RAJALAKSHMI INSTITUTE OF TECHNOLOGY

(An Autonomous Institution, Affiliated to Anna University, Chennai)

DEPARTMENT OF CSE (ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING)

ACADEMIC YEAR 2025 - 2026

SEMESTER III

OBJECT ORIENTED PROGRAMMING LABORATORY

MINI PROJECT REPORT

REGISTER NUMBER	2117240030165
NAME	YASHWANTH.M.C.
PROJECT TITLE	ONLINE ORDER MANAGEMENT SYSTEM
DATE OF SUBMISSION	
FACULTY IN-CHARGE	Mrs STARLIN JENI

Introduction

In recent years, online shopping and delivery platforms have become widely adopted due to their convenience and accessibility. Managing product catalogs, customer records, and order processing forms the core functionality of such systems. The Online Order Management System is developed to automate and simplify this workflow using Java and MySQL. This application enables customers to browse products, add items to the cart, provide personal details, and place orders. The backend manages inventory, billing, and database updates. The system also demonstrates the use of Object-Oriented Programming (OOP) principles and GUI implementation using Java Swing. It ensures accurate recordkeeping through relational database integration and seamless interaction between GUI and backend logic. Thus, this project provides foundational exposure to building real-world e-commerce applications.

Project Description

The Online Order Management System is a GUI-based mini-project developed using Java Swing and MySQL. The application displays a list of products retrieved from the database. Users can add desired products to their cart by selecting quantity. Customer details are collected at checkout and stored along with order records. A dedicated Order module calculates total billing automatically. The project follows object-oriented modular design and uses JDBC for database interaction. Each unit—Product, Customer, Order, Cart—is implemented as a separate class. The system updates product stock after every order to maintain accurate inventory. All data is stored securely in the MySQL database. The system demonstrates basic e-commerce functionality and can be enhanced with features like online payment, user accounts, and admin control.

Objectives:

The main objectives of this project are:

1. To implement a functional order management system using OOP principles.
2. To maintain product catalogs and user orders.
3. To simulate real-time operations like adding items to cart and checking out.
4. To design a structured menu-driven interface for easy interaction.
5. To demonstrate modular programming using Java classes and packages.
6. To introduce exception handling for smoother user operations.
7. To store and retrieve products using Java collections.

System Requirements and Setup:

Hardware Requirements:

- Processor: Intel i3 or above
- RAM: 4GB or above
- Minimum storage: 1GB

Software Requirements:

- OS: Windows / Linux / macOS
- JDK version: 8 or above
- Text editor / IDE:
- VS Code / IntelliJ IDEA / Eclipse / Notepad++

Setup:

1. Install JDK (Java Development Kit).
2. Configure Java PATH in environment variables.
3. Open the project folder in an IDE or terminal.
4. Compile using:

   ```
   javac Main.java
   java Main
   ```

System Architecture (1 Diagram)

High-Level System Architecture

Code:

```
package com.onlineorder;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;
public class DBUtil {
   private static final String URL = "jdbc:mysql://localhost:3306/online_order_db?useSSL=false&serverTimezone=UTC";
   private static final String USER = "root";
   private static final String PASS = "your_mysql_password_here"; // <-- change this
   static {
      try {
         // optional for newer JDBC (Driver auto-registered), kept for clarity
         Class.forName("com.mysql.cj.jdbc.Driver");
      } catch (ClassNotFoundException e) {
         System.err.println("MySQL JDBC Driver not found. Add connector jar to classpath.");
         e.printStackTrace();
      }
   }
   public static Connection getConnection() throws SQLException {
      return DriverManager.getConnection(URL, USER, PASS);
   }
} package com.onlineorder;
public class Product {
   private int id;
   private String name;
   private double price;
   private int stock;

   public Product() {}
```

```
    public Product(int id, String name, double price, int stock) {
        this.id = id; this.name = name; this.price = price; this.stock = stock;
    }
    public int getId() { return id; }
    public String getName() { return name; }
    public double getPrice() { return price; }
    public int getStock() { return stock; }
    public void setStock(int stock) { this.stock = stock; }
    @Override
    public String toString() {
        return id + " - " + name + " (₹" + price + ") Stock: " + stock;
    }
}package com.onlineorder;
public class Customer {
    private int id;
    private String name;
    private String email;
    private String phone;
    public Customer() {}
    public Customer(int id, String name, String email, String phone) {
        this.id = id; this.name = name; this.email = email; this.phone = phone;
    }
    public Customer(String name, String email, String phone) {
        this.name = name; this.email = email; this.phone = phone;
    }
    public int getId() { return id; }
    public String getName() { return name; }
    public String getEmail() { return email; }
    public String getPhone() { return phone; }
    public void setId(int id) { this.id = id; }
} package com.onlineorder;
public class OrderItem {
    private Product product;
    private int quantity;
    private double price; // unit price at time of order
    public OrderItem(Product product, int quantity) {
        this.product = product;
        this.quantity = quantity;
        this.price = product.getPrice();
    }
    public Product getProduct() { return product; }
    public int getQuantity() { return quantity; }
    public double getPrice() { return price; }
    public double getTotal() { return price * quantity; }
} package com.onlineorder;
import java.util.ArrayList;
import java.util.Date;
import java.util.List;
public class Order {
    private int id;
    private Customer customer;
    private Date orderDate;
    private List<OrderItem> items = new ArrayList<>();
    public Order(Customer customer) {
        this.customer = customer;
        this.orderDate = new Date();
    }
    public void addItem(OrderItem item) { items.add(item); }
    public void removeItem(OrderItem item) { items.remove(item); }
    public List<OrderItem> getItems() { return items; }
    public double getTotal() {
        double t = 0;
        for (OrderItem i : items) t += i.getTotal();
        return t;
    }
    public Customer getCustomer() { return customer; }
    public Date getOrderDate() { return orderDate; }
    public void setId(int id) { this.id = id; }
    public int getId() { return id; }
} package com.onlineorder;
import java.sql.*;
import java.util.ArrayList;
import java.util.List;
public class ProductDAO {
    public List<Product> getAllProducts() {
        List<Product> list = new ArrayList<>();
        String sql = "SELECT id, name, price, stock FROM products";
        try (Connection con = DBUtil.getConnection();
             PreparedStatement ps = con.prepareStatement(sql);
             ResultSet rs = ps.executeQuery()) {
            while (rs.next()) {
                list.add(new Product(rs.getInt("id"),
                                     rs.getString("name"),
                                     rs.getDouble("price"),
                                     rs.getInt("stock")));
            }
        } catch (SQLException e) {
            e.printStackTrace();
        }
        return list;
    }
    public Product getProductById(int id) {
        String sql = "SELECT id, name, price, stock FROM products WHERE id = ?";
        try (Connection con = DBUtil.getConnection();
             PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setInt(1, id);
            try (ResultSet rs = ps.executeQuery()) {
                if (rs.next()) {
                    return new Product(rs.getInt("id"),
                                       rs.getString("name"),
                                       rs.getDouble("price"),
                                       rs.getInt("stock"));
                }
            }
        } catch (SQLException e) { e.printStackTrace(); }
        return null;
    }
    public boolean updateStock(int productId, int newStock) throws SQLException {
        String sql = "UPDATE products SET stock = ? WHERE id = ?";
        try (Connection con = DBUtil.getConnection();
             PreparedStatement ps = con.prepareStatement(sql)) {
            ps.setInt(1, newStock);
            ps.setInt(2, productId);
            return ps.executeUpdate() == 1;
        }
    }
} package com.onlineorder;
import java.sql.*;
public class CustomerDAO {
    public int insertCustomer(Customer c, Connection con) throws SQLException {
        String sql = "INSERT INTO customers (name, email, phone) VALUES (?, ?, ?)";
        try (PreparedStatement ps = con.prepareStatement(sql, Statement.RETURN_GENERATED_KEYS)) {
            ps.setString(1, c.getName());
            ps.setString(2, c.getEmail());
            ps.setString(3, c.getPhone());
            ps.executeUpdate();
            try (ResultSet keys = ps.getGeneratedKeys()) {
                if (keys.next()) {
                    int id = keys.getInt(1);
                    c.setId(id);
                    return id;
                }
            }
        }
        throw new SQLException("Creating customer failed, no ID obtained.");
    }
} package com.onlineorder;
import java.sql.*;
import java.util.List;
public class OrderDAO {
    public void placeOrder(Order order) throws SQLException {
        String insertOrder = "INSERT INTO orders (customer_id, total) VALUES (?, ?)";
        String insertItem = "INSERT INTO order_items (order_id, product_id, quantity, price) VALUES (?, ?, ?, ?)";
        ProductDAO productDAO = new ProductDAO();
```

```
CustomerDAO customerDAO = new CustomerDAO();
try (Connection con = DBUtil.getConnection()) {
  try {
    con.setAutoCommit(false);
    // 1) insert or get customer
    Customer c = order.getCustomer();
    int customerId = customerDAO.insertCustomer(c, con);
    // 2) insert order
    try (PreparedStatement psOrder = con.prepareStatement(insertOrder, Statement.RETURN_GENERATED_KEYS)) {
      psOrder.setInt(1, customerId);
      psOrder.setDouble(2, order.getTotal());
      psOrder.executeUpdate();
      try (ResultSet rs = psOrder.getGeneratedKeys()) {
        if (rs.next()) {
          int orderId = rs.getInt(1);
          order.setId(orderId);
          // 3) insert items & update product stock
          for (OrderItem item : order.getItems()) {
            try (PreparedStatement psItem = con.prepareStatement(insertItem)) {
              psItem.setInt(1, orderId);
              psItem.setInt(2, item.getProduct().getId());
              psItem.setInt(3, item.getQuantity());
              psItem.setDouble(4, item.getPrice());
              psItem.executeUpdate();
            }
            // update stock
            int newStock = item.getProduct().getStock() - item.getQuantity();
            if (newStock < 0) throw new SQLException("Not enough stock for product id " +
item.getProduct().getId());
            productDAO.updateStock(item.getProduct().getId(), newStock);
```

```
                    }
                } else {
                    throw new SQLException("Order insert failed, no ID obtained.");
                }
            }
        }
        con.commit();
    } catch (SQLException ex) {
        con.rollback();
        throw ex;
    } finally {
        con.setAutoCommit(true);
    }
  }
}
} package com.onlineorder;
import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.*;
import java.util.List;
public class MainFrame extends JFrame {
  private ProductDAO productDAO = new ProductDAO();
  private OrderDAO orderDAO = new OrderDAO();
  private JTable productTable;
  private DefaultTableModel productTableModel;
  private JTable cartTable;
  private DefaultTableModel cartTableModel;
```

```
// simple cart storage
private java.util.List<OrderItem> cart = new java.util.ArrayList<>();
// customer fields
private JTextField txtName = new JTextField(20);
private JTextField txtEmail = new JTextField(20);
private JTextField txtPhone = new JTextField(15);

public MainFrame() {
  setTitle("Online Order Management System - Mini Project");
  setSize(900, 600);
  setLocationRelativeTo(null);
  setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
  initComponents();
  loadProducts();
}
private void initComponents() {
  JPanel mainPanel = new JPanel(new BorderLayout(10,10));
  add(mainPanel);
  // Top label
  JLabel lblTitle = new JLabel("Online Order Management System");
  lblTitle.setFont(new Font("Arial", Font.BOLD, 20));
  lblTitle.setHorizontalAlignment(SwingConstants.CENTER);
  mainPanel.add(lblTitle, BorderLayout.NORTH);
  // Center split - left products, right cart & customer
  JSplitPane split = new JSplitPane(JSplitPane.HORIZONTAL_SPLIT);
  split.setResizeWeight(0.6);
  mainPanel.add(split, BorderLayout.CENTER);
  // Left: Products panel
  JPanel left = new JPanel(new BorderLayout(5,5));
```

```
productTableModel = new DefaultTableModel(new Object[]{"ID","Name","Price","Stock"}, 0) {
    public boolean isCellEditable(int r, int c) { return false; }
};
productTable = new JTable(productTableModel);
left.add(new JScrollPane(productTable), BorderLayout.CENTER);
JPanel addPanel = new JPanel();
addPanel.add(new JLabel("Quantity:"));
JSpinner spinnerQty = new JSpinner(new SpinnerNumberModel(1, 1, 100, 1));
addPanel.add(spinnerQty);
JButton btnAdd = new JButton("Add to Cart");
addPanel.add(btnAdd);
left.add(addPanel, BorderLayout.SOUTH);
split.setLeftComponent(left);
// Right: Cart & customer
JPanel right = new JPanel(new BorderLayout(5,5));
cartTableModel = new DefaultTableModel(new Object[]{"Product","Qty","Unit Price","Total"}, 0) {
    public boolean isCellEditable(int r, int c) { return false; }
};
cartTable = new JTable(cartTableModel);
right.add(new JScrollPane(cartTable), BorderLayout.CENTER);
// Cart controls
JPanel cartControls = new JPanel();
JButton btnRemove = new JButton("Remove Selected");
cartControls.add(btnRemove);
JButton btnClear = new JButton("Clear Cart");
cartControls.add(btnClear);
right.add(cartControls, BorderLayout.SOUTH);
// Bottom: Customer + PlaceOrder
JPanel bottomPanel = new JPanel(new GridLayout(2,1));
JPanel custPanel = new JPanel(new FlowLayout(FlowLayout.LEFT));
custPanel.setBorder(BorderFactory.createTitledBorder("Customer Details"));
custPanel.add(new JLabel("Name:")); custPanel.add(txtName);
custPanel.add(new JLabel("Email:")); custPanel.add(txtEmail);
custPanel.add(new JLabel("Phone:")); custPanel.add(txtPhone);
JPanel actionPanel = new JPanel();
JButton btnPlaceOrder = new JButton("Place Order");
actionPanel.add(btnPlaceOrder);
bottomPanel.add(custPanel);
bottomPanel.add(actionPanel);
JPanel rightWrapper = new JPanel(new BorderLayout());
rightWrapper.add(right, BorderLayout.CENTER);
rightWrapper.add(bottomPanel, BorderLayout.SOUTH);
split.setRightComponent(rightWrapper);
// Add listeners
btnAdd.addActionListener(e -> {
  int selRow = productTable.getSelectedRow();
  if (selRow == -1) {
    JOptionPane.showMessageDialog(this, "Please select a product first.");
    return;
  }
  int pid = (int) productTableModel.getValueAt(selRow, 0);
  String name = (String) productTableModel.getValueAt(selRow, 1);
  double price = (double) productTableModel.getValueAt(selRow, 2);
  int stock = (int) productTableModel.getValueAt(selRow, 3);
  int qty = (int) spinnerQty.getValue();
  if (qty <= 0) {
    JOptionPane.showMessageDialog(this, "Quantity must be positive.");
    return;
    }
    if (qty > stock) {
        JOptionPane.showMessageDialog(this, "Requested quantity exceeds stock.");
        return;
    }
    Product p = new Product(pid, name, price, stock);
    OrderItem oi = new OrderItem(p, qty);
    cart.add(oi);
    refreshCartTable();
});
btnRemove.addActionListener(e -> {
    int r = cartTable.getSelectedRow();
    if (r == -1) { JOptionPane.showMessageDialog(this, "Select an item in cart to remove."); return; }
    cart.remove(r);
    refreshCartTable();
});
btnClear.addActionListener(e -> {
    cart.clear();
    refreshCartTable();
});
btnPlaceOrder.addActionListener(e -> placeOrder());
// Right panel top toolbar: view orders button
JToolBar toolbar = new JToolBar();
JButton btnRefreshProducts = new JButton("Refresh Products");
toolbar.add(btnRefreshProducts);
JButton btnViewOrders = new JButton("View Orders (Console)");
toolbar.add(btnViewOrders);
mainPanel.add(toolbar, BorderLayout.SOUTH);
```

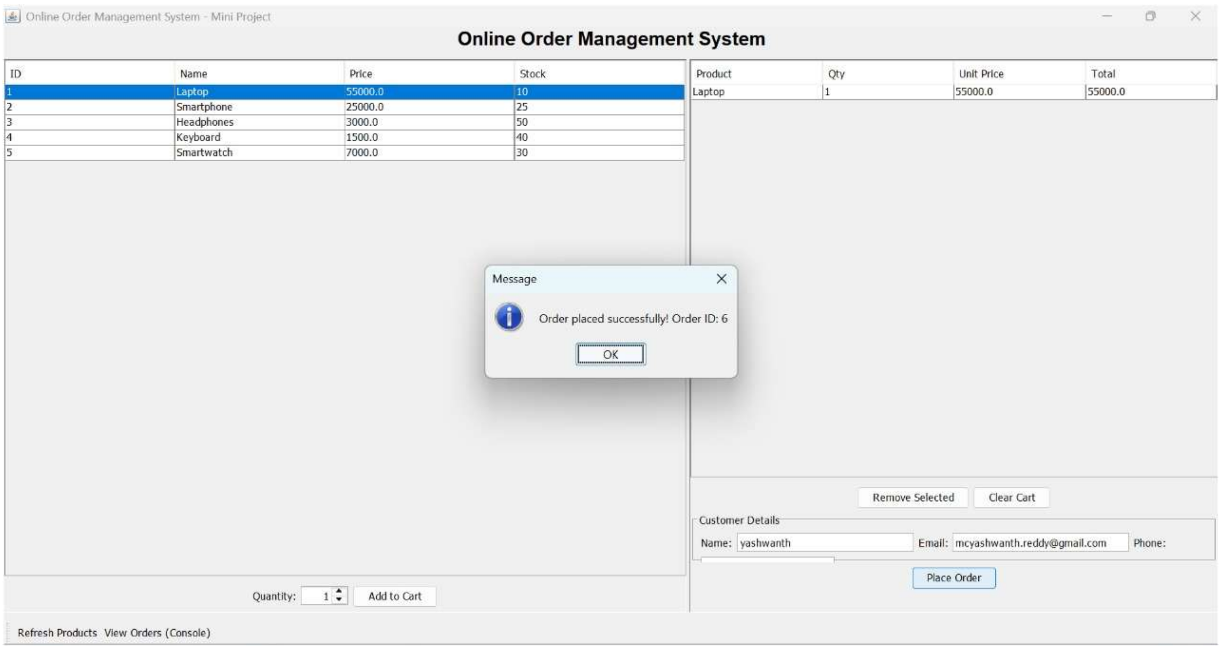
```
    btnRefreshProducts.addActionListener(e -> loadProducts());
    btnViewOrders.addActionListener(e -> viewOrders()); // simple console display
}
private void loadProducts() {
    productTableModel.setRowCount(0);
    List<Product> list = productDAO.getAllProducts();
    for (Product p : list) {
        productTableModel.addRow(new Object[]{p.getId(), p.getName(), p.getPrice(), p.getStock()});
    }
}
private void refreshCartTable() {
    cartTableModel.setRowCount(0);
    for (OrderItem oi : cart) {
        cartTableModel.addRow(new Object[]{
                oi.getProduct().getName(),
                oi.getQuantity(),
                oi.getPrice(),
                oi.getTotal()
        });
    }
}
private void placeOrder() {
    if (cart.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Cart is empty.");
        return;
    }
    String name = txtName.getText().trim();
    String email = txtEmail.getText().trim();
    String phone = txtPhone.getText().trim();
```

```
if (name.isEmpty()) {
  JOptionPane.showMessageDialog(this, "Enter customer name.");
  return;
}
Customer c = new Customer(name, email, phone);
Order order = new Order(c);
// Attach product snapshots (with current stock)
for (OrderItem oi : cart) {
  // fetch latest product from DB to ensure stock correctness
  Product latest = productDAO.getProductById(oi.getProduct().getId());
  if (latest == null) {
    JOptionPane.showMessageDialog(this, "Product no longer exists: " + oi.getProduct().getName());
    return;
  }
  if (latest.getStock() < oi.getQuantity()) {
    JOptionPane.showMessageDialog(this, "Not enough stock for " + latest.getName());
    return;
  }
  // create OrderItem with snapshot product info
  Product prodSnapshot = new Product(latest.getId(), latest.getName(), latest.getPrice(), latest.getStock());
  OrderItem newItem = new OrderItem(prodSnapshot, oi.getQuantity());
  order.addItem(newItem);
}
// confirm
int confirm = JOptionPane.showConfirmDialog(this,
    "Confirm placing order for total ₹" + order.getTotal() + " ?",
    "Confirm Order", JOptionPane.YES_NO_OPTION);
if (confirm != JOptionPane.YES_OPTION) return;
```

```
        // persist
        try {
            orderDAO.placeOrder(order);
            JOptionPane.showMessageDialog(this, "Order placed successfully! Order ID: " + order.getId());
            cart.clear();
            refreshCartTable();
            loadProducts();
            // optionally clear customer form
            //txtName.setText(""); txtEmail.setText(""); txtPhone.setText("");
        } catch (Exception ex) {
            ex.printStackTrace();
            JOptionPane.showMessageDialog(this, "Error placing order: " + ex.getMessage());
        }
    }
    private void viewOrders() {
        // Minimal: show orders and items from DB on console (extendable)
        try (java.sql.Connection con = DBUtil.getConnection();
             java.sql.Statement st = con.createStatement()) {
            java.sql.ResultSet rs = st.executeQuery("SELECT o.id, o.order_date, o.total, c.name FROM orders o JOIN customers c
ON o.customer_id = c.id ORDER BY o.order_date DESC");
            System.out.println("Orders:");
            while (rs.next()) {
                int oid = rs.getInt("id");
                System.out.println("Order ID: " + oid + ", Customer: " + rs.getString("name") +
                        ", Date: " + rs.getTimestamp("order_date") + ", Total: ₹" + rs.getDouble("total"));
                try (java.sql.PreparedStatement ps = con.prepareStatement("SELECT oi.quantity, oi.price, p.name FROM
order_items oi JOIN products p ON oi.product_id = p.id WHERE oi.order_id = ?")) {
                    ps.setInt(1, oid);
                    try (java.sql.ResultSet r2 = ps.executeQuery()) {
                        while (r2.next()) {
```

```
                        System.out.println("   " + r2.getString("name") + " x " + r2.getInt("quantity") + " @ " +
r2.getDouble("price"));
                    }
                }
            }
        }
    } catch (Exception e) {
        e.printStackTrace();
        JOptionPane.showMessageDialog(this, "Error loading orders: " + e.getMessage());
    }
}
public static void main(String[] args) {
    // set look and feel to system
    try { UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName()); } catch (Exception ignored) {}
    SwingUtilities.invokeLater(() -> {
        new MainFrame().setVisible(true);
    });
}
}
```

OUTPUT:



Online Order Management System - Mini Project

Online Order Management System

ID	Name	Price	Stock
1	Laptop	55000.0	10
2	Smartphone	25000.0	25
3	Headphones	3000.0	50
4	Keyboard	1500.0	40
5	Smartwatch	7000.0	30

Product	Qty	Unit Price	Total
Laptop	1	55000.0	55000.0

Message

Order placed successfully! Order ID: 6

OK

Remove Selected Clear Cart

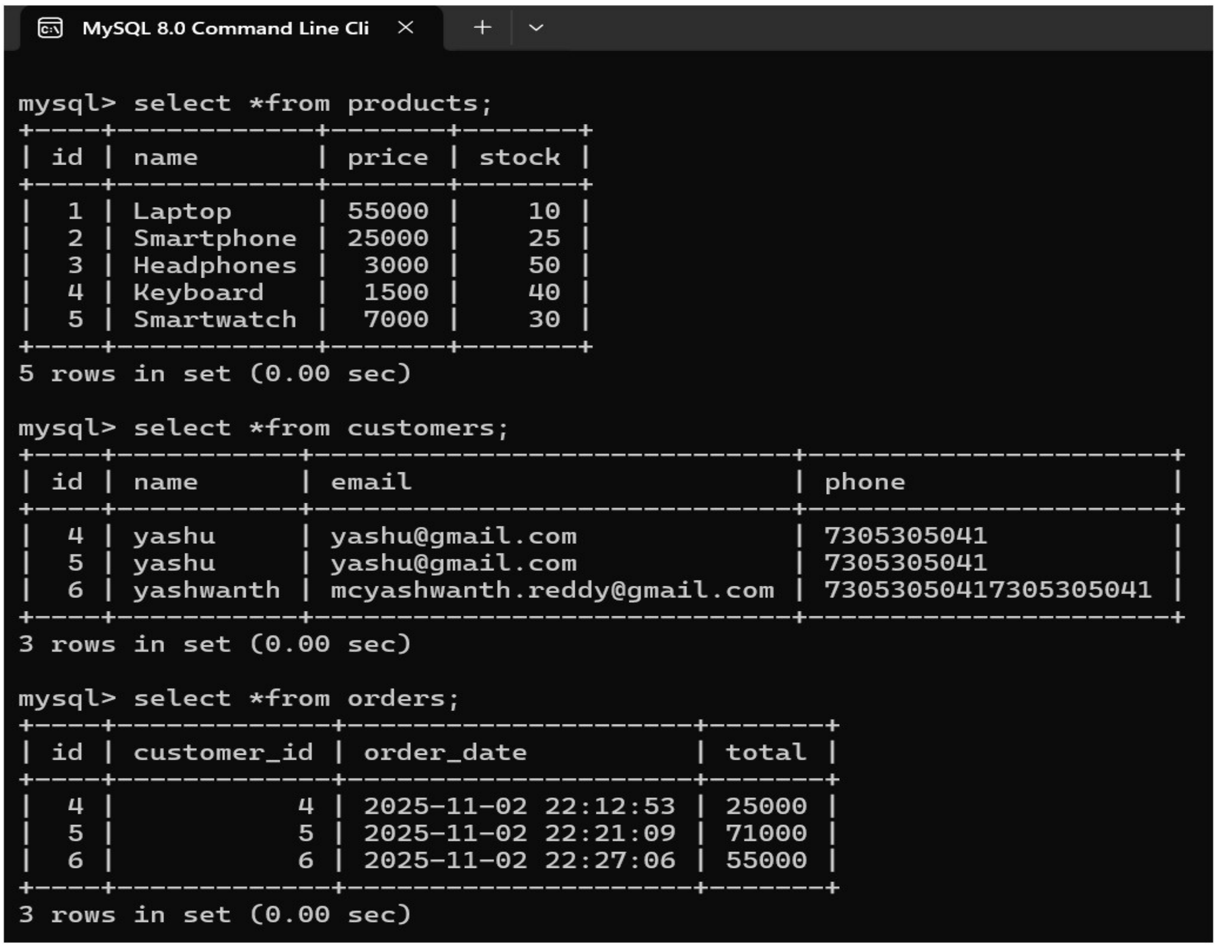
Customer Details

Name: yashwanth Email: mcyashwanth.reddy@gmail.com Phone:

Place Order

Quantity: 1 Add to Cart

Refresh Products View Orders (Console)

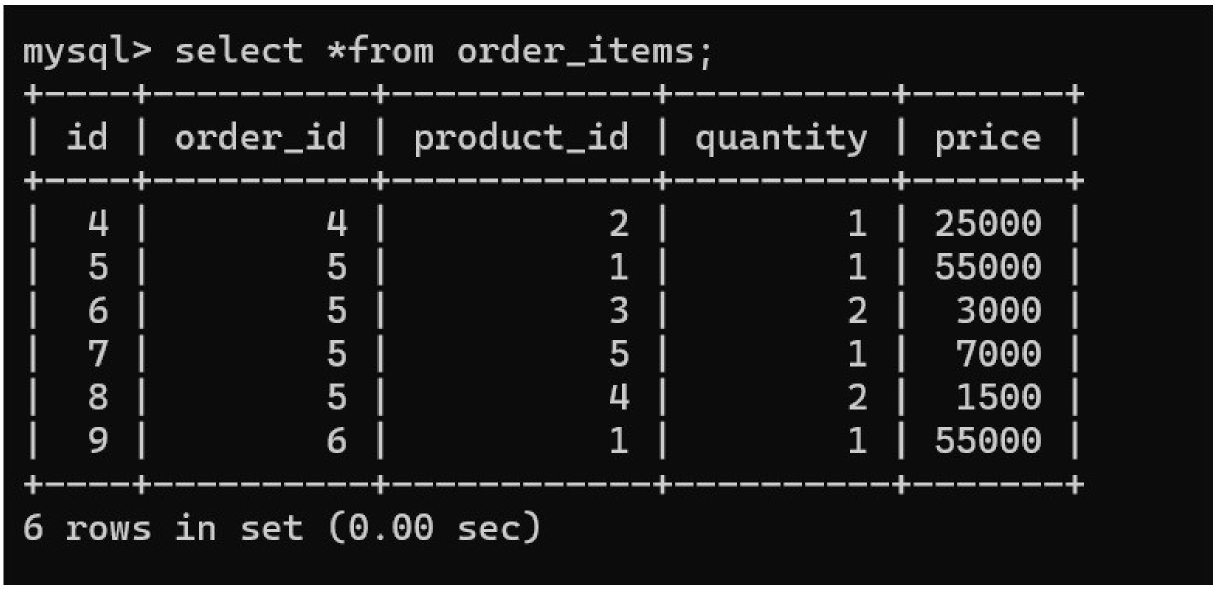


MySQL 8.0 Command Line Cli

```
mysql> select *from products;
+----+------------+-------+-------+
| id | name       | price | stock |
+----+------------+-------+-------+
|  1 | Laptop     | 55000 |    10 |
|  2 | Smartphone | 25000 |    25 |
|  3 | Headphones |  3000 |    50 |
|  4 | Keyboard   |  1500 |    40 |
|  5 | Smartwatch |  7000 |    30 |
+----+------------+-------+-------+
5 rows in set (0.00 sec)

mysql> select *from customers;
+----+-----------+------------------------------+-----------------------+
| id | name      | email                        | phone                 |
+----+-----------+------------------------------+-----------------------+
|  4 | yashu     | yashu@gmail.com              | 7305305041            |
|  5 | yashu     | yashu@gmail.com              | 7305305041            |
|  6 | yashwanth | mcyashwanth.reddy@gmail.com  | 73053050417305305041  |
+----+-----------+------------------------------+-----------------------+
3 rows in set (0.00 sec)

mysql> select *from orders;
+----+-------------+---------------------+-------+
| id | customer_id | order_date          | total |
+----+-------------+---------------------+-------+
|  4 |           4 | 2025-11-02 22:12:53 | 25000 |
|  5 |           5 | 2025-11-02 22:21:09 | 71000 |
|  6 |           6 | 2025-11-02 22:27:06 | 55000 |
+----+-------------+---------------------+-------+
3 rows in set (0.00 sec)
```



```
mysql> select *from order_items;
+----+----------+------------+----------+-------+
| id | order_id | product_id | quantity | price |
+----+----------+------------+----------+-------+
|  4 |        4 |          2 |        1 | 25000 |
|  5 |        5 |          1 |        1 | 55000 |
|  6 |        5 |          3 |        2 |  3000 |
|  7 |        5 |          5 |        1 |  7000 |
|  8 |        5 |          4 |        2 |  1500 |
|  9 |        6 |          1 |        1 | 55000 |
+----+----------+------------+----------+-------+
6 rows in set (0.00 sec)
```

Conclusion & Future Work

This Online Order Management System project was successfully developed using Java Swing and MySQL by applying key OOP concepts. The system enables product browsing, order placement, customer record storage, and inventory management. It integrates GUI and backend database effectively, ensuring proper data flow and usability. The project reflects how modern e-commerce platforms function at a fundamental level. The system can be extended with features like admin login, product search, payment gateway integration, automated invoice email, and role-based access. Future improvements could include analytics/reporting and mobile-friendly interfaces. Overall, the project demonstrates essential programming and application development skills relevant to real-world requirements.

PO-PSO Mapping (1–3 Rating Scale)

Attribute	Description	Rating
PO1	Engineering Knowledge	3
PO2	Problem Analysis	3
PO3	Design/Development	3
PO4	Investigation	2
PO5	Modern Tool Usage	3
PO10	Communication	2
PO11	Project Management	2
PO12	Life-long Learning	3
PSO1	AI-Based Decision Making	2
PSO2	Data Analytics & Visualization	2

GitHub Reference: **https://github.com/Yashwahthmc/oops-mini-project.git**