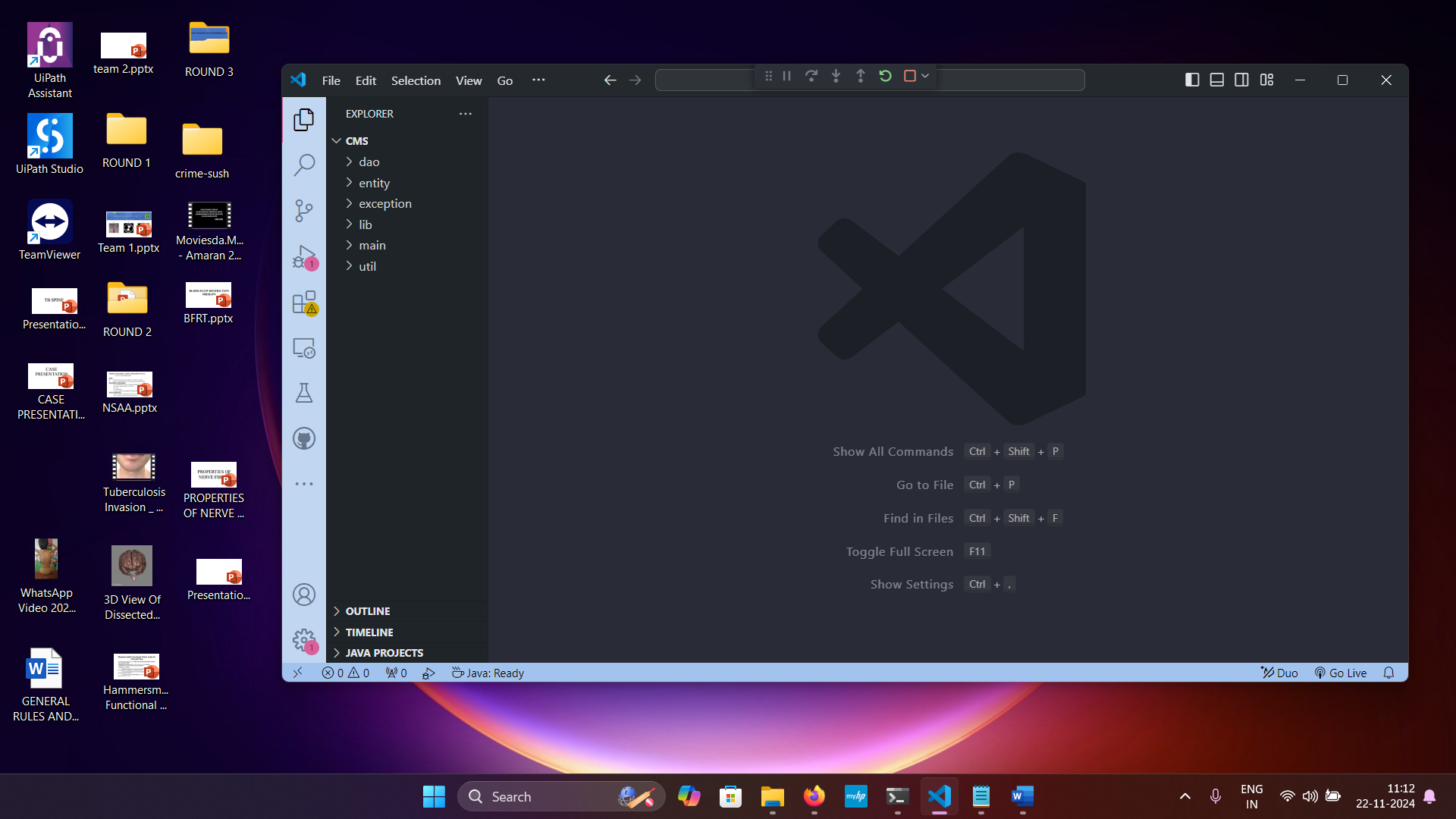
**Java Coding Challenge: Order Management System**

* **J227 Sushmith K**

**Directory structure:**



**Created database named OMS and created tables:**



**entity/Product class:**

package entity;

public class Product {

    private int productId;

    private String productName;

    private String description;

    private double price;

    private int quantityInStock;

    private String type;

    public Product(int productId, String productName, String description, double price, int quantityInStock, String type) {

        this.productId = productId;

        this.productName = productName;

        this.description = description;

        this.price = price;

        this.quantityInStock = quantityInStock;

        this.type = type;

    }

    public int getProductId() { return productId; }

    public void setProductId(int productId) { this.productId = productId; }

    public String getProductName() { return productName; }

    public void setProductName(String productName) { this.productName = productName; }

    public String getDescription() { return description; }

    public void setDescription(String description) { this.description = description; }

    public double getPrice() { return price; }

    public void setPrice(double price) { this.price = price; }

    public int getQuantityInStock() { return quantityInStock; }

    public void setQuantityInStock(int quantityInStock) { this.quantityInStock = quantityInStock; }

    public String getType() { return type; }

    public void setType(String type) { this.type = type; }

}

**entity/Electronics class:**

package entity;

public class Electronics extends Product {

    private String brand;

    private int warrantyPeriod;

    public Electronics(int productId, String productName, String description, double price, int quantityInStock, String brand, int warrantyPeriod) {

        super(productId, productName, description, price, quantityInStock, "Electronics");

        this.brand = brand;

        this.warrantyPeriod = warrantyPeriod;

    }

    public String getBrand() { return brand; }

    public void setBrand(String brand) { this.brand = brand; }

    public int getWarrantyPeriod() { return warrantyPeriod; }

    public void setWarrantyPeriod(int warrantyPeriod) { this.warrantyPeriod = warrantyPeriod; }

}

**entity/Clothing class:**

package entity;

public class Clothing extends Product {

    private String size;

    private String color;

    public Clothing(int productId, String productName, String description, double price, int quantityInStock, String size, String color) {

        super(productId, productName, description, price, quantityInStock, "Clothing");

        this.size = size;

        this.color = color;

    }

    public String getSize() { return size; }

    public void setSize(String size) { this.size = size; }

    public String getColor() { return color; }

    public void setColor(String color) { this.color = color; }

}

**entity/User class:**

package entity;

public class User {

    private int userId;

    private String username;

    private String password;

    private String role;

    public User(int userId, String username, String password, String role) {

        this.userId = userId;

        this.username = username;

        this.password = password;

        this.role = role;

    }

    public int getUserId() { return userId; }

    public void setUserId(int userId) { this.userId = userId; }

    public String getUsername() { return username; }

    public void setUsername(String username) { this.username = username; }

    public String getPassword() { return password; }

    public void setPassword(String password) { this.password = password; }

    public String getRole() { return role; }

    public void setRole(String role) { this.role = role; }

}

**dao/IOrderManagementRepository:**

package dao;

import entity.Product;

import entity.User;

import exception.UserNotFoundException;

import exception.OrderNotFoundException;

import java.util.List;

public interface IOrderManagementRepository {

    void createOrder(User user, List<Product> products) throws UserNotFoundException;

    void cancelOrder(int userId, int orderId) throws UserNotFoundException, OrderNotFoundException;

    void createProduct(User user, Product product) throws UserNotFoundException;

    void createUser(User user);

    List<Product> getAllProducts();

    List<Product> getOrderByUser(User user);

}

**dao/impl/OrderProcessor:**

package dao.impl;

import dao.IOrderManagementRepository;

import entity.Product;

import entity.User;

import exception.UserNotFoundException;

import exception.OrderNotFoundException;

import util.DBUtil;

import java.sql.\*;

import java.util.ArrayList;

import java.util.List;

public class OrderProcessor implements IOrderManagementRepository {

    @Override

    public void createOrder(User user, List<Product> products) throws UserNotFoundException {

        if (user == null) {

            throw new UserNotFoundException("User not found");

        }

        try (Connection connection = DBUtil.getDBConn()) {

            for (Product product : products) {

                String query = "INSERT INTO orders (userId, productId, quantity) VALUES (?, ?, ?)";

                try (PreparedStatement statement = connection.prepareStatement(query)) {

                    statement.setInt(1, user.getUserId());

                    statement.setInt(2, product.getProductId());

                    statement.setInt(3, 1);

                    statement.executeUpdate();

                }

            }

            System.out.println("Order created successfully!");

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    @Override

    public void cancelOrder(int userId, int orderId) throws UserNotFoundException, OrderNotFoundException {

        try (Connection connection = DBUtil.getDBConn()) {

            String checkUserQuery = "SELECT \* FROM users WHERE userId = ?";

            try (PreparedStatement userStatement = connection.prepareStatement(checkUserQuery)) {

                userStatement.setInt(1, userId);

                ResultSet rs = userStatement.executeQuery();

                if (!rs.next()) {

                    throw new UserNotFoundException("User not found");

                }

            }

            String checkOrderQuery = "SELECT \* FROM orders WHERE orderId = ?";

            try (PreparedStatement orderStatement = connection.prepareStatement(checkOrderQuery)) {

                orderStatement.setInt(1, orderId);

                ResultSet rs = orderStatement.executeQuery();

                if (!rs.next()) {

                    throw new OrderNotFoundException("Order not found");

                }

            }

            String cancelOrderQuery = "DELETE FROM orders WHERE orderId = ?";

            try (PreparedStatement cancelStatement = connection.prepareStatement(cancelOrderQuery)) {

                cancelStatement.setInt(1, orderId);

                cancelStatement.executeUpdate();

            }

            System.out.println("Order canceled successfully!");

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    @Override

    public void createProduct(User user, Product product) throws UserNotFoundException {

        if (user == null || !user.getRole().equals("Admin")) {

            throw new UserNotFoundException("Admin user is required to add products");

        }

        try (Connection connection = DBUtil.getDBConn()) {

            String query = "INSERT INTO products (productId, productName, description, price, quantityInStock, type) VALUES (?, ?, ?, ?, ?, ?)";

            try (PreparedStatement statement = connection.prepareStatement(query)) {

                statement.setInt(1, product.getProductId());

                statement.setString(2, product.getProductName());

                statement.setString(3, product.getDescription());

                statement.setDouble(4, product.getPrice());

                statement.setInt(5, product.getQuantityInStock());

                statement.setString(6, product.getType());

                statement.executeUpdate();

            }

            System.out.println("Product created successfully!");

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    @Override

    public void createUser(User user) {

        try (Connection connection = DBUtil.getDBConn()) {

            String query = "INSERT INTO users (userId, username, password, role) VALUES (?, ?, ?, ?)";

            try (PreparedStatement statement = connection.prepareStatement(query)) {

                statement.setInt(1, user.getUserId());

                statement.setString(2, user.getUsername());

                statement.setString(3, user.getPassword());

                statement.setString(4, user.getRole());

                statement.executeUpdate();

            }

            System.out.println("User created successfully!");

        } catch (SQLException e) {

            e.printStackTrace();

        }

    }

    @Override

    public List<Product> getAllProducts() {

        List<Product> products = new ArrayList<>();

        try (Connection connection = DBUtil.getDBConn()) {

            String query = "SELECT \* FROM products";

            try (PreparedStatement statement = connection.prepareStatement(query)) {

                ResultSet rs = statement.executeQuery();

                while (rs.next()) {

                    Product product = new Product(

                            rs.getInt("productId"),

                            rs.getString("productName"),

                            rs.getString("description"),

                            rs.getDouble("price"),

                            rs.getInt("quantityInStock"),

                            rs.getString("type")

                    );

                    products.add(product);

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return products;

    }

    @Override

    public List<Product> getOrderByUser(User user) {

        List<Product> products = new ArrayList<>();

        try (Connection connection = DBUtil.getDBConn()) {

            String query = "SELECT p.productId, p.productName, p.description, p.price, p.quantityInStock, p.type FROM orders o JOIN products p ON o.productId = p.productId WHERE o.userId = ?";

            try (PreparedStatement statement = connection.prepareStatement(query)) {

                statement.setInt(1, user.getUserId());

                ResultSet rs = statement.executeQuery();

                while (rs.next()) {

                    Product product = new Product(

                            rs.getInt("productId"),

                            rs.getString("productName"),

                            rs.getString("description"),

                            rs.getDouble("price"),

                            rs.getInt("quantityInStock"),

                            rs.getString("type")

                    );

                    products.add(product);

                }

            }

        } catch (SQLException e) {

            e.printStackTrace();

        }

        return products;

    }

}

**DBUtil:**  
package util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.SQLException;

public class DBUtil {

    public static Connection getDBConn() throws SQLException {

        String url = "jdbc:mysql://localhost:3306/OMS";

        String username = "root";

        String password = "Sushmith@13";

        return DriverManager.getConnection(url, username, password);

    }

}

**main/OrderManagement main class:**

package main;

import dao.impl.OrderProcessor;

import entity.Product;

import entity.User;

import exception.UserNotFoundException;

import exception.OrderNotFoundException;

import java.util.List;

import java.util.Scanner;

public class OrderManagement {

    static OrderProcessor orderProcessor = new OrderProcessor();

    public static void main(String[] args) {

        Scanner scanner = new Scanner(System.in);

        boolean exit = false;

        while (!exit) {

            System.out.println("Order Management System");

            System.out.println("1. Create User");

            System.out.println("2. Create Product");

            System.out.println("3. Place Order");

            System.out.println("4. Cancel Order");

            System.out.println("5. Get All Products");

            System.out.println("6. Get Order By User");

            System.out.println("7. Exit");

            System.out.print("Enter your choice: ");

            int choice = scanner.nextInt();

            scanner.nextLine(); // Consume newline

            try {

                switch (choice) {

                    case 1:

                        createUser(scanner);

                        break;

                    case 2:

                        createProduct(scanner);

                        break;

                    case 3:

                        placeOrder(scanner);

                        break;

                    case 4:

                        cancelOrder(scanner);

                        break;

                    case 5:

                        getAllProducts();

                        break;

                    case 6:

                        getOrderbyUser(scanner);

                        break;

                    case 7:

                        System.out.println("Exiting the system...");

                        exit = true;

                        break;

                    default:

                        System.out.println("Invalid choice, please try again.");

                }

            } catch (UserNotFoundException | OrderNotFoundException e) {

                System.out.println("Error: " + e.getMessage());

            }

        }

        scanner.close();

    }

    private static void createUser(Scanner scanner) throws UserNotFoundException {

        System.out.print("Enter user ID: ");

        int userId = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter username: ");

        String username = scanner.nextLine();

        System.out.print("Enter password: ");

        String password = scanner.nextLine();

        System.out.print("Enter role (User/Admin): ");

        String role = scanner.nextLine();

        User user = new User(userId, username, password, role);

        orderProcessor.createUser(user);

        System.out.println("User created successfully!");

    }

    private static void createProduct(Scanner scanner) throws UserNotFoundException {

        System.out.print("Enter admin ID: ");

        int adminId = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter product name: ");

        String productName = scanner.nextLine();

        System.out.print("Enter product description: ");

        String productDescription = scanner.nextLine();

        System.out.print("Enter product price: ");

        double price = scanner.nextDouble();

        System.out.print("Enter product stock quantity: ");

        int stockQuantity = scanner.nextInt();

        scanner.nextLine();

        System.out.print("Enter product category: ");

        String category = scanner.nextLine();

        User admin = new User(adminId, "Admin", "admin123", "Admin");

        Product product = new Product(0, productName, productDescription, price, stockQuantity, category);

        orderProcessor.createProduct(admin, product);

        System.out.println("Product created successfully!");

    }

    private static void cancelOrder(Scanner scanner) throws OrderNotFoundException, UserNotFoundException {

        System.out.print("Enter user ID: ");

        int userId = scanner.nextInt();

        System.out.print("Enter order ID: ");

        int orderId = scanner.nextInt();

        orderProcessor.cancelOrder(userId, orderId);

        System.out.println("Order cancelled successfully!");

    }

    private static void getAllProducts() {

        List<Product> products = orderProcessor.getAllProducts();

        System.out.println("All Products:");

        for (Product product : products) {

            System.out.println(product.getProductName());

        }

    }

    private static void getOrderbyUser(Scanner scanner) {

        System.out.print("Enter user ID: ");

        int userId = scanner.nextInt();

        User user = new User(userId, "User", "password", "User");

        List<Product> products = orderProcessor.getOrderByUser(user);

        System.out.println("Orders by " + user.getUsername() + ":");

        for (Product product : products) {

            System.out.println(product.getProductName());

        }

    }

    private static void placeOrder(Scanner scanner) throws UserNotFoundException {

        System.out.print("Enter your user ID: ");

        int userId = scanner.nextInt();

        scanner.nextLine();

        User user = new User(userId, "User", "password", "User");

        List<Product> products = orderProcessor.getAllProducts();

        System.out.println("Available Products:");

        for (int i = 0; i < products.size(); i++) {

            System.out.println((i + 1) + ". " + products.get(i).getProductName());

        }

        System.out.print("Enter product number to add to order: ");

        int productChoice = scanner.nextInt();

        Product selectedProduct = products.get(productChoice - 1);

        List<Product> orderProducts = List.of(selectedProduct);

        orderProcessor.createOrder(user, orderProducts);

        System.out.println("Order placed successfully!");

    }

}

**exception/OrderNotFound:**

package exception;

public class OrderNotFoundException extends Exception {

    public OrderNotFoundException(String message) {

        super(message);

    }

}

**exception/UserNotFound:**

package exception;

public class UserNotFoundException extends Exception {

    public UserNotFoundException(String message) {

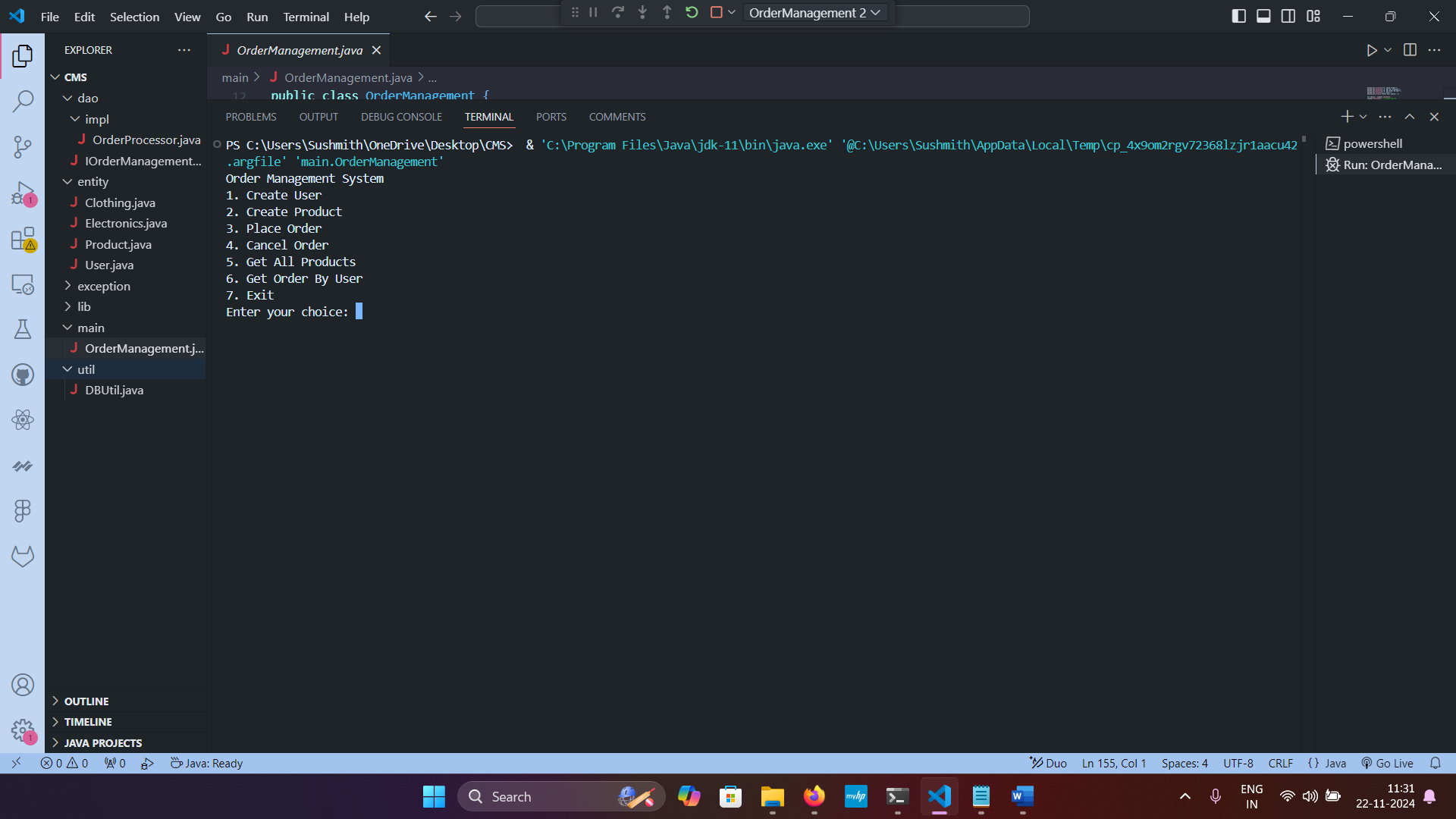
        super(message);

    }

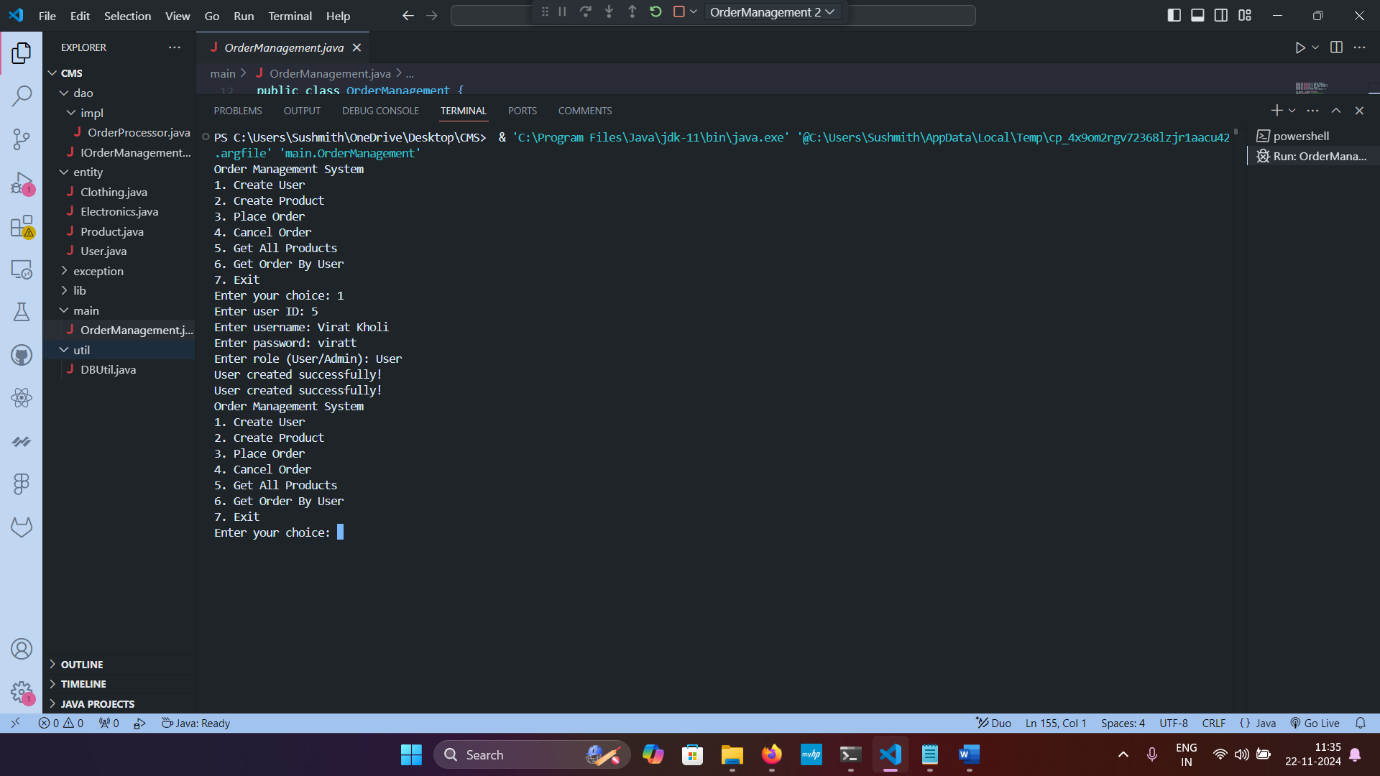
}

**Outputs:**

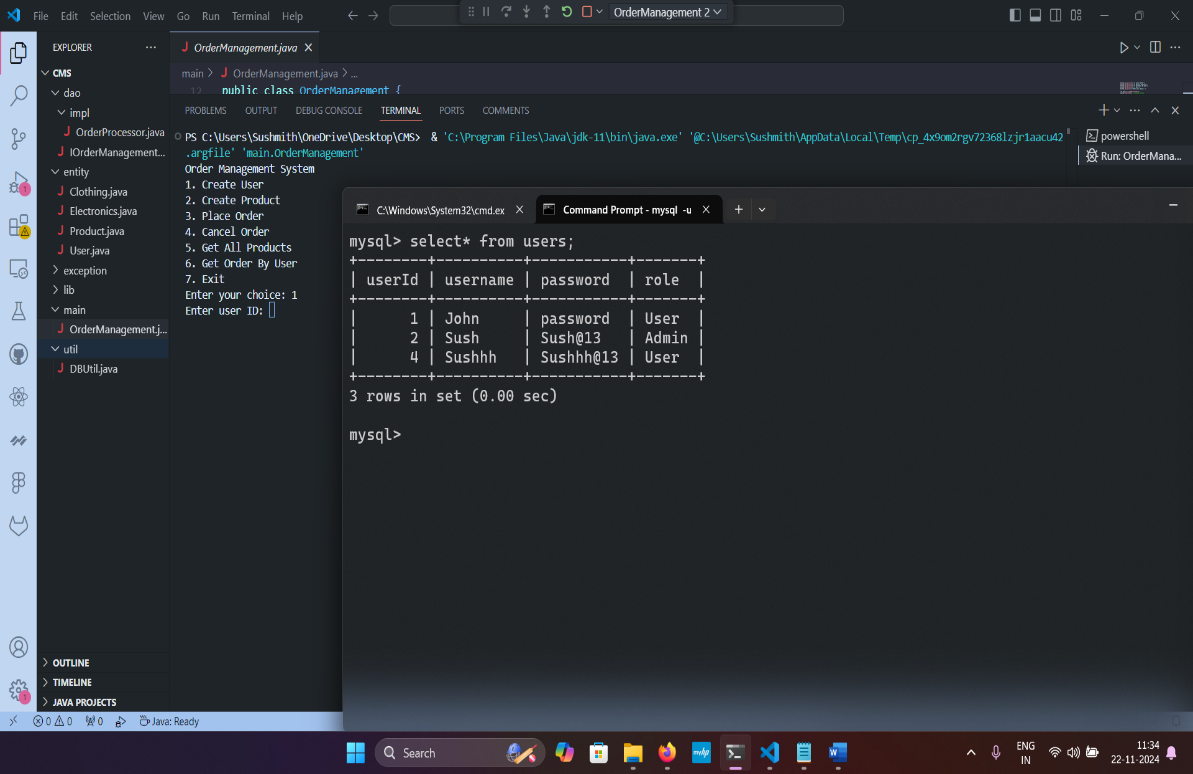
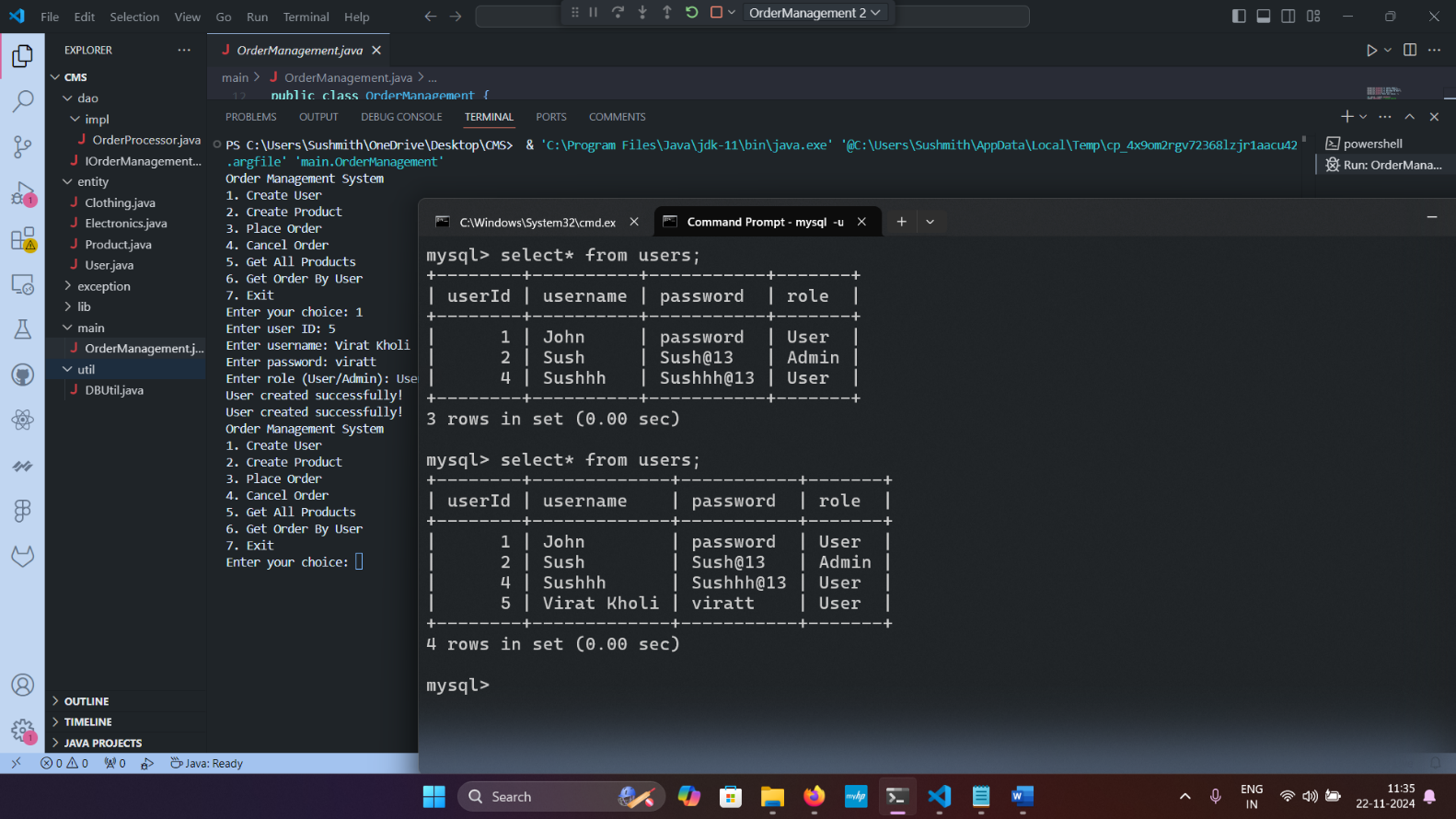
**Entering the choice:**



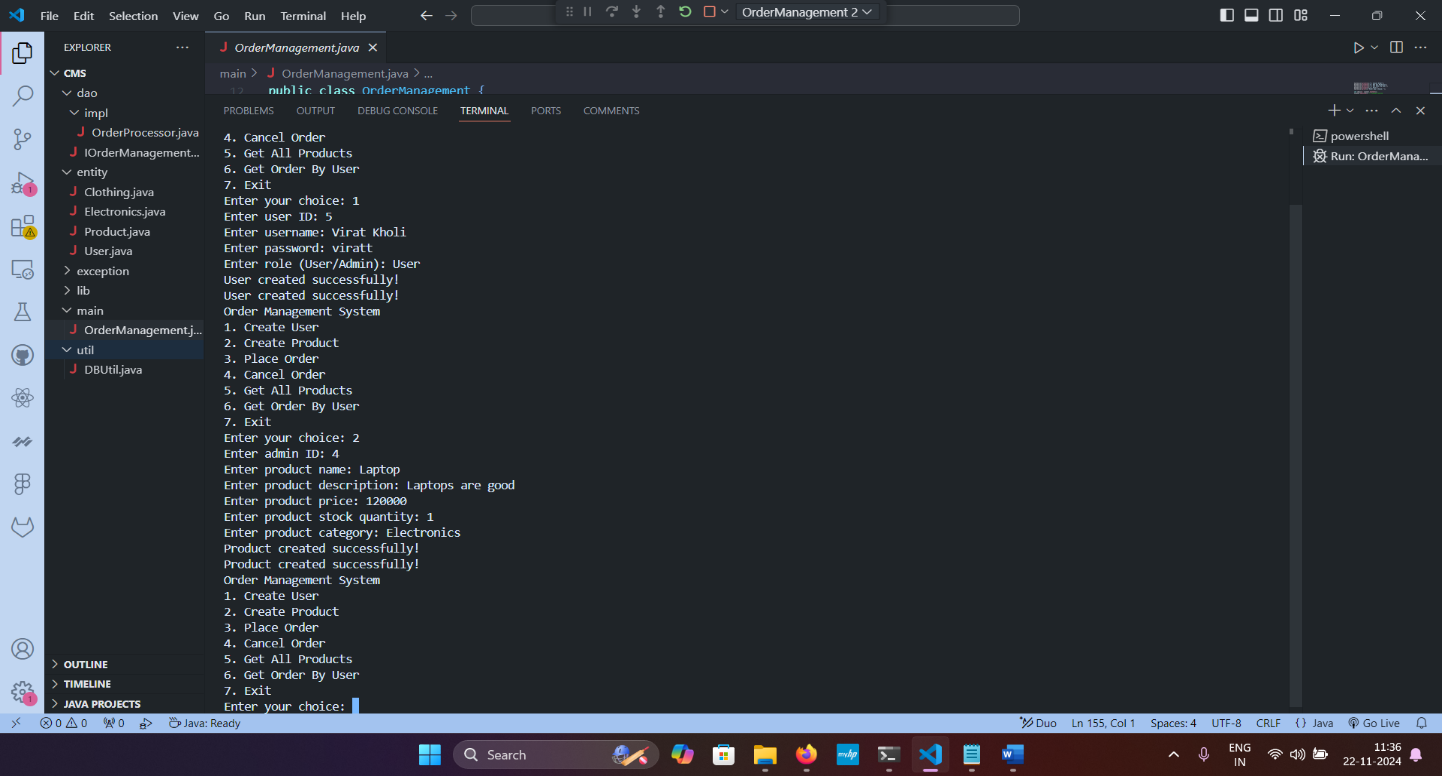
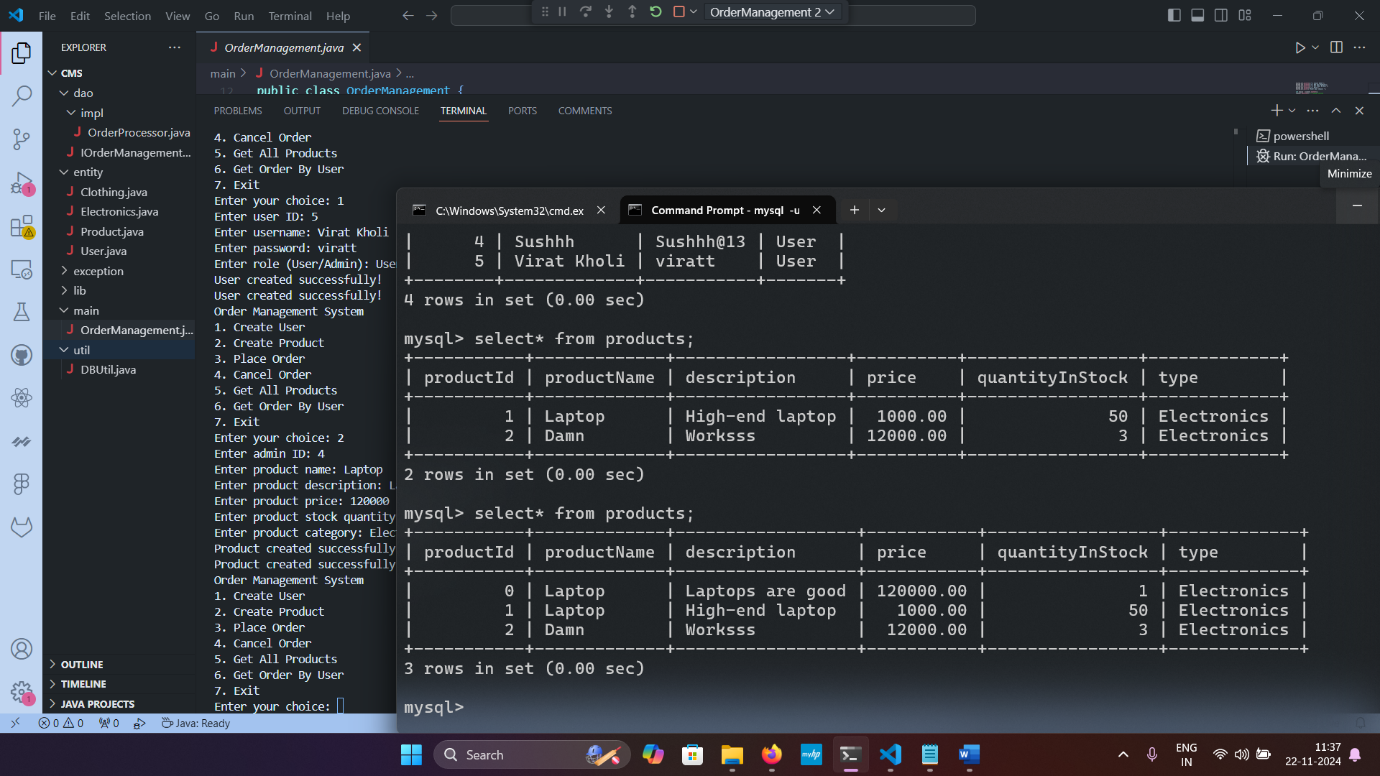
**Choice 1: Creating user. User/Admin.**

****

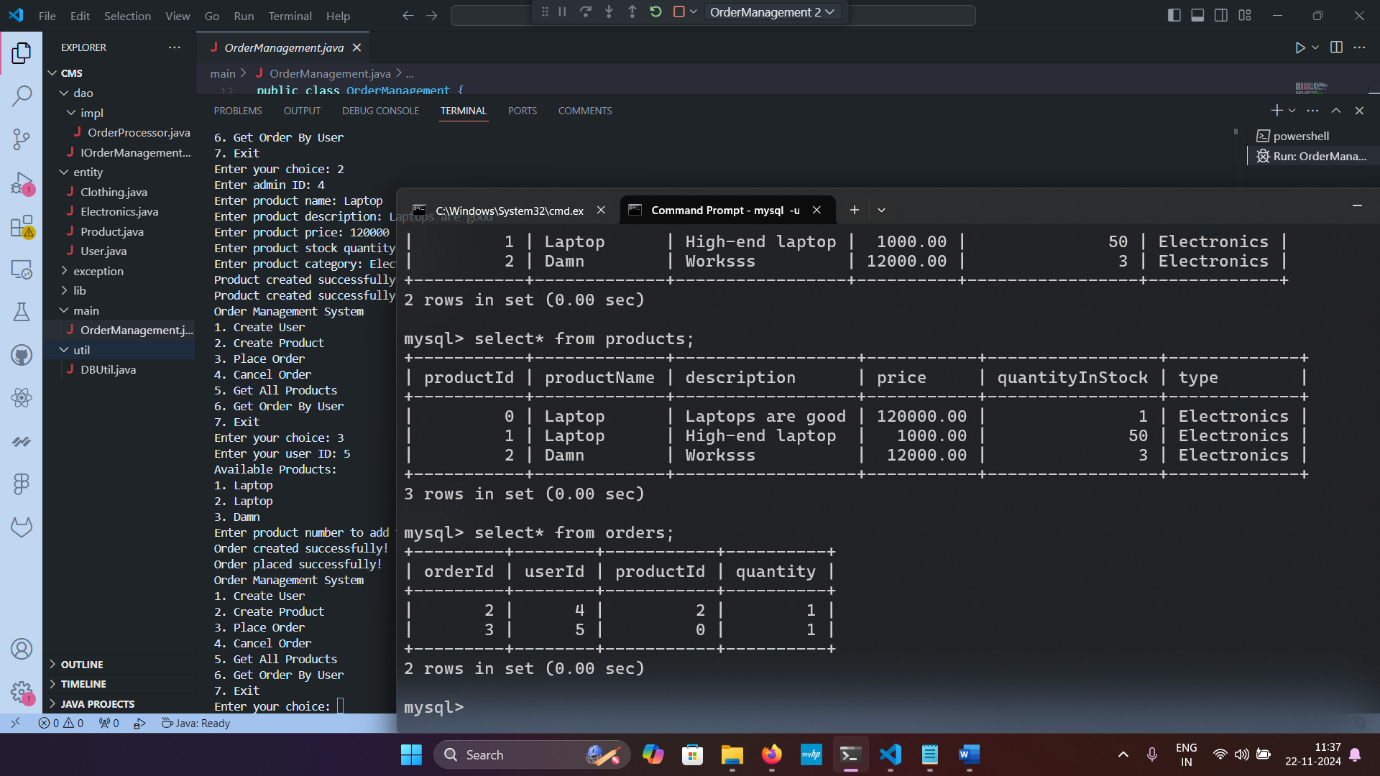
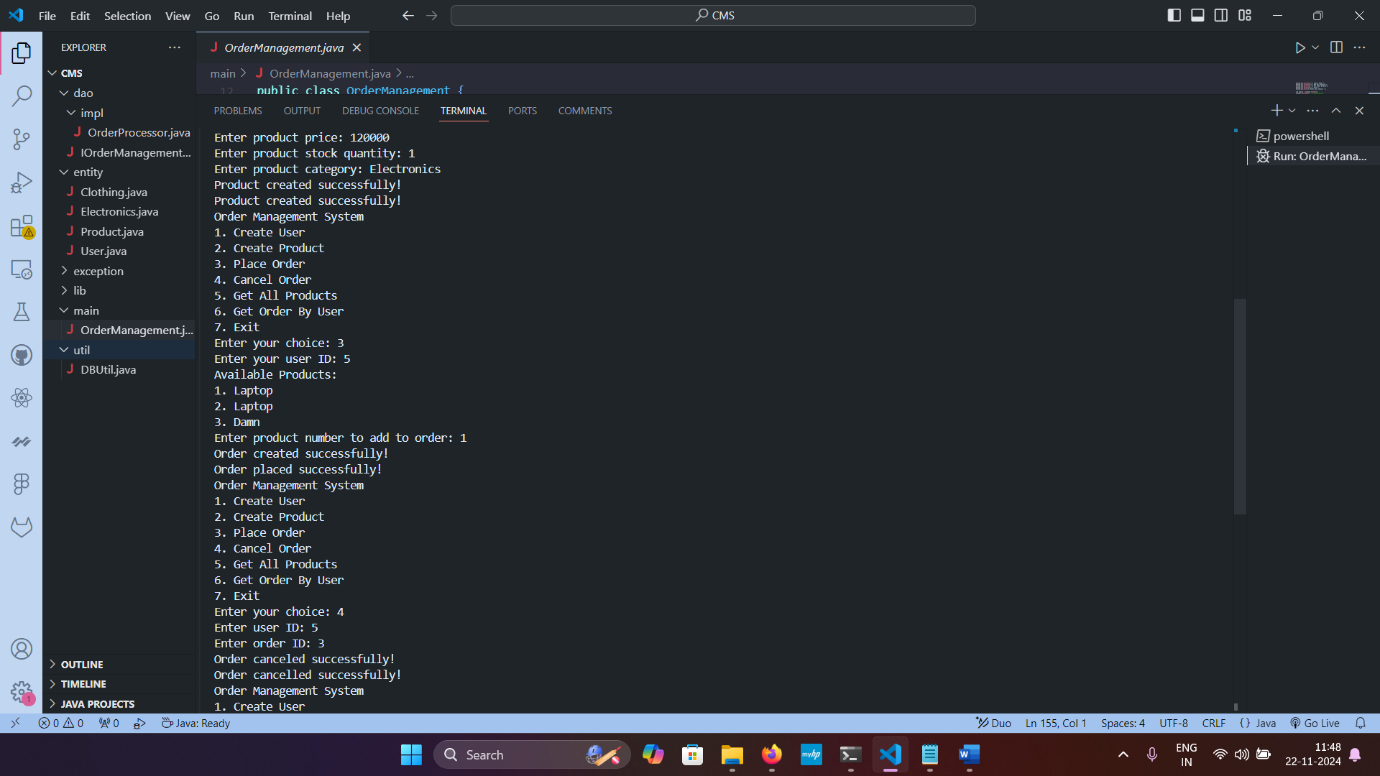
Before: After:



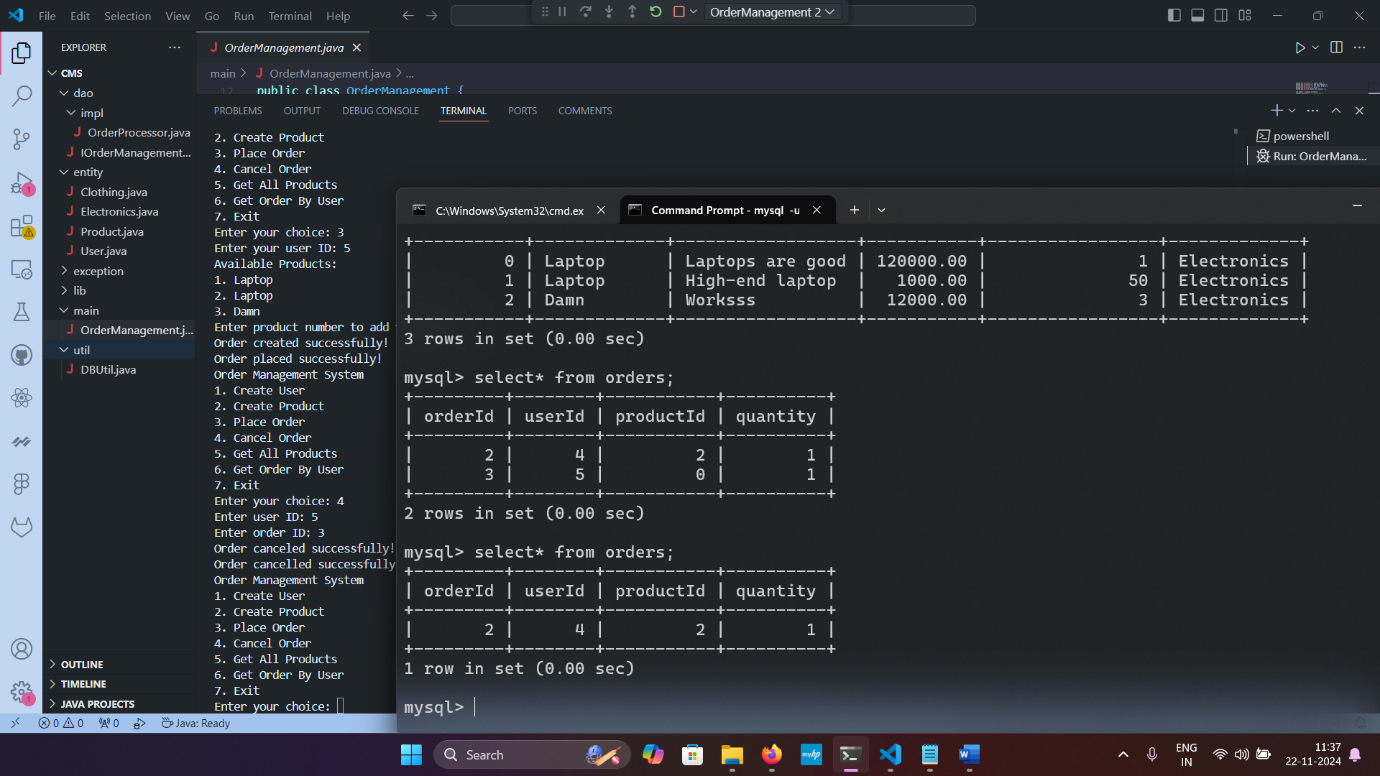
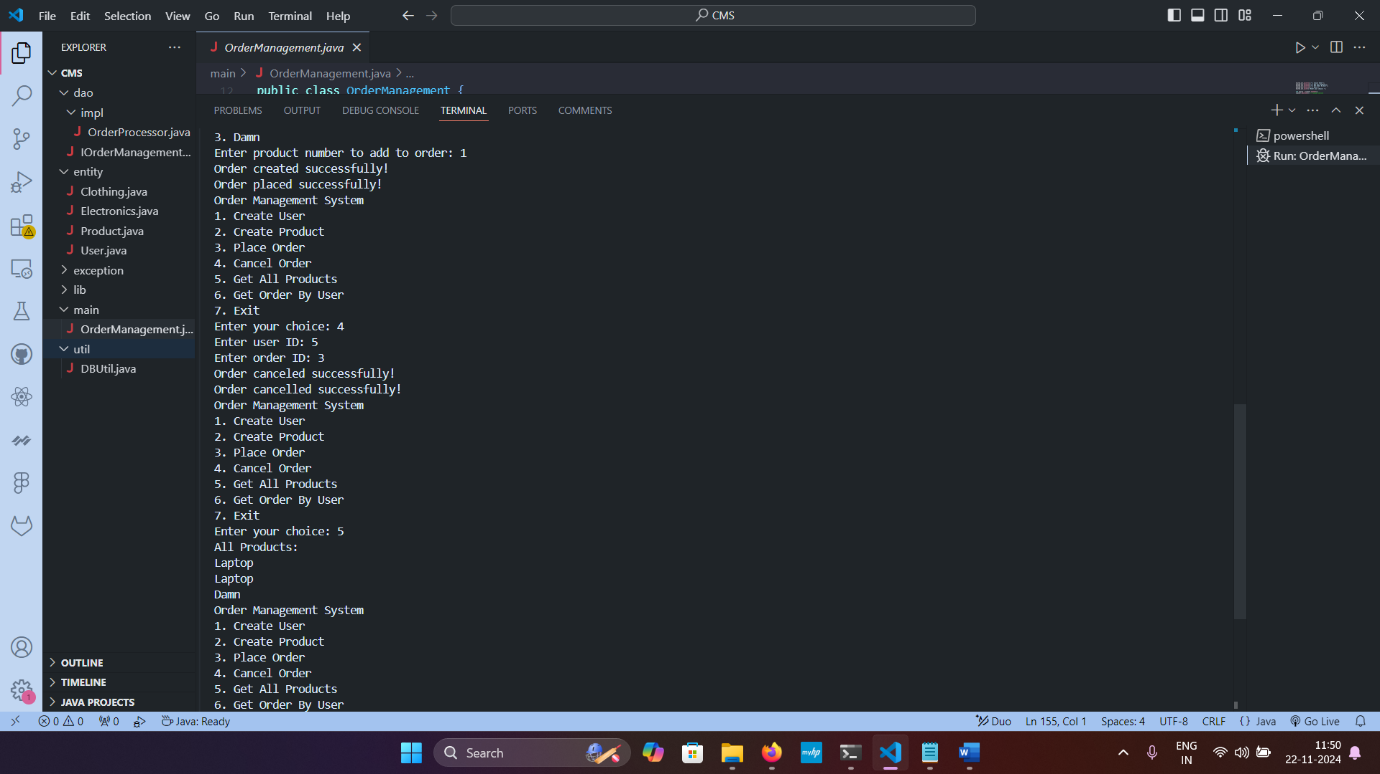
**Choice 2: Create Product.**

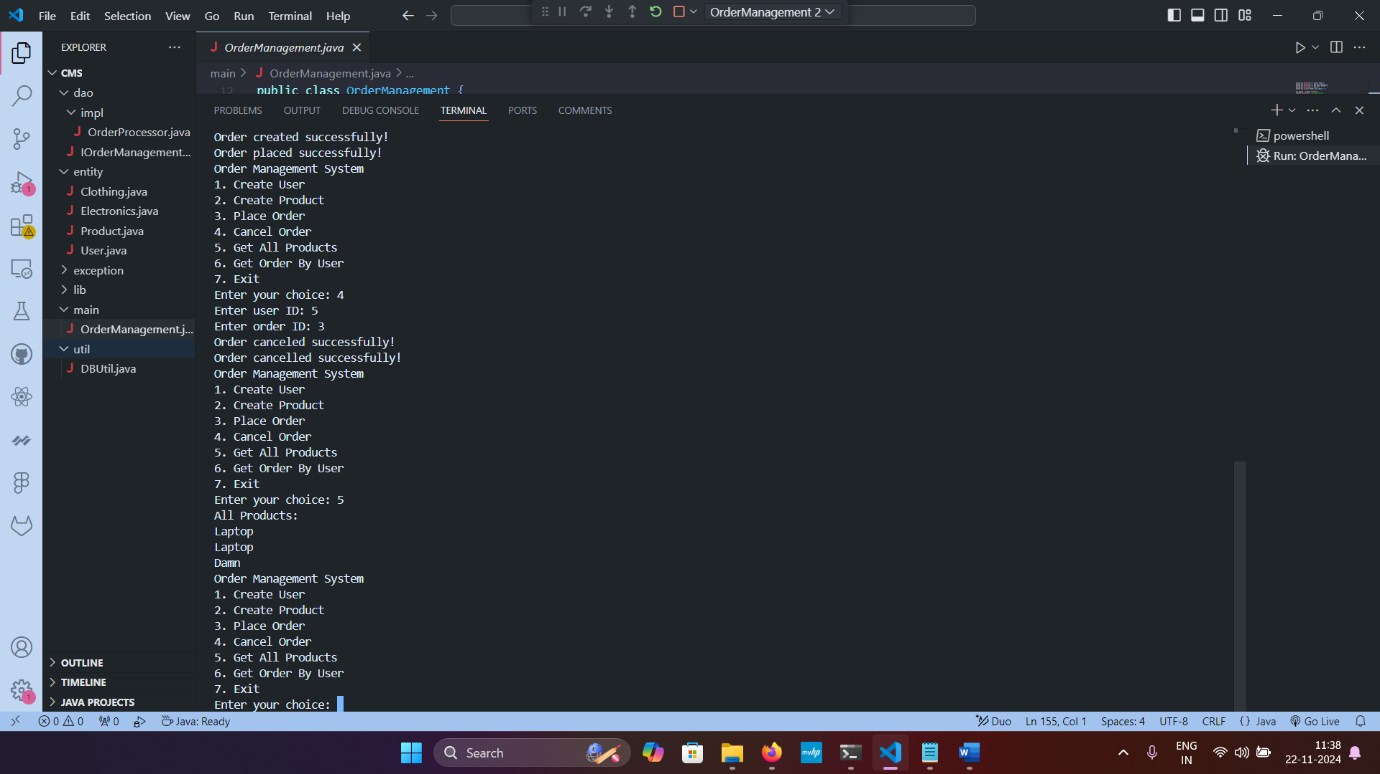
****

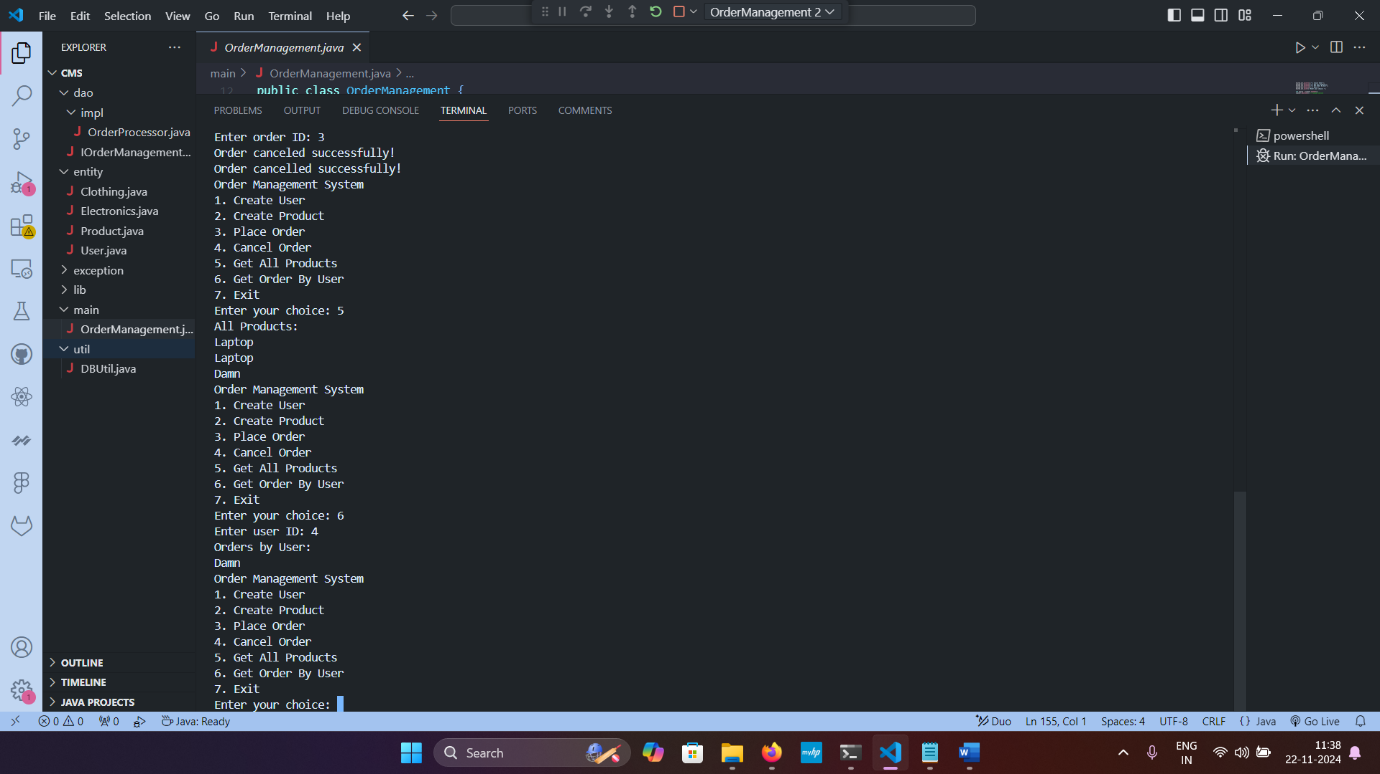
**Choice 3: Place Order.**

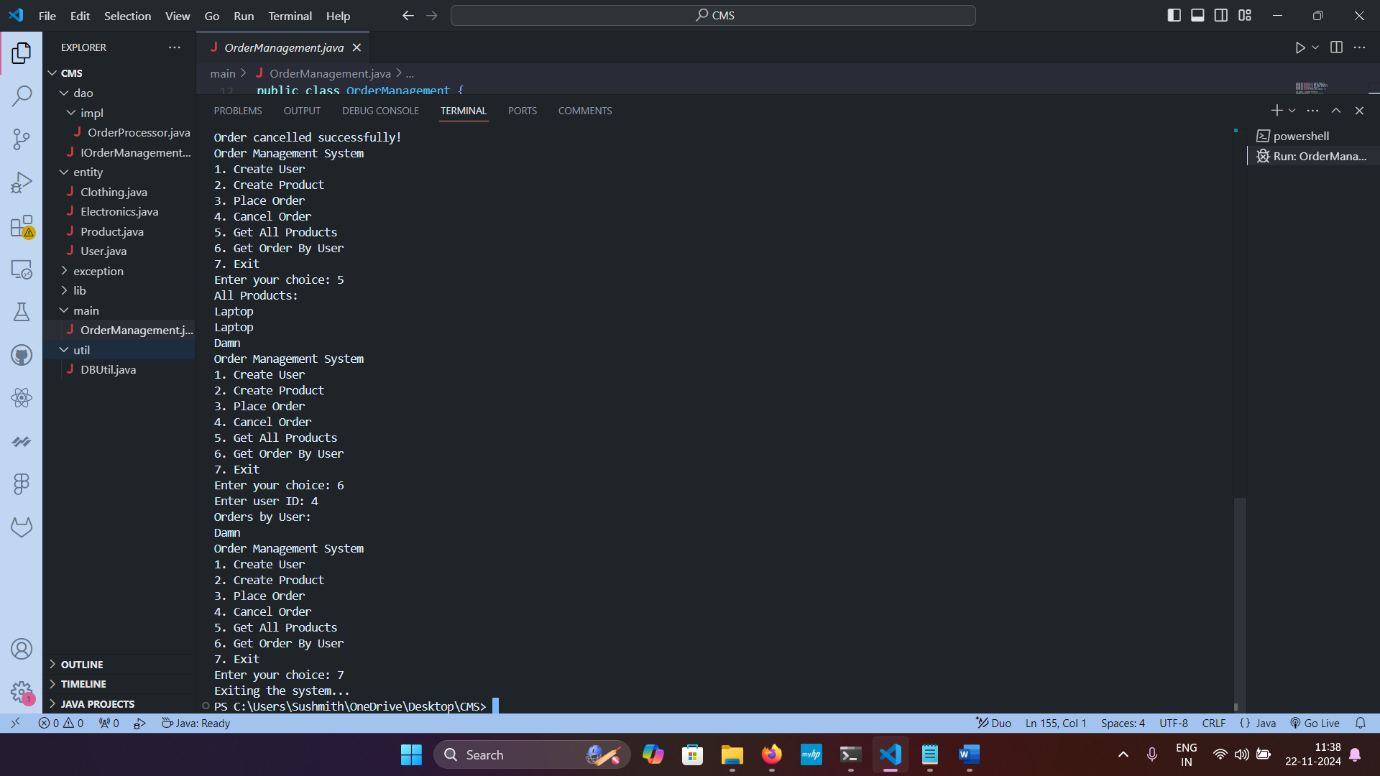
****

**Choice 4: Cancel Order.**

****

**Choice 5: Get all products.**

**Choice 6: Get order by user.**

**Choice 7: Exit.**