

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

MODELS

ORDERS.PY

```
from db_connector.db_adapter import get_db_connection

class Order:

    def __init__(self, order_id, user_id, order_date, total_price,
shipping_address):
        self.connection = get_db_connection()
        self.__order_id = order_id
        self.__user_id = user_id
        self.__order_date = order_date
        self.__total_price = total_price
        self.__shipping_address = shipping_address

    def get_order_id(self):
        return self.__order_id

    def get_customer_id(self):
        return self.__user_id

    def get_order_date(self):
        return self.__order_date

    def get_total_price(self):
        return self.__total_price

    def get_shipping_address(self):
        return self.__shipping_address

    def update_order_info(self, user_id=None, order_date=None,
total_price=None, shipping_address=None):
        try:
            my_cursor = self.connection.cursor()

            if user_id:
                sql = '''
                UPDATE Orders SET userId = %s WHERE order_id = %s
                '''
                para = (user_id, self.__order_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('User ID updated successfully')

            if order_date:
                sql = '''
                UPDATE Orders SET order_date = %s WHERE order_id = %s
                '''
                para = (order_date, self.__order_id)
                my_cursor.execute(sql, para)
                self.connection.commit()
                print('Order date updated successfully')

            if total_price:
                sql = '''
                UPDATE Orders SET total_price = %s WHERE order_id = %s
                '''
                para = (total_price, self.__order_id)
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Total price updated successfully')

    if shipping_address:
        sql = '''
        UPDATE Orders SET shipping_address = %s WHERE order_id = %s
        '''
        para = (shipping_address, self.__order_id)
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Shipping address updated successfully')

    except Exception as e:
        print(f'An error occurred: {e}')

    def print_order_info(self):
        print(f'Order ID: {self.__order_id}')
        print(f'User ID: {self.__user_id}')
        print(f'Order Date: {self.__order_date}')
        print(f'Total Price: {self.__total_price}')
        print(f'Shipping Address: {self.__shipping_address}')
```

product.py

```
from db_connector.db_adapter import *

class Product:

    def __init__(self, product_id, product_name, description, price,
quantity_in_stock, product_type):
        self.connection = get_db_connection()
        self.__productId = product_id
        self.__product_name = product_name
        self.__description = description
        self.__price = price
        self.__quantity_in_stock = quantity_in_stock
        self.__type = product_type

    def get_product_name(self):
        return self.__product_name

    def get_description(self):
        return self.__description

    def get_price(self):
        return self.__price

    def get_quantity_in_stock(self):
        return self.__quantity_in_stock

    def get_product_type(self):
        return self.__type

    def update_product_info(self, product_name=None, description=None,
price=None, quantity_in_stock=None, product_type=None):
        try:
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
my_cursor = self.connection.cursor()

if product_name:
    sql = '''
    UPDATE Product SET productName = %s WHERE productId = %s
    '''
    para = (product_name, self.__productId)
    my_cursor.execute(sql, para)
    self.connection.commit()
    print('Product Name updated successfully')

if description:
    sql = '''
    UPDATE Product SET description = %s WHERE productId = %s
    '''
    para = (description, self.__productId)
    my_cursor.execute(sql, para)
    self.connection.commit()
    print('Product description updated successfully')

if price:
    sql = '''
    UPDATE Product SET price = %s WHERE productId = %s
    '''
    para = (price, self.__productId)
    my_cursor.execute(sql, para)
    self.connection.commit()
    print('Product Price updated successfully')

if quantity_in_stock:
    sql = '''
    UPDATE Product SET quantityInStock = %s WHERE productId =
%s
    '''
    para = (quantity_in_stock, self.__productId)
    my_cursor.execute(sql, para)
    self.connection.commit()
    print('Product Quantity_In_Stock updated successfully')

if product_type:
    sql = '''
    UPDATE Product SET type = %s WHERE productId = %s
    '''
    para = (product_type, self.__productId)
    my_cursor.execute(sql, para)
    self.connection.commit()
    print('Product Type updated successfully')

except Exception as e:
    print(f'An error occurred: {e}')
```

user.py

```
from db_connector.db_adapter import *

class User:
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
def __init__(self, userid, username, password, role):
    self.connection = get_db_connection()
    self.__userId = userid
    self.__user_name = username
    self.__password = password
    self.__role = role

def get_user_id(self):
    return self.__userId

def get_user_name(self):
    return self.__user_name

def get_password(self):
    return self.__password

def get_user_role(self):
    return self.__role

def update_user_info(self, username=None, password=None, role=None):
    try:
        my_cursor = self.connection.cursor()

        if username:
            sql = '''
            UPDATE User SET username = %s WHERE userID = %s
            '''
            para = (username, self.__userId)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('User Name updated successfully')

        if password:
            sql = '''
            UPDATE User SET password = %s WHERE userID = %s
            '''
            para = (password, self.__userId)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('User Password updated successfully')

        if role:
            sql = '''
            UPDATE User SET role = %s WHERE userID = %s
            '''
            para = (role, self.__userId)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('User Role updated successfully')

    except Exception as e:
        print(f'An error occurred: {e}')

def print_user_info(self):
    print('UserID', self.__userId)
    print('User Name', self.__user_name)
    print('Password', self.__password)
    print('Role', self.__role)
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

CUSTOM_EXCEPTION

Custom_exception.py

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

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

DB_CONNECTOR

Db_adapter.py

```
import mysql.connector

def get_db_connection():

    config = {
        'user': 'root',
        'password': 'root',
        'host': 'localhost',
        'database': 'omsdb'
    }

    try:
        connection = mysql.connector.connect(**config)
        # print("Connected to the database")
        # print('hello World')
        return connection
    except mysql.connector.Error as err:
        print(f"Error: {err}")
        return None

def get_ids(table_name, id_column_name):
    mydb = get_db_connection()
    my_cursor = mydb.cursor()
    sql = 'SELECT ' + id_column_name + ' FROM ' + table_name + ' ORDER BY ' + id_column_name + ' DESC LIMIT 1'
    # print(sql)
    my_cursor.execute(sql)
    x = list(my_cursor.fetchone())[0]
    return int(x) + 1
```

SERVICES

Order_manager.py

```
from datetime import date

from db_connector.db_adapter import *
from models.orders import Order
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
from models.product import Product
from models.user import User
from custom_exception.custom_exceptions import *

class OrderManager:

    def __init__(self):
        self.connection = get_db_connection()

    def get_all_products(self):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * FROM Product
            '''
            my_cursor.execute(sql)
            x = [Product(*list(i)) for i in list(my_cursor.fetchall())]
            return x
        except Exception as e:
            print(f'An error occurred: {e}')

    def get_order_by_user(self, user_id):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                SELECT * FROM Orders WHERE userID = %s
            '''
            para = (user_id,)
            my_cursor.execute(sql, para)
            x = [Order(*list(i)) for i in list(my_cursor.fetchall())]
            return x
        except Exception as e:
            print(f'An error occurred: {e}')

    def create_user(self, user_id, user_name, password, role):
        try:
            my_cursor = self.connection.cursor()
            sql = '''
                INSERT INTO User(userID, username, password, role)
                VALUES(%s, %s, %s, %s)
            '''
            para = (user_id, user_name, password, role)
            my_cursor.execute(sql, para)
            self.connection.commit()
            print('User Created successfully')
        except Exception as e:
            print(f'An error occurred: {e}')

    def create_product(self, user, product_name, description, price,
quantity_in_stock, product_type):
        try:
            my_cursor = self.connection.cursor()
            if user.get_user_role() != 'Admin':
                print('Product creation not permitted (Not admin)')
                return
            sql = '''
                INSERT INTO Product(productId, productName,
description, price, quantityInStock, type)
                VALUES(%s, %s, %s, %s, %s, %s)
            '''
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
        para = (get_ids('product', 'productId'), product_name,
description, price, quantity_in_stock, product_type)
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Product Created successfully')
    except Exception as e:
        print(f'An error occurred: {e}')

def user_exists(self, user_id):
    my_cursor = self.connection.cursor()
    sql = '''
select count(*) from user where userId = %s
'''
    para = (user_id,)
    my_cursor.execute(sql, para)
    x = int(my_cursor.fetchone()[0])
    return x != 0

def order_exists(self, order_id):
    my_cursor = self.connection.cursor()
    sql = '''
select count(*) from Orders where order_id = %s
'''
    para = (order_id,)
    my_cursor.execute(sql, para)
    x = int(my_cursor.fetchone()[0])
    return x != 0

def get_user_by_id(self, user_id):
    try:
        if not self.user_exists(user_id):
            raise UserNotFound('The specified user does not exist')
        my_cursor = self.connection.cursor()
        sql = '''
SELECT * FROM User WHERE userId = %s
'''
        para = (user_id,)
        my_cursor.execute(sql, para)
        x = list(my_cursor.fetchone())
        return User(*x)
    except UserNotFound as e:
        print(f'User Not Found: {e}')
    except Exception as e:
        print(f'An error occurred: {e}')

def get_order_by_id(self, order_id):
    try:
        if not self.order_exists(order_id):
            raise OrderNotFound('The specified order does not exist')
        my_cursor = self.connection.cursor()
        sql = '''
SELECT * FROM Orders WHERE order_id = %s
'''
        para = (order_id,)
        my_cursor.execute(sql, para)
        x = list(my_cursor.fetchone())
        return Order(*x)
    except OrderNotFound as e:
        print(f'Order Not Found: {e}')
    except Exception as e:
        print(f'An error occurred: {e}')
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
def create_order(self, user_id, product, shipping_address):
    try:
        my_cursor = self.connection.cursor()
        user_data = self.get_user_by_id(user_id)

        if product.get_quantity_in_stock() == 0:
            print('Product Unavailable')
            return

        sql = '''
        INSERT INTO Orders(order_id, userId, order_date, total_price,
shipping_address)
        VALES(%s, %s, %s, %s, %s)
        '''
        para = (
            get_ids('Orders', 'order_id'), user_data.get_user_id(),
date.today(), product.get_price(), shipping_address)
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Order Created Successfully')
    except Exception as e:
        print(f'An error occurred: {e}')

def cancel_order(self, user_id, order_id):
    try:
        my_cursor = self.connection.cursor()
        sql = '''
        DELETE FROM Orders WHERE order_id = %s AND userId = %s
        '''
        para = (order_id, user_id)
        my_cursor.execute(sql, para)
        self.connection.commit()
        print('Order Cancelled successfully')
    except Exception as e:
        print(f'An error occurred: {e}')
```

MAIN

Main.py

```
from services.order_manager import OrderManager
from custom_exception.custom_exceptions import UserNotFound, OrderNotFound
class OrderManagement:

    @staticmethod
    def print_menu():
        print("\nOrder Management System Menu:")
        print("1. Create User")
        print("2. Create Product")
        print("3. Cancel Order")
        print("4. Get All Products")
        print("5. Get Order by User")
        print("6. Exit")

    @staticmethod
    def get_user_input():
```


GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
        return input("\nEnter your choice (1-6): ")

    @staticmethod
    def create_user(order_manager):
        user_id = input("Enter User ID: ")
        username = input("Enter Username: ")
        password = input("Enter Password: ")
        role = input("Enter Role (Admin/User): ")

        order_manager.create_user(user_id, username, password, role)

    @staticmethod
    def create_product(order_manager):
        admin_user_id = input("Enter Admin User ID: ")
        admin_user = order_manager.get_user_by_id(admin_user_id)

        if admin_user and admin_user.get_user_role() == 'Admin':
            product_name = input("Enter Product Name: ")
            description = input("Enter Product Description: ")
            price = float(input("Enter Product Price: "))
            quantity_in_stock = int(input("Enter Quantity in Stock: "))
            product_type = input("Enter Product Type\n(Electronics/Clothing): ")

            order_manager.create_product(admin_user, product_name,
description, price, quantity_in_stock, product_type)
        else:
            print("Invalid Admin User ID or User is not an admin.")

    @staticmethod
    def cancel_order(order_manager):
        user_id = input("Enter User ID: ")
        order_id = input("Enter Order ID: ")

        try:
            order_manager.cancel_order(user_id, order_id)
        except OrderNotFound as e:
            print(e)
        except UserNotFound as e:
            print(e)

    @staticmethod
    def get_all_products(order_manager):
        products = order_manager.get_all_products()
        print("\nAll Products:")
        for product in products:
            print(f"{product.get_product_name()} - {product.get_price()} -\n{product.get_quantity_in_stock()} in stock")

    @staticmethod
    def get_order_by_user(order_manager):
        user_id = input("Enter User ID: ")
        orders = order_manager.get_order_by_user(user_id)

        if orders:
            print(f"\nOrders for User ID {user_id}:")
            for order in orders:
                print(f"Order ID: {order.get_order_id()}, Date:\n{order.get_order_date()}, Total Price: {order.get_total_price()}")
        else:
            print(f"No orders found for User ID {user_id}")
```

GAUTAM SHARMA
ORDER_MANAGEMENT_SYSTEM

```
@staticmethod
def main():
    order_manager = OrderManager()

    while True:
        OrderManagement.print_menu()
        choice = OrderManagement.get_user_input()

        if choice == "1":
            OrderManagement.create_user(order_manager)
        elif choice == "2":
            OrderManagement.create_product(order_manager)
        elif choice == "3":
            OrderManagement.cancel_order(order_manager)
        elif choice == "4":
            OrderManagement.get_all_products(order_manager)
        elif choice == "5":
            OrderManagement.get_order_by_user(order_manager)
        elif choice == "6":
            print("Exiting Order Management System.")
            break
        else:
            print("Invalid choice. Please enter a number between 1 and 6.")

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

output screenshots-

1) Create User-

```
Enter your choice (1-6): 1
Enter User ID: 10
Enter Username: Gautam
Enter Password: root
Enter Role (Admin/User): user
User Created successfully

Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit
```

2) Create product

```
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit

Enter your choice (1-6): 2
Enter Admin User ID: 1
Enter Product Name: blackberry
Enter Product Description: phone
Enter Product Price: 1090909
Enter Quantity in Stock: 10
Enter Product Type (Electronics/Clothing): electronics
```

3:Canceling the Order:

```
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit

Enter your choice (1-6): 3
Enter User ID: 2
Enter Order ID: 1
Order Cancelled successfully
```

4: Get all products:

```
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit

Enter your choice (1-6): 4

All Products:
Laptop - 1200 - 50 in stock
Coffee Maker - 80 - 30 in stock
Running Shoes - 70 - 100 in stock
Bluetooth Speaker - 50 - 20 in stock
```

5: Get order by user:

```
Enter your choice (1-6): 5
Enter User ID: 77

No orders found for User ID 77
```

6: Exit

```
Order Management System Menu:
1. Create User
2. Create Product
3. Cancel Order
4. Get All Products
5. Get Order by User
6. Exit

Enter your choice (1-6): 6
Exiting Order Management System.

Process finished with exit code 0
```

7:

