delete_customer()}')
                elif ch == 5:
                    print(f'Added to Cart :
{order_processor.add_to_cart()}')
                elif ch == 6:
                    print(f'Removed from
Cart{order_processor.remove_from_cart()}')
                elif ch == 7:
                    print(f'All from Cart :
{order_processor.get_all_from_cart()}')
                elif ch == 8:
                    print(f'Order Placed :
{order_processor.place_order()}')
                elif ch == 9:
                    print(f'Customer Orders :
{order_processor.viewCustomerOrder(int(input("Enter Customer Id : ")))}')
                elif ch == 0:
                    break
                else:
                    print("Invalid choice")



        except CustomerNotFoundException as e:
```

```
            print(e)

        except OrderNotFoundException as e:
            print(e)

        except ProductNotFoundException as e:
            print(e)

        except Exception as e:
            print(e)


if __name__ == "__main__":
    EcommerceApplicationMain.main()
```

Util

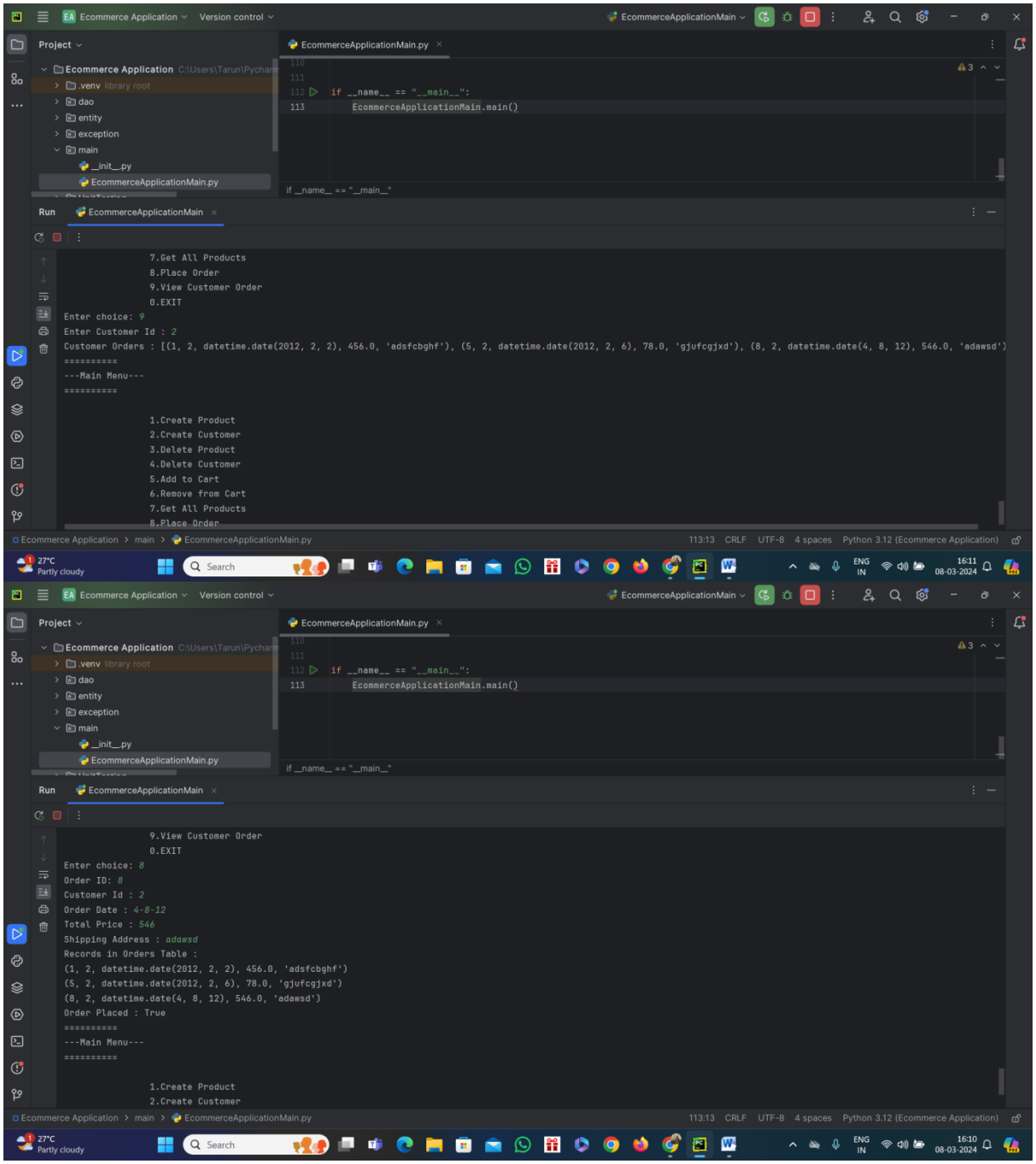
```
class DBUtil:
    connection_properties = None

    @staticmethod
    def getDBConn():
        if DBUtil.connection_properties is None:
            host = 'localhost'
            database = 'Ecommerce_application'
            port = '3306'
            user = 'root'
            password = 'root'
            DBUtil.connection_properties = {'host': host,'database':
database,'port':port,'user': user,'password': password}
        return DBUtil.connection_properties
```

```
import sys
import mysql.connector as sql
from util.DButil import DBUtil


class DBConnection:
    def open(self):
        try:
            connection_properties=DBUtil.getDBConn()
            self.conn=sql.connect(**connection_properties)
            self.stmt=self.conn.cursor()
        except Exception as e:
            print(str(e) + 'Database not Connected:')
            sys.exit(1)

    def close(self):
        self.conn.close()
```

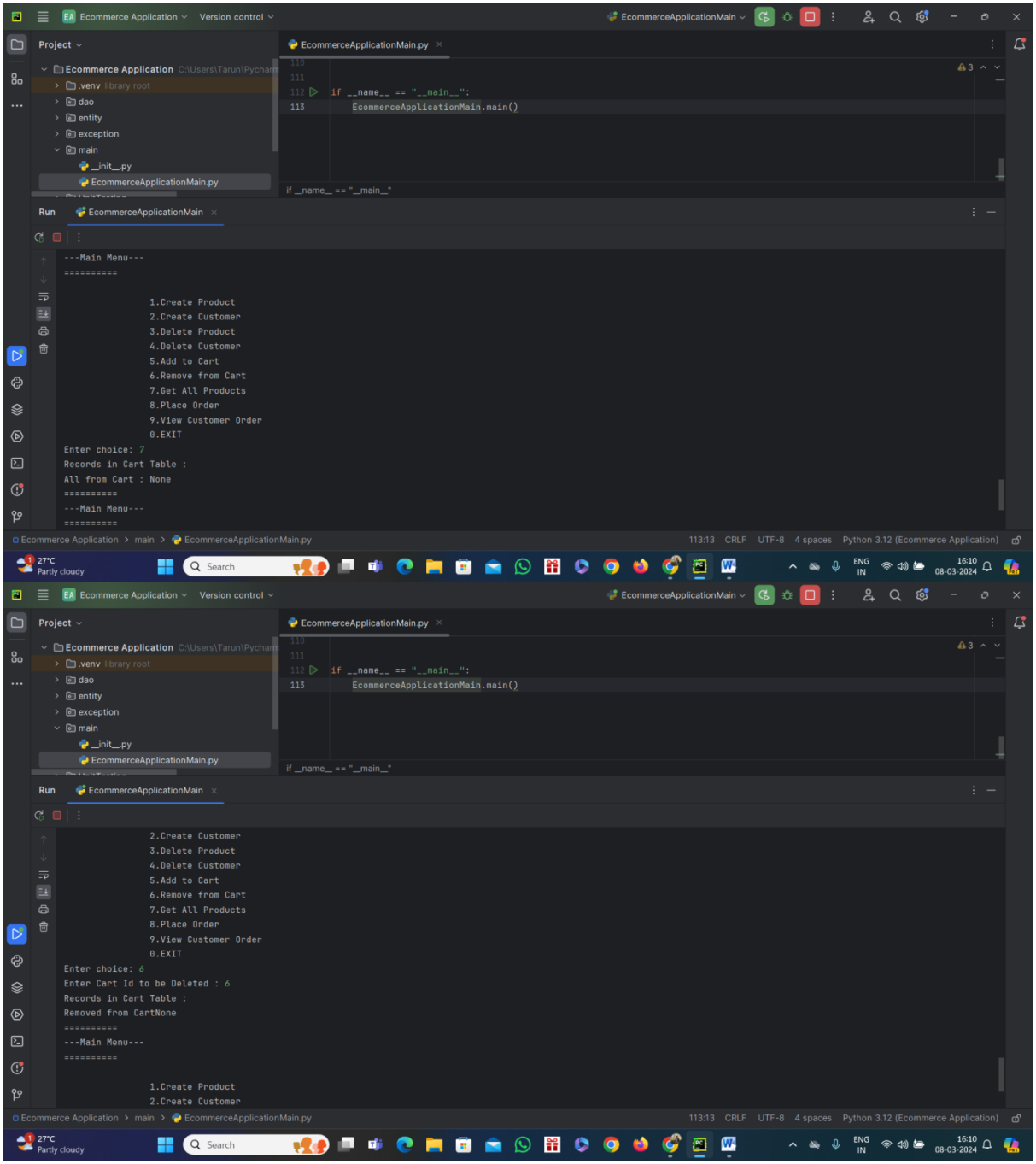
Unit Testing

```
import unittest
from unittest.mock import Mock
from dao.OrderProcessorRepository import OrderProcessorRepository
from dao.productsdao import ProductsDao
from dao.customersdao import CustomersDao



class TestOrderProcessorRepository(unittest.TestCase):
    def setUp(self):
        # Set up any necessary objects or mocks
        self.order_processor_repo = OrderProcessorRepository

    def test_create_product(self):
        # Mocking a product
        product = ProductsDao()

        # Mocking the repository's interaction with the database
        self.order_processor_repo.create_product = Mock(return_value=True)

        # Testing the create_product method
        result = self.order_processor_repo.create_product(product)

        self.assertTrue(result)

    def test_add_to_cart(self):

        customer = CustomersDao()
        product = ProductsDao()
        quantity = 3

        self.order_processor_repo.add_to_cart = Mock(return_value=True)
        result = self.order_processor_repo.add_to_cart(customer,
product,quantity)

        self.assertTrue(result)


if __name__ == '__main__':
    unittest.main()
```

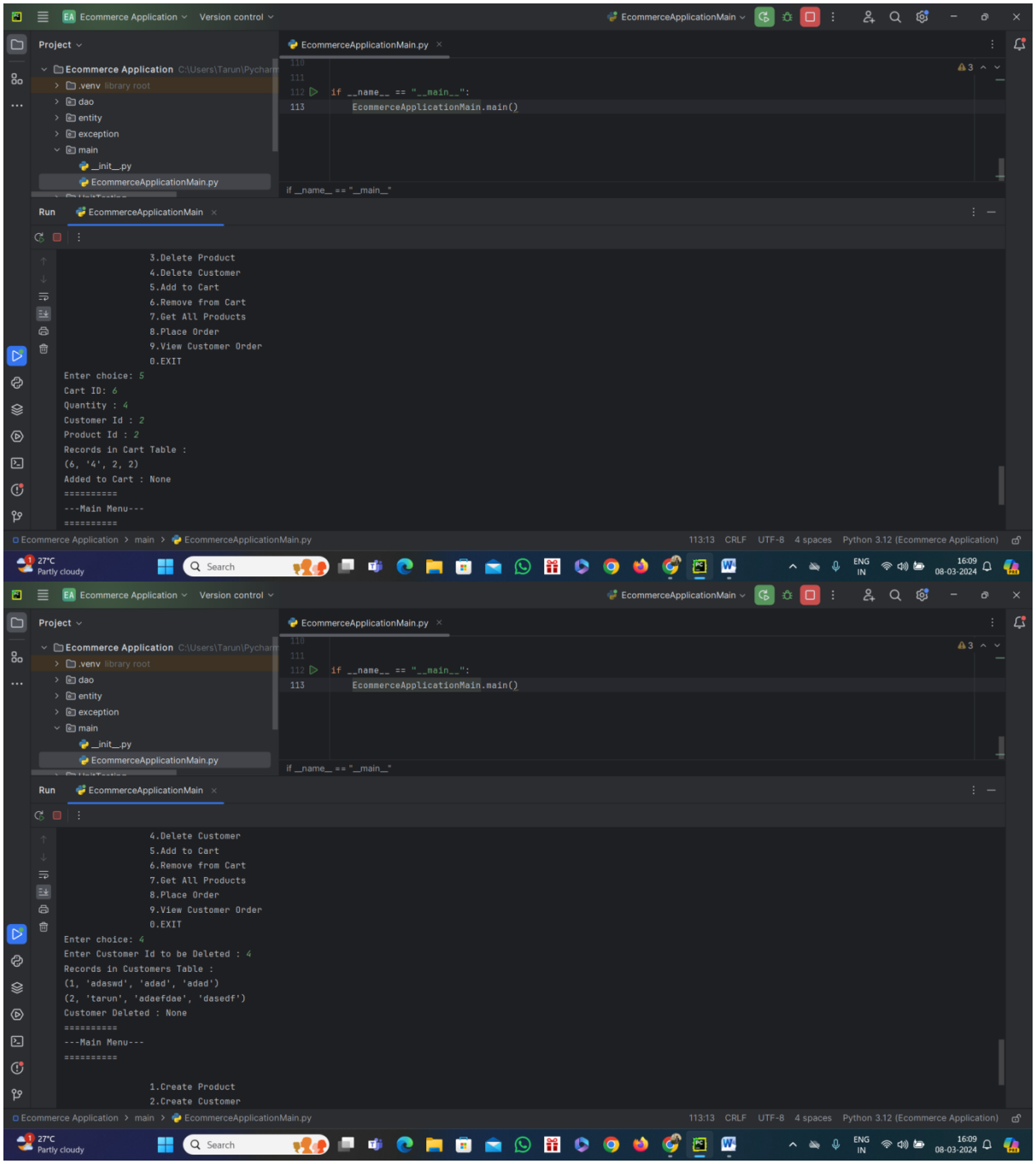
Some Screen Shots


Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 9
Enter Customer Id : 2
Customer Orders : [(1, 2, datetime.date(2012, 2, 2), 456.0, 'adsfcbghf'), (5, 2, datetime.date(2012, 2, 6), 78.0, 'gjufcgjxd'), (8, 2, datetime.date(4, 8, 12), 546.0, 'adawsd')]
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
3.Delete Product
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:11
08-03-2024

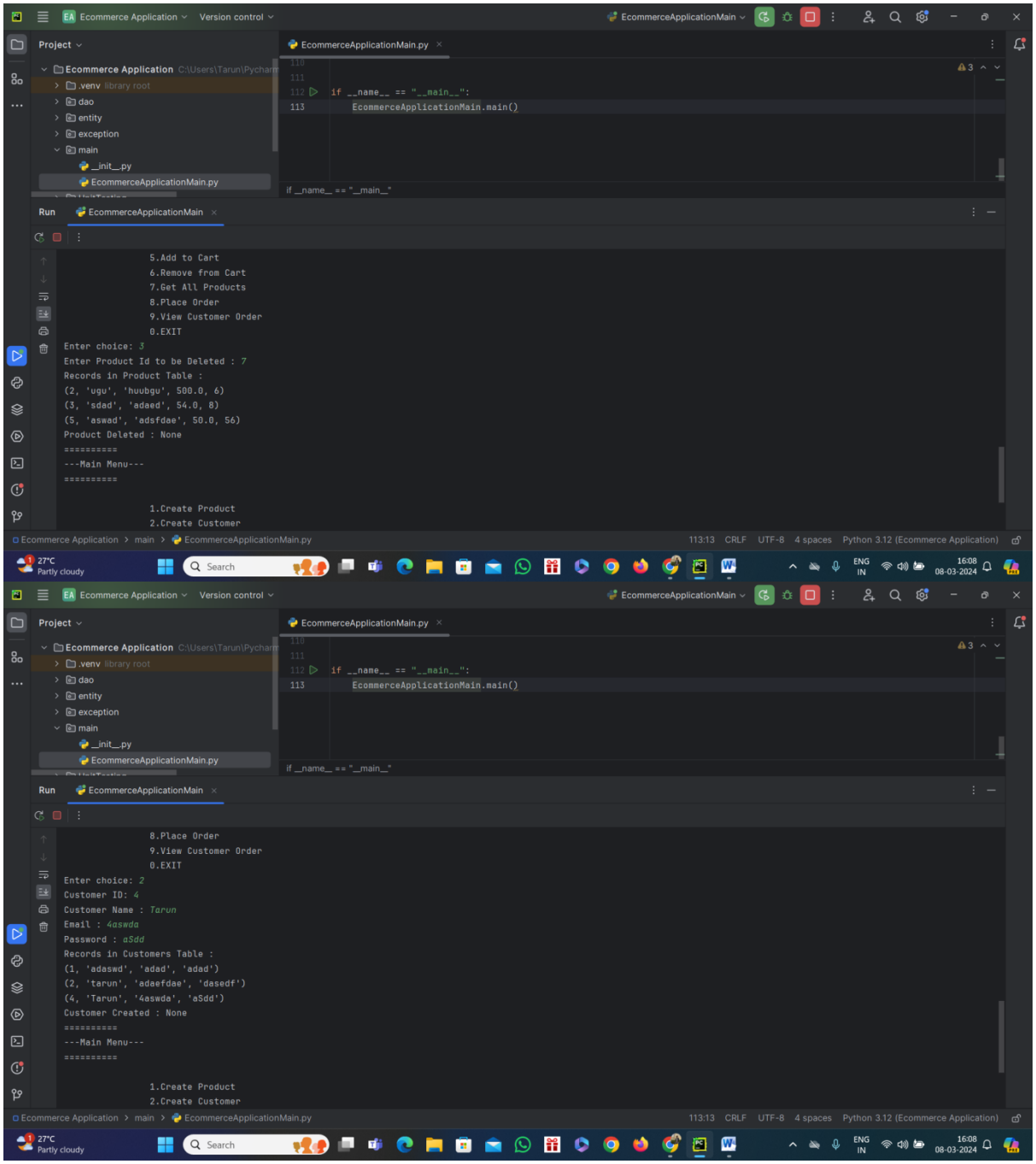
Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
9.View Customer Order
0.EXIT
Enter choice: 8
Order ID: 8
Customer Id : 2
Order Date : 4-8-12
Total Price : 546
Shipping Address : adawsd
Records in Orders Table :
(1, 2, datetime.date(2012, 2, 2), 456.0, 'adsfcbghf')
(5, 2, datetime.date(2012, 2, 6), 78.0, 'gjufcgjxd')
(8, 2, datetime.date(4, 8, 12), 546.0, 'adawsd')
Order Placed : True
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:10
08-03-2024


Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
---Main Menu---
==========
1.Create Product
2.Create Customer
3.Delete Product
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 7
Records in Cart Table :
All from Cart : None
==========
---Main Menu---
==========
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:10
08-03-2024

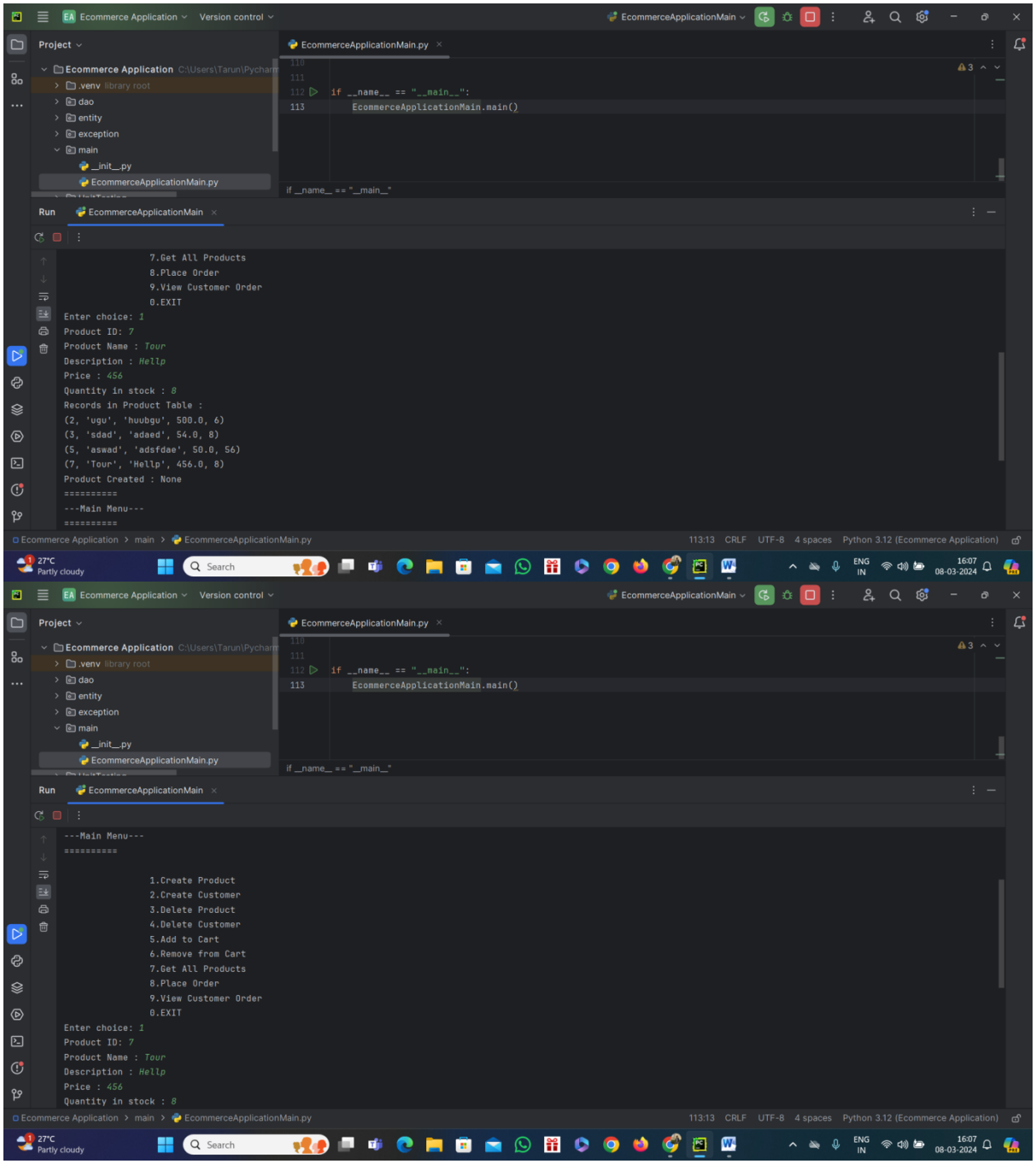
Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
2.Create Customer
3.Delete Product
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 6
Enter Cart Id to be Deleted : 6
Records in Cart Table :
Removed from CartNone
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:10
08-03-2024


Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
3.Delete Product
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 5
Cart ID: 6
Quantity : 4
Customer Id : 2
Product Id : 2
Records in Cart Table :
(6, '4', 2, 2)
Added to Cart : None
==========
---Main Menu---
==========
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:09
08-03-2024

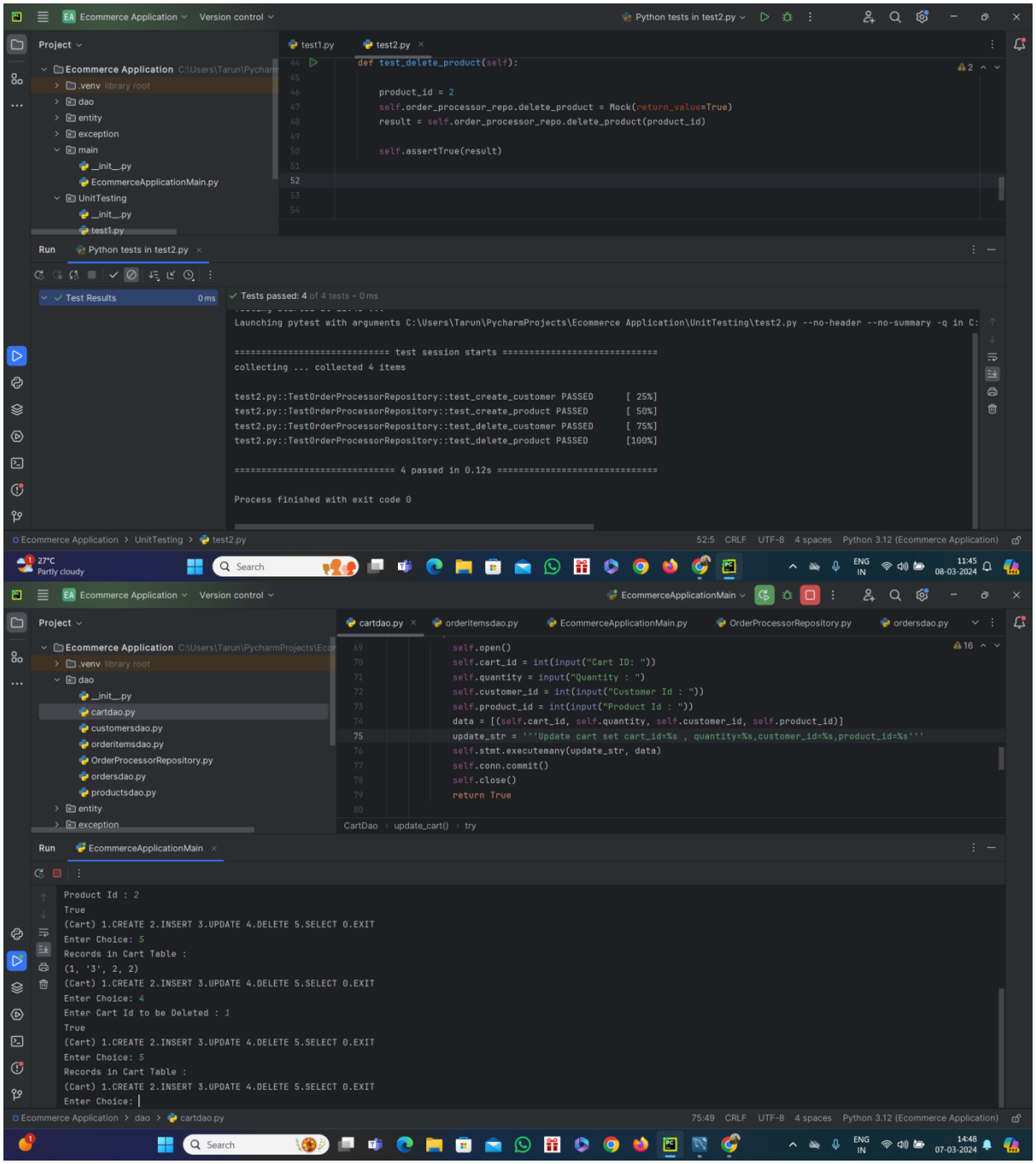
Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 4
Enter Customer Id to be Deleted : 4
Records in Customers Table :
(1, 'adaswd', 'adad', 'adad')
(2, 'tarun', 'adaefdae', 'dasedf')
Customer Deleted : None
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:09
08-03-2024


Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 3
Enter Product Id to be Deleted : 7
Records in Product Table :
(2, 'ugu', 'huubgu', 500.0, 6)
(3, 'sdad', 'adaed', 54.0, 8)
(5, 'aswad', 'adsfdae', 50.0, 56)
Product Deleted : None
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:08
08-03-2024

Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 2
Customer ID: 4
Customer Name : Tarun
Email : 4aswda
Password : aSdd
Records in Customers Table :
(1, 'adaswd', 'adad', 'adad')
(2, 'tarun', 'adaefdae', 'dasedf')
(4, 'Tarun', '4aswda', 'aSdd')
Customer Created : None
==========
---Main Menu---
==========
1.Create Product
2.Create Customer
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:08
08-03-2024


Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 1
Product ID: 7
Product Name : Tour
Description : Hellp
Price : 456
Quantity in stock : 8
Records in Product Table :
(2, 'ugu', 'huubgu', 500.0, 6)
(3, 'sdad', 'adaed', 54.0, 8)
(5, 'aswad', 'adsfdae', 50.0, 56)
(7, 'Tour', 'Hellp', 456.0, 8)
Product Created : None
==========
---Main Menu---
==========
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:07
08-03-2024

Ecommerce Application
Version control
EcommerceApplicationMain
Project
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
EcommerceApplicationMain.py
if __name__ == "__main__":
EcommerceApplicationMain.main()
if __name__ == "__main__"
Run
EcommerceApplicationMain
---Main Menu---
==========
1.Create Product
2.Create Customer
3.Delete Product
4.Delete Customer
5.Add to Cart
6.Remove from Cart
7.Get All Products
8.Place Order
9.View Customer Order
0.EXIT
Enter choice: 1
Product ID: 7
Product Name : Tour
Description : Hellp
Price : 456
Quantity in stock : 8
Ecommerce Application > main > EcommerceApplicationMain.py
113:13 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
16:07
08-03-2024


Ecommerce Application
Version control
Python tests in test2.py
Project
test1.py
test2.py
def test_delete_product(self):
product_id = 2
self.order_processor_repo.delete_product = Mock(return_value=True)
result = self.order_processor_repo.delete_product(product_id)
self.assertTrue(result)
Ecommerce Application C:\Users\Tarun\Pycharm
.venv library root
dao
entity
exception
main
__init__.py
EcommerceApplicationMain.py
UnitTesting
__init__.py
test1.py
Run
Python tests in test2.py
Test Results 0 ms
Tests passed: 4 of 4 tests – 0 ms
Launching pytest with arguments C:\Users\Tarun\PycharmProjects\Ecommerce Application\UnitTesting\test2.py --no-header --no-summary -q in C:
============================= test session starts ==============================
collecting ... collected 4 items
test2.py::TestOrderProcessorRepository::test_create_customer PASSED [25%]
test2.py::TestOrderProcessorRepository::test_create_product PASSED [50%]
test2.py::TestOrderProcessorRepository::test_delete_customer PASSED [75%]
test2.py::TestOrderProcessorRepository::test_delete_product PASSED [100%]
============================== 4 passed in 0.12s ===============================
Process finished with exit code 0
Ecommerce Application > UnitTesting > test2.py
52:5 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
27°C
Partly cloudy
Search
ENG
IN
11:45
08-03-2024

Ecommerce Application
Version control
EcommerceApplicationMain
Project
cartdao.py
orderitemsdao.py
EcommerceApplicationMain.py
OrderProcessorRepository.py
ordersdao.py
self.open()
self.cart_id = int(input("Cart ID: "))
self.quantity = input("Quantity : ")
self.customer_id = int(input("Customer Id : "))
self.product_id = int(input("Product Id : "))
data = [(self.cart_id, self.quantity, self.customer_id, self.product_id)]
update_str = '''Update cart set cart_id=%s , quantity=%s,customer_id=%s,product_id=%s'''
self.stmt.executemany(update_str, data)
self.conn.commit()
self.close()
return True
Ecommerce Application C:\Users\Tarun\PycharmProjects\Ecor
.venv library root
dao
__init__.py
cartdao.py
customersdao.py
orderitemsdao.py
OrderProcessorRepository.py
ordersdao.py
productsdao.py
entity
exception
CartDao > update_cart() > try
Run
EcommerceApplicationMain
Product Id : 2
True
(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Cart Table :
(1, '3', 2, 2)
(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 4
Enter Cart Id to be Deleted : 1
True
(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice: 5
Records in Cart Table :
(Cart) 1.CREATE 2.INSERT 3.UPDATE 4.DELETE 5.SELECT 0.EXIT
Enter Choice:
Ecommerce Application > dao > cartdao.py
75:49 CRLF UTF-8 4 spaces Python 3.12 (Ecommerce Application)
Search
ENG
IN
14:48
07-03-2024