RAJALAKSHMI ENGINEERING COLLEGE

RAJALAKSHMI NAGAR, THANDALAM – 602 105

RAJALAKSHMI ENGINEERING COLLEGE

CS23333 OOPS Using Java

Laboratory Record Note Book

Name :

Year / Branch / Section :

University Register No. : ..

College Roll No. : .

Semester : .

Academic Year : .

RAJALAKSHMI ENGINEERING COLLEGE

An Autonomous Institution

BONAFIDE CERTIFICATE

Name……………………………………………………………………………….

Academic Year: …………… Semester: …………… Branch: ………………

Register No.

Certified that this is the bonafide record of work done by the above student in

the.. Laboratory

during the academic year 2025- 2026

Signature of Faculty in-charge

Submitted for the Practical Examination held on……………………………

Internal Examiner External Examiner

INDEX

EX.NO	DATE	NAME OF THE EXPERIMENT	GITHUB QR
1		I/O, Data Types, Operators	
2		Control Structures	
3		Arrays	
4		Strings	
5		Classes & Objects	
6		Inheritance	
7		Interface	
8		Exceptions	
9		Collections	
10		Collections	
11		Project	
12		Lambda	

ABSTRACT

The **Online Retail Management System** is a web-enabled application planned to automate retail activities such as managing products, inventory control, order processing, and payment processing. Designed using a **3-tier architecture**, it contains a front end developed using front end technology (**HTML, CSS, JavaScript**) and server -side programming apps (**PHP/Python/Java**), and an efficient MySQL database to store and retrieve data. The system is designed to maintain secure authentication, role-based access, and analytical reports for making informed decisions. Digitizing and financing the retail operation process, the system will enhance operational efficiency, reduce manual errors, and provide a centralized solution to retail management for the modern era**.**

ACKNOWLEDGEMENT

We express our sincere thanks to our beloved and honourable chairman MR. S. MEGANATHAN and the chairperson DR. M. THANGAM MEGANATHAN for their timely support and encouragement. We are greatly indebted to our respected and honourable principal Dr. S. N. MURUGESAN for his able support and guidance. Our heartfelt thanks to our Head of Department Dr. E. M. MALATHY and Deputy Head Dr. J. MANORANJINI for their constant encouragement throughout our project. We also extend our sincere gratitude to our internal guide Mr N. KARTHIKEYAN for her valuable guidance and motivation during the completion of this project. Finally, we thank our family members, friends, and all staff members of the Computer Science and Engineering department for their continuous support.

240701192-Himesh Niranjhan A

240701208-Janani G

240701200-Jaishuriya J

TABLE OF CONTENTS

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

1.1 INTRODUCTION

The Online Retail Management System is a web application that facilitates and automates several core retail functionalities, including product management, inventory management, order processing, and payment processing. To support data management, it integrates a MySQL database with HTML/CSS/JS/PHP/Python or Java. The system offers secure authentication, access control based on role, and updates in real time to improve accuracy and performance. It enhances operational efficiency and enables data-driven business decisions by digitizing retail functionality.

1.2 SCOPE OF THE WORK

The Online Retail Management System uses a single web-based environment to automate product management, order entry, inventory control, and billing. Special features allow secure transmissions and multi-user access as well as real-time data. It increases efficiency and decreases manual errors and can be scaled for businesses of different sizes.

1.3 PROBLEM STATEMENT

Retail management using manual methods leads to mistakes, lag time,and difficulties in managing inventory. Existing systems lack automation or timely updates and do not operate efficiently and effectively. Therefore, we need an online system that provides automation of retail functions, simplifies data accuracy and monitoring, and provides improved efficiency in overall business operations.

1.4 AIM AND OBJECTIVES OF THE PROJECT

The main objective of this project is to create an online-based Online Retail Management System aimed towards automating several retail operations such as product management, inventory tracking, billing, processing orders, and etc. The system objectives are a reduction of manual labor, improved data accuracy, and better customer convenience. The system offers secure authentication and role-based access. It also provides real-time updates of data for improved management. It also allows customers to shop online and produces analytical reports for better decisions, which ultimately correlates and better the performance of retail businesses.

CHAPTER 2

SYSTEM SPECIFICATIONS

2.1 HARDWARE SPECIFICATIONS

Processor	:	Intel i5
Memory Size	:	8GB (Minimum)
HDD	:	1 TB (Minimum)

2.2 SOFTWARE SPECIFICATIONS

Operating System	:	WINDOWS , macOS or Linux
Front – End	:	HTML
Back - End	:	Java,
API	:	JDBC

CHAPTER 3

MODULE DESCRIPTION

The application is structured into four primary modules: Data Model, Data Access Layer (DAO), Database Utility, and User Interfaces (UI).

1.DATA MODULE MODEL

Product.java: Acts as the blueprint for every item in the catalog. It encapsulates data fields (id, name, price, stock, category, image Path) and includes constructors, getters, and setters for data manipulation.

2. DATA ACCESS LAYER (DAO) MODULE

This module contains the core business logic for database interaction.

- **ProductDAO.java**: Manages all interactions with the products table.
    - ✦ **Methods Implemented:** add Product, getAllProducts, getProductById, update Product, delete Product.
    - ✦ **Critical Stock Management:** Includes the reduce Stock (int productid, int quantity) method, which is transactional and crucial for preventing overselling.

Filtering: Includes get Categories () and getProductsFiltered (String category, String search Term) to support dynamic catalog browsing.

3. DATABASE UTILITY MODULE

- **DBConnection.java**: A singleton class responsible for managing the SQLite connection.
    - ✦ **Core Function:** Provides a static method get Connection () to obtain a live database connection.

Initialization: Contains initialize Database (), which runs on application start to create the products and cart items tables and insert initial sample data

4. **USER INTERFACE (UI) MODULE**

The UI Module is built using **Java Swing** and comprises three main frames that manage the user experience and interaction.

1. **Welcome Frame**: The system entry point that initializes the database and directs users to either the Customer or Admin access points.
2. **Product Frame (Customer View)**: This is the main catalog, displaying products in a dynamic Table. It features a custom **Image Renderer** for visuals and includes **filtering** (by category) and **search** functionality. Crucially, it uses a custom Button Editor to handle "Add to Cart" actions while reflecting real-time stock status (e.g., showing "Out of Stock").
3. **Admin Frame (Management View)**: This secure panel is dedicated to inventory control. It allows administrators to select products from the table and perform **CRUD** (Create, Read, Update, and Delete) operations via dedicated input fields and action buttons, ensuring data synchronization with the MySQL database.

CHAPTER 4

SAMPLE CODING

1. Online retail

```
package com.ecommerce;

import javax.swing.*;

 import java.awt.*;

 import java.sql.*;

public class ECommerceApp {

    public static final String DB_URL = "jdbc:mysql://localhost:3306/shop_db";

    public static final String DB_USER = "root";

    public static final String DB_PASSWORD = "Akhil@13";

     public static int currentUserId = -1;

    public static String currentUsername = "";

    public static String currentUserRole = "";

     public static void main(String[] args) {

        SwingUtilities.invokeLater(() -> new LoginPage());

    }
    public static Connection getConnection() throws SQLException {

        try {

            Class.forName("com.mysql.cj.jdbc.Driver");

            return DriverManager.getConnection(DB_URL, DB_USER, DB_PASSWORD);

        } catch (ClassNotFoundException e) {

            throw new SQLException("MySQL JDBC Driver not found");

        }

    }

}
```

2. Admin Dashboard

```
package com.ecommerce;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;

public class AdminDashboard extends JFrame {
    private JTable productTable;
    private DefaultTableModel tableModel;
    private JTextField nameField, priceField, categoryField, stockField;
    private JTextArea descArea;

    public AdminDashboard() {
        setTitle("Admin Dashboard");
        setSize(1000, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

        // Top Panel
        JPanel topPanel = new JPanel(new BorderLayout());
        topPanel.setBackground(new Color(70, 130, 180));
        topPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15, 15));

        JLabel titleLabel = new JLabel("Admin Dashboard - Manage Products",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
        titleLabel.setForeground(Color.WHITE);

        JButton logoutBtn = new JButton("Logout");
        logoutBtn.setBackground(new Color(220, 20, 60));
        logoutBtn.setForeground(Color.WHITE);
        logoutBtn.addActionListener(e -> logout());

        topPanel.add(titleLabel, BorderLayout.CENTER);
        topPanel.add(logoutBtn, BorderLayout.EAST);
        mainPanel.add(topPanel, BorderLayout.NORTH);

        // Center - Table
        String[] columns = {"ID", "Name", "Description", "Price", "Category", "Stock"};
        tableModel = new DefaultTableModel(columns, 0);
        productTable = new JTable(tableModel);
        productTable.getSelectionModel().addListSelectionListener(e -> fillForm());

        JScrollPane scrollPane = new JScrollPane(productTable);
        mainPanel.add(scrollPane, BorderLayout.CENTER);

        // Right Panel - Form
        JPanel formPanel = new JPanel(new GridLayout(7, 2, 10, 10));
        formPanel.setBorder(BorderFactory.createTitledBorder("Product Details"));
        formPanel.setPreferredSize(new Dimension(300, 0));

        formPanel.add(new JLabel("Name:"));
        nameField = new JTextField();
        formPanel.add(nameField);
        formPanel.add(new JLabel("Description:"));
        descArea = new JTextArea(2, 20);
        formPanel.add(new JScrollPane(descArea));

        formPanel.add(new JLabel("Price:"));
        priceField = new JTextField();
        formPanel.add(priceField);

        formPanel.add(new JLabel("Category:"));
        categoryField = new JTextField();
        formPanel.add(categoryField);

        formPanel.add(new JLabel("Stock:"));
        stockField = new JTextField();
        formPanel.add(stockField);

        JButton addBtn = new JButton("Add");
        addBtn.setBackground(new Color(34, 139, 34));
        addBtn.setForeground(Color.WHITE);
        addBtn.addActionListener(e -> addProduct());
        formPanel.add(addBtn);

        JButton updateBtn = new JButton("Update");
        updateBtn.setBackground(new Color(30, 144, 255));
        updateBtn.setForeground(Color.WHITE);
        updateBtn.addActionListener(e -> updateProduct());
        formPanel.add(updateBtn);

        JButton deleteBtn = new JButton("Delete");
        deleteBtn.setBackground(new Color(220, 20, 60));
        deleteBtn.setForeground(Color.WHITE);
        deleteBtn.addActionListener(e -> deleteProduct());
        formPanel.add(deleteBtn);

        JButton clearBtn = new JButton("Clear");
        clearBtn.addActionListener(e -> clearForm());
        formPanel.add(clearBtn);

        mainPanel.add(formPanel, BorderLayout.EAST);

        add(mainPanel);
        loadProducts();
```

```
        setVisible(true);
    }

    private void loadProducts() {
        tableModel.setRowCount(0);
        try (Connection conn = ECommerceApp.getConnection()) {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM products");

            while (rs.next()) {
                tableModel.addRow(new Object[] {
                    rs.getInt("product_id"),
                    rs.getString("product_name"),
                    rs.getString("description"),
                    rs.getDouble("price"),
                    rs.getString("category"),
                    rs.getInt("stock_quantity")
                });
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void fillForm() {
        int row = productTable.getSelectedRow();
        if (row >= 0) {
            nameField.setText(tableModel.getValueAt(row, 1).toString());
            descArea.setText(tableModel.getValueAt(row, 2).toString());
            priceField.setText(tableModel.getValueAt(row, 3).toString());
            categoryField.setText(tableModel.getValueAt(row, 4).toString());
            stockField.setText(tableModel.getValueAt(row, 5).toString());
        }
    }

    private void addProduct() {
        try (Connection conn = ECommerceApp.getConnection()) {
            String query = "INSERT INTO products (product_name, description, price,
category, stock_quantity) VALUES (?, ?, ?, ?, ?)";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, nameField.getText());
            pst.setString(2, descArea.getText());
            pst.setDouble(3, Double.parseDouble(priceField.getText()));
            pst.setString(4, categoryField.getText());
            pst.setInt(5, Integer.parseInt(stockField.getText()));
            pst.executeUpdate();

            JOptionPane.showMessageDialog(this, "Product added!");
            clearForm();
            loadProducts();
        } catch (SQLException | NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void updateProduct() {
        int row = productTable.getSelectedRow();
        if (row < 0) {
            JOptionPane.showMessageDialog(this, "Select a product!");
            return;
        }

        int id = (int) tableModel.getValueAt(row, 0);

        try (Connection conn = ECommerceApp.getConnection()) {
            String query = "UPDATE products SET product_name=?, description=?,
price=?, category=?, stock_quantity=? WHERE product_id=?";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, nameField.getText());
            pst.setString(2, descArea.getText());
            pst.setDouble(3, Double.parseDouble(priceField.getText()));
            pst.setString(4, categoryField.getText());
            pst.setInt(5, Integer.parseInt(stockField.getText()));
            pst.setInt(6, id);
            pst.executeUpdate();

            JOptionPane.showMessageDialog(this, "Product updated!");
            clearForm();
            loadProducts();
        } catch (SQLException | NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void deleteProduct() {
        int row = productTable.getSelectedRow();
        if (row < 0) {
            JOptionPane.showMessageDialog(this, "Select a product!");
            return;
        }

        int id = (int) tableModel.getValueAt(row, 0);

        int confirm = JOptionPane.showConfirmDialog(this, "Delete this product?");
        if (confirm == JOptionPane.YES_OPTION) {
            try (Connection conn = ECommerceApp.getConnection()) {
                String query = "DELETE FROM products WHERE product_id=?";
                PreparedStatement pst = conn.prepareStatement(query);
                pst.setInt(1, id);
                pst.executeUpdate();

                JOptionPane.showMessageDialog(this, "Product deleted!");
                clearForm();
                loadProducts();
            } catch (SQLException ex) {
                JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
            }
        }
    }

    private void clearForm() {
        nameField.setText("");
        descArea.setText("");
        priceField.setText("");
        categoryField.setText("");
        stockField.setText("");
        productTable.clearSelection();
    }

    private void logout() {
        dispose();
        new LoginPage();
    }
}
```

3. Cart Page

```
package com.ecommerce;
```

```
import javax.swing.*;

import javax.swing.table.DefaultTableModel;

import java.awt.*;

import java.util.ArrayList;



public class CartPage extends JFrame {

    private JTable cartTable;

    private DefaultTableModel tableModel;

    private ArrayList<CartItem> cartItems;

    private UserDashboard parent;

    private JLabel totalLabel;



    public CartPage(ArrayList<CartItem> cartItems, UserDashboard parent) {

        this.cartItems = cartItems;

        this.parent = parent;



        setTitle("Shopping Cart");
```

```
setSize(700, 500);

setLocationRelativeTo(parent);

setDefaultCloseOperation(DISPOSE_ON_CLOSE);



JPanel mainPanel = new JPanel(new BorderLayout(10, 10));

mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

mainPanel.setBackground(Color.WHITE);



// Title

JLabel titleLabel = new JLabel("🛒 Your Shopping Cart", SwingConstants.CENTER);

titleLabel.setFont(new Font("Arial", Font.BOLD, 24));

titleLabel.setForeground(new Color(70, 130, 180));

titleLabel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));

mainPanel.add(titleLabel, BorderLayout.NORTH);



// Cart Table

String[] columns = {"Product Name", "Price (₹)", "Quantity", "Total (₹)"};
```

```
tableModel = new DefaultTableModel(columns, 0) {

    @Override

    public boolean isCellEditable(int row, int column) {

        return false;

    }

};


cartTable = new JTable(tableModel);

cartTable.setFont(new Font("Arial", Font.PLAIN, 14));

cartTable.setRowHeight(35);

cartTable.getTableHeader().setFont(new Font("Arial", Font.BOLD, 14));

cartTable.getTableHeader().setBackground(new Color(70, 130, 180));

cartTable.getTableHeader().setForeground(Color.BLACK);


loadCartItems();


JScrollPane scrollPane = new JScrollPane(cartTable);

scrollPane.setBorder(BorderFactory.createTitledBorder("Cart Items"));
```

```
mainPanel.add(scrollPane, BorderLayout.CENTER);


// Bottom Panel

JPanel bottomPanel = new JPanel(new BorderLayout(10, 10));

bottomPanel.setBackground(Color.WHITE);


// Total Panel

JPanel totalPanel = new JPanel(new FlowLayout(FlowLayout.RIGHT));

totalPanel.setBackground(new Color(240, 248, 255));

totalPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10, 10));


totalLabel = new JLabel("Grand Total: ₹" + calculateTotal());

totalLabel.setFont(new Font("Arial", Font.BOLD, 20));

totalLabel.setForeground(new Color(0, 100, 0));

totalPanel.add(totalLabel);


bottomPanel.add(totalPanel, BorderLayout.NORTH);
```

```
// Button Panel

JPanel buttonPanel = new JPanel(new FlowLayout(FlowLayout.CENTER, 15,
10));

buttonPanel.setBackground(Color.WHITE);



JButton removeBtn = new JButton("Remove Selected");

removeBtn.setFont(new Font("Arial", Font.BOLD, 13));

removeBtn.setBackground(new Color(255, 182, 193));

removeBtn.setForeground(Color.BLACK);

removeBtn.setPreferredSize(new Dimension(150, 40));

removeBtn.addActionListener(e -> removeItem());

buttonPanel.add(removeBtn);



JButton clearBtn = new JButton("Clear Cart");

clearBtn.setFont(new Font("Arial", Font.BOLD, 13));

clearBtn.setBackground(new Color(255, 215, 0));

clearBtn.setForeground(Color.BLACK);

clearBtn.setPreferredSize(new Dimension(150, 40));
clearBtn.addActionListener(e -> clearCart());
buttonPanel.add(clearBtn);

JButton checkoutBtn = new JButton("💳 Checkout");
checkoutBtn.setFont(new Font("Arial", Font.BOLD, 14));
checkoutBtn.setBackground(new Color(144, 238, 144));
checkoutBtn.setForeground(Color.BLACK);
checkoutBtn.setPreferredSize(new Dimension(150, 40));
checkoutBtn.addActionListener(e -> checkout());
buttonPanel.add(checkoutBtn);

JButton backBtn = new JButton("Continue Shopping");
backBtn.setFont(new Font("Arial", Font.BOLD, 13));
backBtn.setBackground(new Color(211, 211, 211));
backBtn.setForeground(Color.BLACK);
backBtn.setPreferredSize(new Dimension(180, 40));
backBtn.addActionListener(e -> dispose());
buttonPanel.add(backBtn);
```

```
    bottomPanel.add(buttonPanel, BorderLayout.CENTER);

    mainPanel.add(bottomPanel, BorderLayout.SOUTH);

    add(mainPanel);
    setVisible(true);
}

private void loadCartItems() {
    tableModel.setRowCount(0);
    for (CartItem item : cartItems) {
        tableModel.addRow(new Object[] {
            item.productName,
            String.format("%.2f", item.price),
            item.quantity,
            String.format("%.2f", item.getTotal())
        });
    }
}


private String calculateTotal() {
    double total = 0;
    for (CartItem item : cartItems) {
        total += item.getTotal();
    }
    return String.format("%.2f", total);
}


private void removeItem() {
    int selectedRow = cartTable.getSelectedRow();
    if (selectedRow < 0) {
        JOptionPane.showMessageDialog(this,
            "Please select an item to remove!",
            "No Selection",
            JOptionPane.WARNING_MESSAGE);
        return;
    }


    String productName = tableModel.getValueAt(selectedRow, 0).toString();
    int confirm = JOptionPane.showConfirmDialog(this,
        "Remove " + productName + " from cart?",
        "Confirm Remove",
        JOptionPane.YES_NO_OPTION);


    if (confirm == JOptionPane.YES_OPTION) {
        cartItems.remove(selectedRow);
        loadCartItems();
        totalLabel.setText("Grand Total: ₹" + calculateTotal());


        if (cartItems.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Cart is now empty!");
            dispose();
        }
    }
}

private void clearCart() {
    if (cartItems.isEmpty()) {
        JOptionPane.showMessageDialog(this, "Cart is already empty!");
        return;
    }

    int confirm = JOptionPane.showConfirmDialog(this,
        "Are you sure you want to clear the entire cart?",
        "Clear Cart",
        JOptionPane.YES_NO_OPTION);

    if (confirm == JOptionPane.YES_OPTION) {
        cartItems.clear();
        JOptionPane.showMessageDialog(this, "Cart cleared successfully!");
        dispose();
    }

}


private void checkout() {

    if (cartItems.isEmpty()) {

        JOptionPane.showMessageDialog(this, "Cart is empty!");

        return;

    }



    double total = 0;

    StringBuilder orderSummary = new StringBuilder("Order Summary:\n\n");



    for (CartItem item : cartItems) {

        total += item.getTotal();

        orderSummary.append(String.format("%s x %d = ₹%.2f\n",

            item.productName, item.quantity, item.getTotal()));

    }
```

```
orderSummary.append(String.format("\n------------------\n"));

orderSummary.append(String.format("Grand Total: ₹%.2f\n\n", total));

orderSummary.append("Proceed with payment?");



int confirm = JOptionPane.showConfirmDialog(this,

    orderSummary.toString(),

    "Confirm Order",

    JOptionPane.YES_NO_OPTION);



if (confirm == JOptionPane.YES_OPTION) {

    // Show success message

    JOptionPane.showMessageDialog(this,

        String.format("✅ Order Placed Successfully!\n\n" +

                "Order Details:\n" +

                "Total Items: %d\n" +

                "Total Amount: ₹%.2f\n\n" +

                "Thank you for shopping with us!\n" +

                "Your order will be delivered soon.",
                cartItems.size(), total),

            "Order Successful",

            JOptionPane.INFORMATION_MESSAGE);



        // Clear cart after successful checkout

        cartItems.clear();

        dispose();

      }

    }

  }
```

4. Login Page

```
package com.ecommerce;

import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class LoginPage extends JFrame {
    private JTextField usernameField;
    private JPasswordField passwordField;
    private JComboBox<String> roleComboBox;

    public LoginPage() {
        setTitle("E-Commerce Login");
        setSize(450, 500);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new GridLayout(8, 2, 10, 15));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(30, 30, 30,
30));
        mainPanel.setBackground(new Color(240, 248, 255));

        JLabel titleLabel = new JLabel("🛒 E-Commerce Login",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 24));
        titleLabel.setForeground(new Color(25, 25, 112));
        mainPanel.add(titleLabel);
        mainPanel.add(new JLabel(""));

        mainPanel.add(new JLabel("Username:"));
        usernameField = new JTextField();
        usernameField.setFont(new Font("Arial", Font.PLAIN, 14));
        mainPanel.add(usernameField);

        mainPanel.add(new JLabel("Password:"));
        passwordField = new JPasswordField();
        passwordField.setFont(new Font("Arial", Font.PLAIN, 14));
        mainPanel.add(passwordField);

        mainPanel.add(new JLabel("Login As:"));
        String[] roles = {"User", "Admin"};
        roleComboBox = new JComboBox<>(roles);
        roleComboBox.setFont(new Font("Arial", Font.PLAIN, 14));
        mainPanel.add(roleComboBox);

        JLabel testLabel = new JLabel("Test: admin/admin123 |
Akhil/Akhil123", SwingConstants.CENTER);
        testLabel.setFont(new Font("Arial", Font.ITALIC, 10));
        testLabel.setForeground(Color.GRAY);
        mainPanel.add(testLabel);
        mainPanel.add(new JLabel(""));

        JButton loginBtn = new JButton("Login");
        loginBtn.setFont(new Font("Arial", Font.BOLD, 14));
        loginBtn.setBackground(new Color(144, 238, 144));
        loginBtn.setForeground(Color.BLACK);
```

```
      loginBtn.addActionListener(e -> login());
      mainPanel.add(loginBtn);

      JButton exitBtn = new JButton("Exit");
      exitBtn.setFont(new Font("Arial", Font.BOLD, 14));
      exitBtn.setBackground(new Color(255, 182, 193));
      exitBtn.setForeground(Color.BLACK);
      exitBtn.addActionListener(e -> System.exit(0));
      mainPanel.add(exitBtn);

      JLabel registerLabel = new JLabel("Don't have an account?",
SwingConstants.CENTER);
      registerLabel.setFont(new Font("Arial", Font.PLAIN, 12));
      mainPanel.add(registerLabel);

      JButton registerBtn = new JButton("Register Here");
      registerBtn.setFont(new Font("Arial", Font.BOLD, 14));
      registerBtn.setBackground(new Color(255, 215, 0));
      registerBtn.setForeground(Color.BLACK);
      registerBtn.addActionListener(e -> new RegisterPage());
      mainPanel.add(registerBtn);

      add(mainPanel);
      setVisible(true);
   }

   private void login() {
      String username = usernameField.getText().trim();
      String password = new String(passwordField.getPassword());
      String selectedRole =
roleComboBox.getSelectedItem().toString().toLowerCase();

      if (username.isEmpty() || password.isEmpty()) {
         JOptionPane.showMessageDialog(this, "Please enter username and
password!");
         return;
      }

      try (Connection conn = ECommerceApp.getConnection()) {
         String query = "SELECT * FROM users WHERE username=? AND
password=? AND role=?";
         PreparedStatement pst = conn.prepareStatement(query);
```

```
            pst.setString(1, username);
            pst.setString(2, password);
            pst.setString(3, selectedRole);
            ResultSet rs = pst.executeQuery();

            if (rs.next()) {
                ECommerceApp.currentUserId = rs.getInt("user_id");
                ECommerceApp.currentUsername = rs.getString("username");
                ECommerceApp.currentUserRole = rs.getString("role");

                JOptionPane.showMessageDialog(this, "✅ Welcome, " +
ECommerceApp.currentUsername + "!");
                dispose();

                if ("admin".equals(selectedRole)) {
                    new AdminDashboard();
                } else {
                    new UserDashboard();
                }
            } else {
                JOptionPane.showMessageDialog(this,
                    "❌ Invalid credentials or wrong role selected!",
                    "Login Failed",
                    JOptionPane.ERROR_MESSAGE);
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Database Error: " +
ex.getMessage());
        }
    }
}
```

5. Register Page

```
package com.ecommerce;

import javax.swing.*;
import java.awt.*;
import java.sql.*;

public class RegisterPage extends JFrame {
    private JTextField usernameField, emailField;
    private JPasswordField passwordField, confirmPasswordField;

    public RegisterPage() {
        setTitle("User Registration");
        setSize(450, 450);
        setDefaultCloseOperation(DISPOSE_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(20, 30, 20,
30));
        mainPanel.setBackground(new Color(240, 248, 255));

        // Title Panel
        JPanel titlePanel = new JPanel();
        titlePanel.setBackground(new Color(240, 248, 255));
        JLabel titleLabel = new JLabel("📝 Create New Account",
SwingConstants.CENTER);
        titleLabel.setFont(new Font("Arial", Font.BOLD, 26));
        titleLabel.setForeground(new Color(25, 25, 112));
        titlePanel.add(titleLabel);
        mainPanel.add(titlePanel, BorderLayout.NORTH);

        // Form Panel
        JPanel formPanel = new JPanel(new GridLayout(5, 2, 10, 15));
        formPanel.setBackground(new Color(240, 248, 255));
        formPanel.setBorder(BorderFactory.createEmptyBorder(20, 10, 20,
10));

        // Username
        JLabel userLabel = new JLabel("Username:");
        userLabel.setFont(new Font("Arial", Font.BOLD, 14));
        formPanel.add(userLabel);

        usernameField = new JTextField();
        usernameField.setFont(new Font("Arial", Font.PLAIN, 14));
        formPanel.add(usernameField);

        // Email
        JLabel emailLabel = new JLabel("Email:");
        emailLabel.setFont(new Font("Arial", Font.BOLD, 14));
        formPanel.add(emailLabel);

        emailField = new JTextField();
        emailField.setFont(new Font("Arial", Font.PLAIN, 14));
        formPanel.add(emailField);

        // Password
        JLabel passLabel = new JLabel("Password:");
        passLabel.setFont(new Font("Arial", Font.BOLD, 14));
        formPanel.add(passLabel);

        passwordField = new JPasswordField();
        passwordField.setFont(new Font("Arial", Font.PLAIN, 14));
        formPanel.add(passwordField);

        // Confirm Password
        JLabel confirmLabel = new JLabel("Confirm Password:");
        confirmLabel.setFont(new Font("Arial", Font.BOLD, 14));
        formPanel.add(confirmLabel);

        confirmPasswordField = new JPasswordField();
        confirmPasswordField.setFont(new Font("Arial", Font.PLAIN, 14));
        formPanel.add(confirmPasswordField);

        // Info Label
        JLabel infoLabel = new JLabel("(New users will be registered as 'User'
    role)");
        infoLabel.setFont(new Font("Arial", Font.ITALIC, 11));
        infoLabel.setForeground(Color.GRAY);
        formPanel.add(infoLabel);
        formPanel.add(new JLabel(""));

        mainPanel.add(formPanel, BorderLayout.CENTER);

        // Button Panel
        JPanel buttonPanel = new JPanel(new
    FlowLayout(FlowLayout.CENTER, 15, 10));
        buttonPanel.setBackground(new Color(240, 248, 255));

        JButton registerBtn = new JButton("Register");
        registerBtn.setFont(new Font("Arial", Font.BOLD, 15));
        registerBtn.setBackground(new Color(144, 238, 144));
        registerBtn.setForeground(Color.BLACK);
        registerBtn.setPreferredSize(new Dimension(140, 45));
        registerBtn.setFocusPainted(false);
        registerBtn.addActionListener(e -> register());

        JButton cancelBtn = new JButton("Cancel");
        cancelBtn.setFont(new Font("Arial", Font.BOLD, 15));
        cancelBtn.setBackground(new Color(255, 182, 193));
        cancelBtn.setForeground(Color.BLACK);
        cancelBtn.setPreferredSize(new Dimension(140, 45));
        cancelBtn.setFocusPainted(false);
        cancelBtn.addActionListener(e -> dispose());

        buttonPanel.add(registerBtn);
        buttonPanel.add(cancelBtn);

        mainPanel.add(buttonPanel, BorderLayout.SOUTH);

        add(mainPanel);
        setVisible(true);
    }

    private void register() {
        String username = usernameField.getText().trim();
        String email = emailField.getText().trim();
        String password = new String(passwordField.getPassword());
        String confirmPassword = new
String(confirmPasswordField.getPassword());

        // Validation
        if (username.isEmpty() || email.isEmpty() || password.isEmpty() ||
confirmPassword.isEmpty()) {
            JOptionPane.showMessageDialog(this,
                "⚠ Please fill all fields!",
                "Missing Information",
                JOptionPane.WARNING_MESSAGE);
            return;
        }

        if (username.length() < 3) {
            JOptionPane.showMessageDialog(this,
                "Username must be at least 3 characters long!",
            "Invalid Username",
            JOptionPane.WARNING_MESSAGE);
        return;
    }

    if (password.length() < 6) {
        JOptionPane.showMessageDialog(this,
            "Password must be at least 6 characters long!",
            "Weak Password",
            JOptionPane.WARNING_MESSAGE);
        return;
    }

    if (!password.equals(confirmPassword)) {
        JOptionPane.showMessageDialog(this,
            "❌ Passwords don't match!\nPlease re-enter your password.",
            "Password Mismatch",
            JOptionPane.ERROR_MESSAGE);
        return;
    }

    // Register user in database
    try (Connection conn = ECommerceApp.getConnection()) {
        String query = "INSERT INTO users (username, password, role)
VALUES (?, ?, 'user')";
        PreparedStatement pst = conn.prepareStatement(query);
        pst.setString(1, username);
        pst.setString(2, password);

        pst.executeUpdate();

        JOptionPane.showMessageDialog(this,
            "✅ Registration Successful!\n\n" +
            "Username: " + username + "\n" +
            "Role: User\n\n" +
            "You can now login with your credentials.",
            "Success",
            JOptionPane.INFORMATION_MESSAGE);

        dispose();

    } catch (SQLException ex) {
            if (ex.getMessage().contains("Duplicate entry")) {
                JOptionPane.showMessageDialog(this,
                    "✕ Username already exists!\n\nPlease choose a different
username.",
                    "Registration Failed",
                    JOptionPane.ERROR_MESSAGE);
            } else {
                JOptionPane.showMessageDialog(this,
                    "⚠ Database Error!\n\n" + ex.getMessage(),
                    "Error",
                    JOptionPane.ERROR_MESSAGE);
            }
        }
    }
}
```

6. User Dashboard

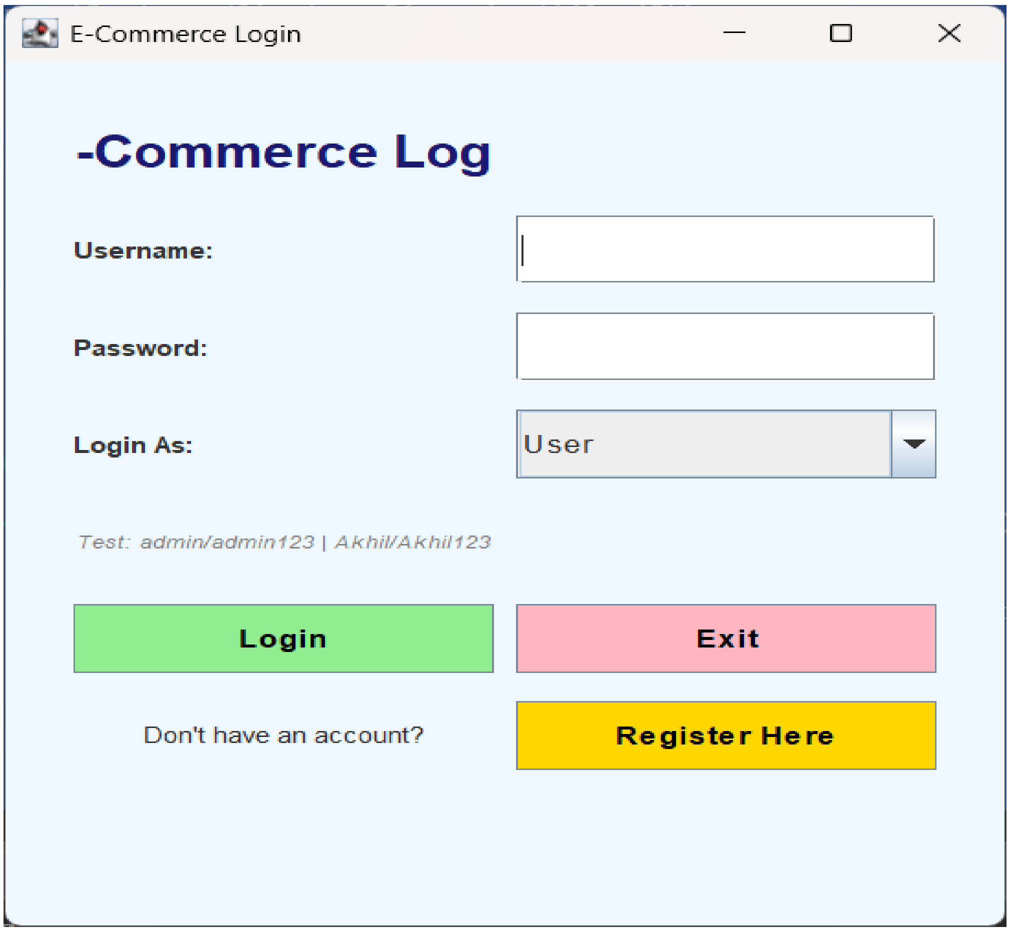
```
package com.ecommerce;

import javax.swing.*;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.sql.*;
import java.util.ArrayList;

public class UserDashboard extends JFrame {
    private JTable productTable;
    private DefaultTableModel tableModel;
    private JTextField searchField;
    private JComboBox<String> categoryCombo;
    private ArrayList<CartItem> cartItems = new ArrayList<>();

    public UserDashboard() {
        setTitle("User Dashboard");
        setSize(1100, 600);
        setDefaultCloseOperation(EXIT_ON_CLOSE);
        setLocationRelativeTo(null);

        JPanel mainPanel = new JPanel(new BorderLayout(10, 10));
        mainPanel.setBorder(BorderFactory.createEmptyBorder(10, 10, 10,
10));
```

```
    // Top Panel
    JPanel topPanel = new JPanel(new BorderLayout());
    topPanel.setBackground(new Color(70, 130, 180));
    topPanel.setBorder(BorderFactory.createEmptyBorder(15, 15, 15,
15));

    JLabel titleLabel = new JLabel("Welcome, " +
ECommerceApp.currentUsername + "!", SwingConstants.CENTER);
    titleLabel.setFont(new Font("Arial", Font.BOLD, 20));
    titleLabel.setForeground(Color.WHITE);

    JButton logoutBtn = new JButton("Logout");
    logoutBtn.setBackground(new Color(255, 182, 193));
    logoutBtn.setForeground(Color.BLACK);
    logoutBtn.addActionListener(e -> logout());

    topPanel.add(titleLabel, BorderLayout.CENTER);
    topPanel.add(logoutBtn, BorderLayout.EAST);
    mainPanel.add(topPanel, BorderLayout.NORTH);

    // Filter Panel
    JPanel filterPanel = new JPanel(new FlowLayout(FlowLayout.LEFT, 10,
10));
    filterPanel.setBackground(Color.WHITE);

    filterPanel.add(new JLabel("Category:"));
    String[] categories = {"All Products", "Smartphones", "Laptops",
"Tablets", "Smartwatches", "Cameras", "Audio"};
    categoryCombo = new JComboBox<>(categories);
    categoryCombo.addActionListener(e -> filterCategory());
    filterPanel.add(categoryCombo);

    filterPanel.add(new JLabel("Search:"));
    searchField = new JTextField(20);
    filterPanel.add(searchField);

    JButton searchBtn = new JButton("Search");
    searchBtn.setBackground(new Color(173, 216, 230));
    searchBtn.setForeground(Color.BLACK);
    searchBtn.addActionListener(e -> searchProducts());
    filterPanel.add(searchBtn);
    JButton refreshBtn = new JButton("Show All");
    refreshBtn.setBackground(new Color(211, 211, 211));
    refreshBtn.setForeground(Color.BLACK);
    refreshBtn.addActionListener(e -> loadProducts());
    filterPanel.add(refreshBtn);

    mainPanel.add(filterPanel, BorderLayout.PAGE_START);

    // Table
    String[] columns = {"ID", "Product", "Description", "Price", "Category",
"Stock"};
    tableModel = new DefaultTableModel(columns, 0) {
      public boolean isCellEditable(int row, int col) { return false; }
    };
    productTable = new JTable(tableModel);
    productTable.setRowHeight(30);
    productTable.getTableHeader().setBackground(new Color(70, 130,
180));
    productTable.getTableHeader().setForeground(Color.BLACK);

    JScrollPane scrollPane = new JScrollPane(productTable);
    mainPanel.add(scrollPane, BorderLayout.CENTER);

    // Bottom Panel
    JPanel bottomPanel = new JPanel(new
FlowLayout(FlowLayout.CENTER, 20, 10));

    JButton addCartBtn = new JButton("Add to Cart");
    addCartBtn.setBackground(new Color(144, 238, 144));
    addCartBtn.setForeground(Color.BLACK);
    addCartBtn.setPreferredSize(new Dimension(150, 40));
    addCartBtn.addActionListener(e -> addToCart());

    JButton viewCartBtn = new JButton("View Cart");
    viewCartBtn.setBackground(new Color(255, 215, 0));
    viewCartBtn.setForeground(Color.BLACK);
    viewCartBtn.setPreferredSize(new Dimension(150, 40));
    viewCartBtn.addActionListener(e -> viewCart());

    bottomPanel.add(addCartBtn);
    bottomPanel.add(viewCartBtn);
```

```
        mainPanel.add(bottomPanel, BorderLayout.SOUTH);

        add(mainPanel);
        loadProducts();
        setVisible(true);
    }

    private void loadProducts() {
        tableModel.setRowCount(0);
        try (Connection conn = ECommerceApp.getConnection()) {
            Statement stmt = conn.createStatement();
            ResultSet rs = stmt.executeQuery("SELECT * FROM products WHERE
stock_quantity > 0");

            while (rs.next()) {
                tableModel.addRow(new Object[] {
                    rs.getInt("product_id"),
                    rs.getString("product_name"),
                    rs.getString("description"),
                    rs.getDouble("price"),
                    rs.getString("category"),
                    rs.getInt("stock_quantity")
                });
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void filterCategory() {
        String category = (String) categoryCombo.getSelectedItem();
        tableModel.setRowCount(0);

        try (Connection conn = ECommerceApp.getConnection()) {
            String query = "All Products".equals(category) ?
                "SELECT * FROM products WHERE stock_quantity > 0" :
                "SELECT * FROM products WHERE category = ? AND
stock_quantity > 0";

            PreparedStatement pst = conn.prepareStatement(query);
            if (!"All Products".equals(category)) {
                pst.setString(1, category);
            }

            ResultSet rs = pst.executeQuery();
            while (rs.next()) {
                tableModel.addRow(new Object[] {
                    rs.getInt("product_id"),
                    rs.getString("product_name"),
                    rs.getString("description"),
                    rs.getDouble("price"),
                    rs.getString("category"),
                    rs.getInt("stock_quantity")
                });
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void searchProducts() {
        String term = searchField.getText().trim();
        if (term.isEmpty()) {
            loadProducts();
            return;
        }

        tableModel.setRowCount(0);
        try (Connection conn = ECommerceApp.getConnection()) {
            String query = "SELECT * FROM products WHERE (product_name
LIKE ? OR description LIKE ?) AND stock_quantity > 0";
            PreparedStatement pst = conn.prepareStatement(query);
            pst.setString(1, "%" + term + "%");
            pst.setString(2, "%" + term + "%");
            ResultSet rs = pst.executeQuery();

            while (rs.next()) {
                tableModel.addRow(new Object[] {
                    rs.getInt("product_id"),
                    rs.getString("product_name"),
                    rs.getString("description"),
                    rs.getDouble("price"),
                    rs.getString("category"),
                    rs.getInt("stock_quantity")
                });
            }
        } catch (SQLException ex) {
            JOptionPane.showMessageDialog(this, "Error: " + ex.getMessage());
        }
    }

    private void addToCart() {
        int row = productTable.getSelectedRow();
        if (row < 0) {
            JOptionPane.showMessageDialog(this, "Select a product!");
            return;
        }

        int id = (int) tableModel.getValueAt(row, 0);
        String name = tableModel.getValueAt(row, 1).toString();
        double price = (double) tableModel.getValueAt(row, 3);
        int stock = (int) tableModel.getValueAt(row, 5);

        String qtyStr = JOptionPane.showInputDialog(this, "Quantity (Max: " +
stock + "):", "1");
        if (qtyStr == null) return;

        try {
            int qty = Integer.parseInt(qtyStr);
            if (qty > 0 && qty <= stock) {
                cartItems.add(new CartItem(id, name, price, qty));
                JOptionPane.showMessageDialog(this, "Added to cart!");
            } else {
                JOptionPane.showMessageDialog(this, "Invalid quantity!");
            }
        } catch (NumberFormatException ex) {
            JOptionPane.showMessageDialog(this, "Invalid number!");
        }
    }

    private void viewCart() {
        if (cartItems.isEmpty()) {
            JOptionPane.showMessageDialog(this, "Cart is empty!");
            return;
        }
        new CartPage(cartItems, this);
    }

    private void logout() {
        dispose();
        new LoginPage();
    }
}

class CartItem {
    int productId;
    String productName;
    double price;
    int quantity;

    public CartItem(int productId, String productName, double price, int
quantity) {
        this.productId = productId;
        this.productName = productName;
        this.price = price;
        this.quantity = quantity;
    }

    public double getTotal() {
        return price * quantity;
    }
}
```

CHAPTER 5

SCREEN SHOTS;


E-Commerce Login

-Commerce Log

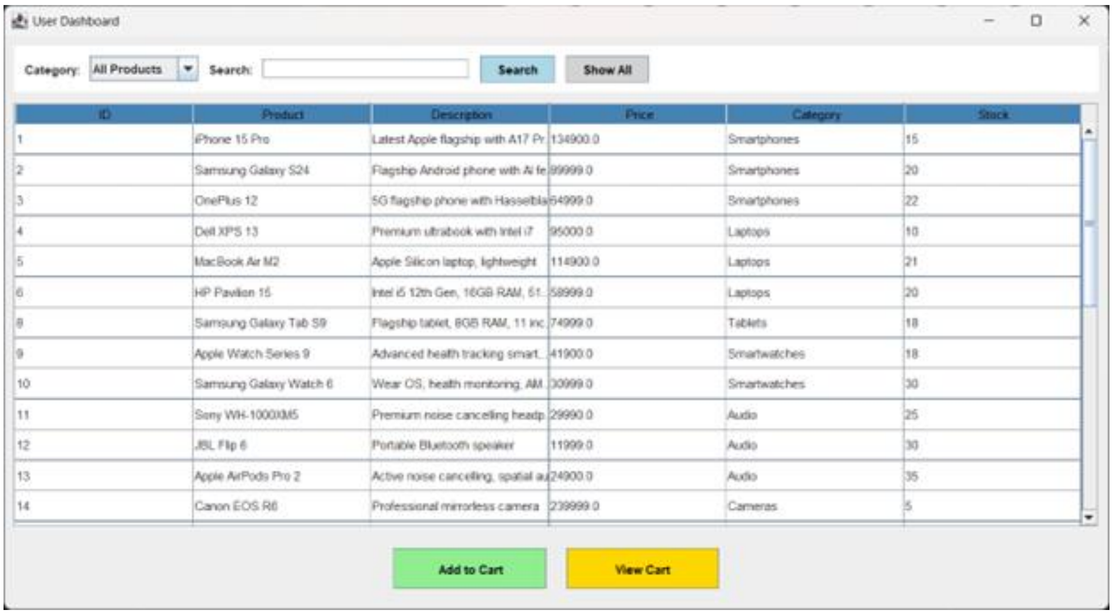
Username:

Password:

Login As: User

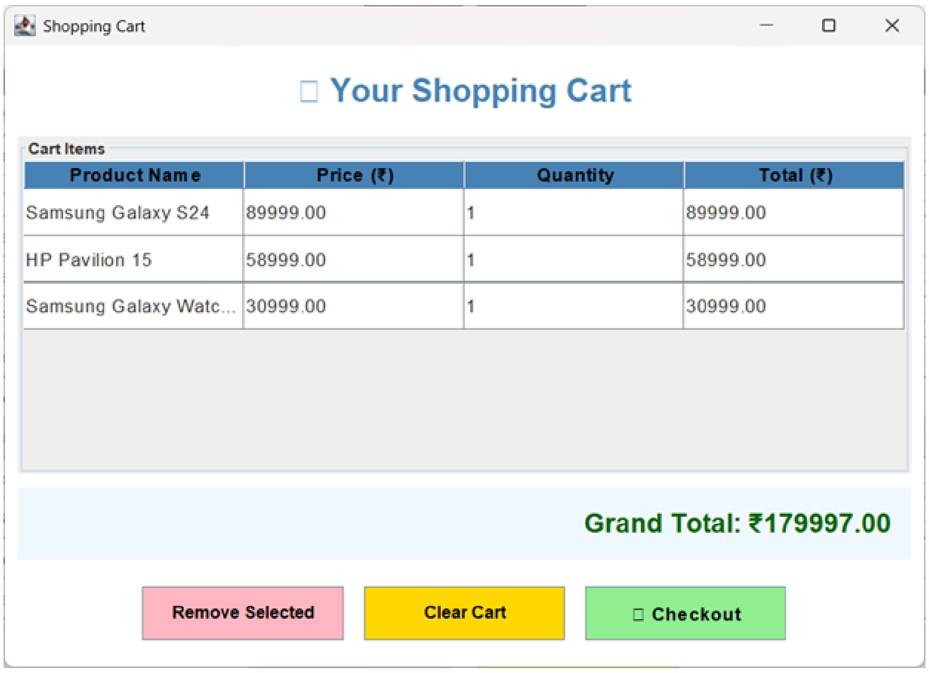
Test: admin/admin123 | Akhil/Akhil123

Fig 5.1 Login page



ID	Product	Description	Price	Category	Stock
1	iPhone 15 Pro	Latest Apple flagship with A17 Pr	134900.0	Smartphones	15
2	Samsung Galaxy S24	Flagship Android phone with AI fe	89999.0	Smartphones	20
3	OnePlus 12	5G flagship phone with Hasselbla	64999.0	Smartphones	22
4	Dell XPS 13	Premium ultrabook with Intel i7	95000.0	Laptops	10
5	MacBook Air M2	Apple Silicon laptop, lightweight	114900.0	Laptops	21
6	HP Pavilion 15	Intel i5 12th Gen, 16GB RAM, 51...	58999.0	Laptops	20
8	Samsung Galaxy Tab S9	Flagship tablet, 8GB RAM, 11 inc...	74999.0	Tablets	18
9	Apple Watch Series 9	Advanced health tracking smart...	41900.0	Smartwatches	18
10	Samsung Galaxy Watch 6	Wear OS, health monitoring, AM...	30999.0	Smartwatches	30
11	Sony WH-1000XM5	Premium noise cancelling headp...	29990.0	Audio	25
12	JBL Flip 6	Portable Bluetooth speaker	11999.0	Audio	30
13	Apple AirPods Pro 2	Active noise cancelling, spatial au	24900.0	Audio	35
14	Canon EOS R6	Professional mirrorless camera	239999.0	Cameras	5

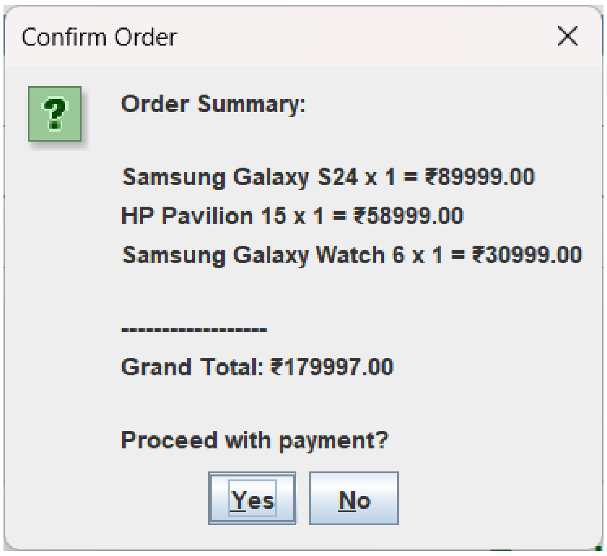
Fig 5.2 Product Frame



Product Name	Price (₹)	Quantity	Total (₹)
Samsung Galaxy S24	89999.00	1	89999.00
HP Pavilion 15	58999.00	1	58999.00
Samsung Galaxy Watc...	30999.00	1	30999.00

Grand Total: ₹179997.00

Fig 5.3 Cart Frame


Confirm Order

Order Summary:

Samsung Galaxy S24 x 1 = ₹89999.00
HP Pavilion 15 x 1 = ₹58999.00
Samsung Galaxy Watch 6 x 1 = ₹30999.00

Grand Total: ₹179997.00

Proceed with payment?

Fig 5.4 Checkout Frame

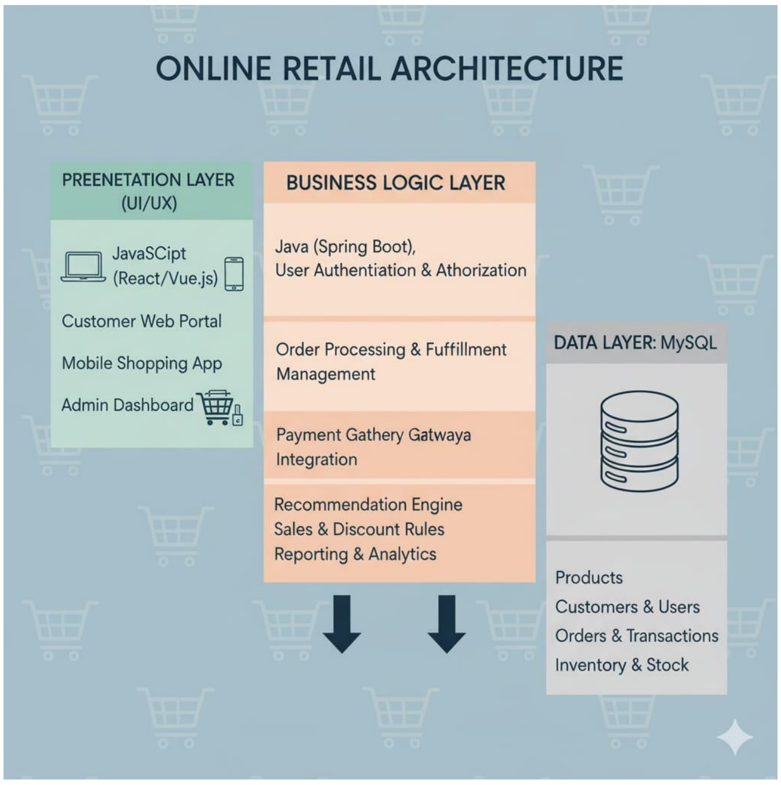


Fig 5.5 Architecture

CHAPTER 6

CONCLUSION AND FUTURE ENHANCEMENT

The E-Commerce Desktop Application built using **Java Swing** and **MySQL (JAVA-JDBC)** successfully fulfils the core requirements of a basic digital retail solution. The system offers two distinct user interfaces — a **customer module** for browsing and shopping, and an **admin module** for complete product management.

By integrating GUI design principles with MySQL-backed JDBC connectivity, the application ensures smooth interaction between frontend components and backend data operations. Features such as login and registration, cart management simulation, and robust CRUD-based inventory control contribute to a seamless e-commerce experience.

The project successfully demonstrates principles of modular design, data persistence, and event-driven programming. It is highly suitable for a standalone environment such as college projects, small shops, or offline demos, and provides a solid foundation for further feature-rich expansion.

To evolve this project into a more comprehensive retail platform, the following future enhancements are recommended:

Even though the current version of the application meets its foundational goals, the following improvements can be added to make it more aligned with modern e-commerce platforms:

1. **Complete Checkout and Payment Flow**
   Implement full cart-to-order placement with quantity updates, payment simulation (or gateway integration), and order confirmation.
2. **Order Management and History Modules**
   add a dedicated order table and history feature so users and admins can track purchases, downloads invoices, and manage order statuses.
3. **User Profile and Security Enhancements**
   Migrate to **hashed password storage** using SHA-256 or crypt; allow users to update personal data, reset passwords, and upload profile pictures.

4. **Product Image and Multimedia Support**
extend the admin panel to allow image uploads per product; display images via Label or scaled icon renderer in Table.

5. **Reporting and Analytics**
Provide admin reports such as sales summary, most-viewed items, low-stock alerts, and category-wise revenue insights using SQL aggregation functions.

6. **Multi-Platform Expansion**
Convert the existing desktop implementation to a JavaFX UI or migrate to a web-based framework such as Spring Boot + JSP/HTML for broader accessibility.

REFERENCES

https://docs.oracle.com/javase/8/docs/api/javax/swing/packag esummary.html
https://docs.oracle.com/javase/tutorial/jdbc/index.html

https://dev.mysql.com/doc/refman/8.0/en/

https://dev.mysql.com/doc/connector-j/8.0/en/

https://www.oracle.com/java/technologies/data-access-object.html