

# INVENTORY MANAGEMENT SYSTEM

## AIM:

To develop a Java-based Inventory Management System that simplifies stock tracking by enabling efficient addition, updating, and removal of items. The system aims to streamline inventory operations, reduce errors, and ensure real-time data accuracy.

## ALGORITHM:

Algorithm: Inventory Management System

### 1. Initialization

1.1 Start the program.

1.2 Initialize the inventory database (can be a file, an in-memory structure, or a database connection).

1.3 Define classes for Item and InventoryManager.

Item: Fields for item ID, name, quantity, price, and category.

InventoryManager: Methods for adding, removing, updating, and searching items.

### 2. Main Menu

2.1 Display options to the user:

Add a new item

Remove an item

Update item details

Search for an item

View all items

Exit

2.2 Capture the user's choice.

### 3. Add Item

3.1 Prompt the user for item details (ID, name, quantity, price, category).

3.2 Validate input:

Ensure the ID is unique.

Ensure quantity and price are non-negative.

3.3 Add the item to the inventory.

3.4 Confirm success to the user.

#### **4. Remove Item**

4.1 Prompt the user for the item ID to remove.

4.2 Search for the item in the inventory:

If found, delete it.

If not found, notify the user.

4.3 Confirm removal or failure to the user.

#### **5. Update Item**

5.1 Prompt the user for the item ID to update.

5.2 Search for the item:

If found, display its current details.

If not found, notify the user.

5.3 Prompt the user for updated details (quantity, price, or name).

5.4 Validate input and update the item in the inventory.

5.5 Confirm success or failure to the user.

#### **6. Search for an Item**

6.1 Prompt the user for the search criteria:

By ID

By Name

By Category

6.2 Search the inventory based on the chosen criteria.

6.3 Display matching items or notify if no matches are found.

#### **7. View All Items**

7.1 Retrieve and display all items in the inventory in a tabular format.

7.2 Allow sorting by name, price, or quantity (optional).

#### **8. Exit**

8.1 Save inventory changes (if applicable).

8.2 Close the program gracefully.

#### **9. Error Handling**

9.1 Handle invalid inputs (e.g., non-numeric quantities or prices).

9.2 Handle unexpected errors gracefully to ensure program stability.

## CODE:

```
CREATE DATABASE `inventory` ;
```

```
USE `inventory`;
```

```
CREATE TABLE `currentstock` (
```

```
    `productcode` varchar(45),
```

```
    `quantity` int,
```

```
    PRIMARY KEY (`productcode`)
```

```
);
```

```
INSERT INTO `currentstock` VALUES
```

```
('prod1',146),('prod2',100),('prod3',202),('prod4',172),('prod5',500),('prod6',500),('prod7',10),('prod8',20);
```

```
CREATE TABLE `customers` (
```

```
    `cid` int NOT NULL AUTO_INCREMENT,
```

```
    `customercode` varchar(45),
```

```
    `fullname` varchar(45),
```

```
    `location` varchar(45),
```

```
    `phone` varchar(45),
```

```
    PRIMARY KEY (`cid`)
```

```
);
```

```
INSERT INTO `customers` VALUES (301,'vip1','John Seed','New
```

```
York','9818562354'),(302,'vip2','Jacob Seed','Texas','9650245489'),(303,'std1','Ajay
```

```
Kumar','Mumbai','9236215622'),(304,'std2','Astha
```

```
Walia','Chandigarh','8854612478'),(306,'vip3','Madhu Chitkara','Chandigarh','9826546182');
```

```
CREATE TABLE `products` (
```

```
    `pid` int NOT NULL AUTO_INCREMENT,
```

```
    `productcode` varchar(45),
```

```
    `productname` varchar(45),
```

```
    `costprice` double,
```

```
    `sellprice` double,
```

```
    `brand` varchar(45),
```

```
    PRIMARY KEY (`pid`),
```

```

    UNIQUE KEY `productcode_UNIQUE` (`productcode`)
);

INSERT INTO `products` VALUES
(111,'prod1','Laptop',85000,90000,'Dell'),(112,'prod2','Laptop',70000,72000,'HP'),(113,'prod3',
'Mobile',60000,64000,'Apple'),(114,'prod4','Mobile',50000,51000,'Samsung'),(121,'prod5','C
harger',2000,2100,'Apple'),(122,'prod6','Mouse',1700,1900,'Dell'),(128,'prod7','Power
Adapter',3000,3500,'Dell'),(129,'prod8','Smart Watch',15000,17000,'Apple');

CREATE TABLE `purchaseinfo` (
  `purchaseID` int NOT NULL AUTO_INCREMENT,
  `suppliercode` varchar(45),
  `productcode` varchar(45),
  `date` varchar(45),
  `quantity` int,
  `totalcost` double,
  PRIMARY KEY (`purchaseID`)
);

INSERT INTO `purchaseinfo` VALUES (1001,'sup1','prod1','Wed Jan 14 00:15:19 IST
2021',10,850000),(1002,'sup1','prod6','Wed Jan 14 00:15:19 IST
2021',20,340000),(1003,'sup2','prod3','Wed Jan 14 00:15:19 IST
2021',5,300000),(1004,'sup2','prod5','Wed Jan 14 00:15:19 IST
2021',5,100000),(1005,'sup3','prod2','Wed Jan 14 00:15:19 IST
2021',2,140000),(1006,'sup4','prod4','Wed Jan 14 00:15:19 IST
2021',2,100000),(1009,'sup2','prod3','Wed Sep 01 04:11:13 IST
2021',2,120000),(1010,'sup1','prod7','Wed Sep 01 04:25:06 IST
2021',10,300000),(1011,'sup2','prod8','Fri Sep 03 00:00:00 IST 2021',20,300000);

CREATE TABLE `users` (
  `id` int NOT NULL AUTO_INCREMENT,
  `name` varchar(45),
  `location` varchar(45),
  `phone` varchar(10),
  `username` varchar(20),
  `password` varchar(200),
  `usertype` varchar(45),
  PRIMARY KEY (`id`)
);

```

```
INSERT INTO `users` VALUES
(20,'Admin','Local','9876543210','root','root','ADMINISTRATOR');
```

## **DATABASE CONNECTION MODULE**

```
1. import java.sql.Connection;
2. import java.sql.DriverManager;
3. import java.sql.ResultSet;
4. import java.sql.Statement;
5.
6. public class ConnectionFactory {
7.
8.     Connection conn = null;
9.     Statement statement = null;
10.    ResultSet resultSet = null;
11.
12.    public ConnectionFactory(){
13.        try {
14.            Class.forName("com.mysql.cj.jdbc.Driver");
15.            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/inventory",
"root", "root");
16.            statement = conn.createStatement();
17.        } catch (Exception e) {
18.        }
19.    }
20.
21.    public Connection getConn() {
22.        try {
23.            Class.forName("com.mysql.cj.jdbc.Driver");
24.            conn = DriverManager.getConnection("jdbc:mysql://localhost:3306/inventory",
"root", "root");
25.            System.out.println("Connected successfully.");
26.        } catch (Exception e) {
27.        }
28.        return conn;
29.    }
30.
31.    //Login verification method
32.    public boolean checkLogin(String username, String password, String userType){
33.        String query = "SELECT * FROM users WHERE username="
34.            + username
35.            + " AND password="
36.            + password
37.            + " AND usertype="
38.            + userType
39.            + " LIMIT 1";
```

```
40.
41.     try {
42.         resultSet = statement.executeQuery(query);
43.         if(resultSet.next()) return true;
44.     } catch (Exception ex) {
45.     }
46.     return false;
47. }
48.
49. }
```

### **CustomerDAO MODULE**

```
1. import javax.swing.*;
2. import java.sql.*;
3. import javax.swing.table.DefaultTableModel;
4. import java.util.Locale;
5. import java.util.Vector;
6.
7. class CustomerDTO {
8.
9.     int custID;
10.    String custCode, fullName, location, phone;
11.    double debit, credit, balance;
12.
13.    public int getCustID() {
14.        return custID;
15.    }
16.
17.    public void setCustID(int custID) {
18.        this.custID = custID;
19.    }
20.
21.    public String getCustCode() {
22.        return custCode;
23.    }
24.
25.    public void setCustCode(String custCode) {
26.        this.custCode = custCode;
27.    }
28.
29.    public String getFullName() {
30.        return fullName;
31.    }
32.
33.    public void setFullName(String fullName) {
```

```
34.     this.fullName = fullName;
35. }
36.
37. public String getLocation() {
38.     return location;
39. }
40.
41. public void setLocation(String location) {
42.     this.location = location;
43. }
44.
45. public String getPhone() {
46.     return phone;
47. }
48.
49. public void setPhone(String phone) {
50.     this.phone = phone;
51. }
52.
53. public double getDebit() {
54.     return debit;
55. }
56.
57. public void setDebit(double debit) {
58.     this.debit = debit;
59. }
60.
61. public double getCredit() {
62.     return credit;
63. }
64.
65. public void setCredit(double credit) {
66.     this.credit = credit;
67. }
68.
69. public double getBalance() {
70.     return balance;
71. }
72.
73. public void setBalance(double balance) {
74.     this.balance = balance;
75. }
76.
77. }
78.
79.
```

```

80. public class CustomerDAO {
81.     Connection conn = null;
82.     PreparedStatement prepStatement= null;
83.     Statement statement = null;
84.     ResultSet resultSet = null;
85.
86.     public CustomerDAO() {
87.         try {
88.             conn = new ConnectionFactory().getConn();
89.             statement = conn.createStatement();
90.         } catch (SQLException e) {
91.             e.printStackTrace();
92.         }
93.     }
94.
95.     // Methods to add new custoemr
96.     public void addCustomerDAO(CustomerDTO customerDTO) {
97.         try {
98.             String query = "SELECT * FROM customers WHERE fullname="
99.                 +customerDTO.getFullName()
100.                 + " AND location="
101.                 +customerDTO.getLocation()
102.                 + " AND phone="
103.                 +customerDTO.getPhone()
104.                 + """;
105.             resultSet = statement.executeQuery(query);
106.             if (resultSet.next())
107.                 JOptionPane.showMessageDialog(null, "Customer already exists.");
108.             else
109.                 addFunction(customerDTO);
110.         } catch (SQLException e) {
111.             e.printStackTrace();
112.         }
113.     }
114.     public void addFunction(CustomerDTO customerDTO) {
115.         try {
116.             String query = "INSERT INTO customers VALUES(null,?,?,?,?)";
117.             prepStatement = conn.prepareStatement(query);
118.             prepStatement.setString(1, customerDTO.getCustCode());
119.             prepStatement.setString(2, customerDTO.getFullName());
120.             prepStatement.setString(3, customerDTO.getLocation());
121.             prepStatement.setString(4, customerDTO.getPhone());
122.             prepStatement.executeUpdate();
123.             JOptionPane.showMessageDialog(null, "New customer has been added.");
124.         } catch (SQLException e) {
125.             e.printStackTrace();

```



```

126.     }
127.
128.     }
129.
130.     // Method to edit existing customer details
131.     public void editCustomerDAO(CustomerDTO customerDTO) {
132.         try {
133.             String query = "UPDATE customers SET fullname=?,location=?,phone=?
WHERE customercode=?";
134.             preparedStatement = conn.prepareStatement(query);
135.             preparedStatement.setString(1, customerDTO.getFullName());
136.             preparedStatement.setString(2, customerDTO.getLocation());
137.             preparedStatement.setString(3, customerDTO.getPhone());
138.             preparedStatement.setString(4, customerDTO.getCustCode());
139.             preparedStatement.executeUpdate();
140.             JOptionPane.showMessageDialog(null, "Customer details have been
updated.");
141.         } catch (SQLException e) {
142.             e.printStackTrace();
143.         }
144.     }
145.
146.     // Method to delete existing customer
147.     public void deleteCustomerDAO(String custCode) {
148.         try {
149.             String query = "DELETE FROM customers WHERE customercode="
+custCode+ """;
150.             statement.executeUpdate(query);
151.             JOptionPane.showMessageDialog(null, "Customer removed.");
152.         } catch (SQLException e) {
153.             e.printStackTrace();
154.         }
155.     }
156.
157.     // Method to retrieve data set to be displayed
158.     public ResultSet getQueryResult() {
159.         try {
160.             String query = "SELECT customercode,fullname,location,phone FROM
customers";
161.             resultSet = statement.executeQuery(query);
162.         } catch (SQLException e) {
163.             e.printStackTrace();
164.         }
165.         return resultSet;
166.     }
167.

```

```

168.    // Method to retrieve search data
169.    public ResultSet getCustomerSearch(String text) {
170.        try {
171.            String query = "SELECT customercode,fullname,location,phone FROM
            customers " +
172.                "WHERE customercode LIKE '%" + text + "%' OR fullname LIKE
            '%" + text + "%' OR " +
173.                "location LIKE '%" + text + "%' OR phone LIKE '%" + text + "%'";
174.            resultSet = statement.executeQuery(query);
175.        } catch (SQLException e) {
176.            e.printStackTrace();
177.        }
178.        return resultSet;
179.    }
180.
181.    public ResultSet getCustName(String custCode) {
182.        try {
183.            String query = "SELECT * FROM customers WHERE customercode="
            + custCode + """;
184.            resultSet = statement.executeQuery(query);
185.        } catch (SQLException e) {
186.            e.printStackTrace();
187.        }
188.        return resultSet;
189.    }
190.
191.    public ResultSet getProdName(String prodCode) {
192.        try {
193.            String query = "SELECT productname,currentstock.quantity FROM products
            " +
194.                "INNER JOIN currentstock ON
            products.productcode=currentstock.productcode " +
195.                "WHERE currentstock.productcode=" + prodCode + """;
196.            resultSet = statement.executeQuery(query);
197.        } catch (SQLException e) {
198.            e.printStackTrace();
199.        }
200.        return resultSet;
201.    }
202.
203.    // Method to display data set in tabular form
204.    public DefaultTableModel buildTableModel(ResultSet resultSet) throws
        SQLException {
205.        ResultSetMetaData metaData = resultSet.getMetaData();
206.        Vector<String> columnNames = new Vector<String>();
207.        int colCount = metaData.getColumnCount();

```

```

208.
209.     for (int col=1; col <= colCount; col++){
210.         columnNames.add(metaData.getColumnName(col).toUpperCase(Locale.ROOT));
211.     }
212.
213.     Vector<Vector<Object>> data = new Vector<Vector<Object>>();
214.     while (resultSet.next()) {
215.         Vector<Object> vector = new Vector<Object>();
216.         for (int col=1; col<=colCount; col++) {
217.             vector.add(resultSet.getObject(col));
218.         }
219.         data.add(vector);
220.     }
221.     return new DefaultTableModel(data, columnNames);
222. }
223.
224. }

```

### **ProductDAO Module**

```

1. import javax.swing.*;
2. import java.sql.*;
3. import java.util.Vector;
4. import javax.swing.table.DefaultTableModel;
5. import java.util.Locale;
6.
7. class ProductDTO {
8.
9.     int prodID, quantity, userID;
10.    double costPrice, sellPrice;
11.    Double totalCost, totalRevenue;
12.    String prodCode, prodName, date, suppCode, custCode, custName, brand;
13.
14.    public int getProdID() {
15.        return prodID;
16.    }
17.
18.    public void setProdID(int prodID) {
19.        this.prodID = prodID;
20.    }
21.
22.    public int getQuantity() {
23.        return quantity;
24.    }
25.
26.    public void setQuantity(int quantity) {

```

```
27.     this.quantity = quantity;
28. }
29.
30. public int getUserID() {
31.     return userID;
32. }
33.
34. public void setUserID(int userID) {
35.     this.userID = userID;
36. }
37.
38. public double getCostPrice() {
39.     return costPrice;
40. }
41.
42. public void setCostPrice(double costPrice) {
43.     this.costPrice = costPrice;
44. }
45.
46. public double getSellPrice() {
47.     return sellPrice;
48. }
49.
50. public void setSellPrice(double sellPrice) {
51.     this.sellPrice = sellPrice;
52. }
53.
54. public Double getTotalCost() {
55.     return totalCost;
56. }
57.
58. public void setTotalCost(Double totalCost) {
59.     this.totalCost = totalCost;
60. }
61.
62. public Double getTotalRevenue() {
63.     return totalRevenue;
64. }
65.
66. public void setTotalRevenue(Double totalRevenue) {
67.     this.totalRevenue = totalRevenue;
68. }
69.
70. public String getProdCode() {
71.     return prodCode;
72. }
```

```
73.
74. public void setProdCode(String prodCode) {
75.     this.prodCode = prodCode;
76. }
77.
78. public String getProdName() {
79.     return prodName;
80. }
81.
82. public void setProdName(String prodName) {
83.     this.prodName = prodName;
84. }
85.
86. public String getDate() {
87.     return date;
88. }
89.
90. public void setDate(String date) {
91.     this.date = date;
92. }
93.
94. public String getSuppCode() {
95.     return suppCode;
96. }
97.
98. public void setSuppCode(String suppCode) {
99.     this.suppCode = suppCode;
100. }
101.
102. public String getCustCode() {
103.     return custCode;
104. }
105.
106. public void setCustCode(String custCode) {
107.     this.custCode = custCode;
108. }
109.
110. public String getCustName() {
111.     return custName;
112. }
113.
114. public void setCustName(String custName) {
115.     this.custName = custName;
116. }
117.
118. public String getBrand() {
```

```

119.     return brand;
120. }
121.
122. public void setBrand(String brand) {
123.     this.brand = "Dummy Brand";
124. }
125. }
126.
127. public class ProductDAO {
128.
129.     Connection conn = null;
130.     PreparedStatement prepStatement = null;
131.     PreparedStatement prepStatement2 = null;
132.     Statement statement = null;
133.     Statement statement2 = null;
134.     ResultSet resultSet = null;
135.
136.     public ProductDAO() {
137.         try {
138.             conn = new ConnectionFactory().getConn();
139.             statement = conn.createStatement();
140.             statement2 = conn.createStatement();
141.         } catch (Exception ex) {
142.         }
143.     }
144.
145.     public ResultSet getSuppInfo() {
146.         try {
147.             String query = "SELECT * FROM suppliers";
148.             resultSet = statement.executeQuery(query);
149.         } catch (Exception e) {
150.         }
151.         return resultSet;
152.     }
153.
154.     public ResultSet getCustInfo() {
155.         try {
156.             String query = "SELECT * FROM customers";
157.             resultSet = statement.executeQuery(query);
158.         } catch (Exception e) {
159.         }
160.         return resultSet;
161.     }
162.
163.     public ResultSet getProdStock() {
164.         try {

```

```

165.         String query = "SELECT * FROM currentstock";
166.         resultSet = statement.executeQuery(query);
167.     } catch (Exception e) {
168.     }
169.     return resultSet;
170. }
171.
172. public ResultSet getProdInfo() {
173.     try {
174.         String query = "SELECT * FROM products";
175.         resultSet = statement.executeQuery(query);
176.     } catch (Exception e) {
177.     }
178.     return resultSet;
179. }
180.
181. public Double getProdCost(String prodCode) {
182.     Double costPrice = null;
183.     try {
184.         String query = "SELECT costprice FROM products WHERE productcode="
+prodCode+ "";
185.         resultSet = statement.executeQuery(query);
186.         if (resultSet.next())
187.             costPrice = resultSet.getDouble("costprice");
188.     } catch (Exception e) {
189.         e.printStackTrace();
190.     }
191.     return costPrice;
192. }
193.
194. public Double getProdSell(String prodCode) {
195.     Double sellPrice = null;
196.     try {
197.         String query = "SELECT sellprice FROM products WHERE productcode="
+prodCode+ "";
198.         resultSet = statement.executeQuery(query);
199.         if (resultSet.next())
200.             sellPrice = resultSet.getDouble("sellprice");
201.     } catch (Exception e) {
202.         e.printStackTrace();
203.     }
204.     return sellPrice;
205. }
206.
207. String suppCode;
208. public String getSuppCode(String suppName) {

```

```

209.         try {
210.             String query = "SELECT suppliercode FROM suppliers WHERE fullname="
+suppName+ "";
211.             resultSet = statement.executeQuery(query);
212.             while (resultSet.next()) {
213.                 suppCode = resultSet.getString("suppliercode");
214.             }
215.         } catch (SQLException e) {
216.         }
217.         return suppCode;
218.     }
219.
220.     String prodCode;
221.     public String getProdCode(String prodName) {
222.         try {
223.             String query = "SELECT productcode FROM products WHERE
productname=" +prodName+ "";
224.             resultSet = statement.executeQuery(query);
225.             while (resultSet.next()) {
226.                 suppCode = resultSet.getString("productcode");
227.             }
228.         } catch (SQLException e) {
229.         }
230.         return prodCode;
231.     }
232.
233.     String custCode;
234.     public String getCustCode(String custName) {
235.         try {
236.             String query = "SELECT customercode FROM suppliers WHERE fullname="
+custName+ "";
237.             resultSet = statement.executeQuery(query);
238.             while (resultSet.next()) {
239.                 suppCode = resultSet.getString("customercode");
240.             }
241.         } catch (SQLException e) {
242.         }
243.         return custCode;
244.     }
245.
246.     // Method to check for availability of stock in Inventory
247.     boolean flag = false;
248.     public boolean checkStock(String prodCode) {
249.         try {
250.             String query = "SELECT * FROM currentstock WHERE productcode="
+prodCode+ "";

```



```

251.         resultSet = statement.executeQuery(query);
252.         while (resultSet.next()) {
253.             flag = true;
254.         }
255.     } catch (SQLException e) {
256.     }
257.     return flag;
258. }
259.
260. // Methods to add a new product
261. public void addProductDAO(ProductDTO productDTO) {
262.     try {
263.         String query = "SELECT * FROM products WHERE productname="
264.             + productDTO.getProdName()
265.             + " AND costprice="
266.             + productDTO.getCostPrice()
267.             + " AND sellprice="
268.             + productDTO.getSellPrice()
269.             + " AND brand="
270.             + productDTO.getBrand()
271.             + "";
272.         resultSet = statement.executeQuery(query);
273.         if (resultSet.next())
274.             JOptionPane.showMessageDialog(null, "Product has already been added.");
275.         else
276.             addFunction(productDTO);
277.     } catch (SQLException e) {
278.     }
279. }
280. public void addFunction(ProductDTO productDTO) {
281.     try {
282.         String query = "INSERT INTO products VALUES(null,?,?,?,?,?)";
283.         preparedStatement = (PreparedStatement) conn.prepareStatement(query);
284.         preparedStatement.setString(1, productDTO.getProdCode());
285.         preparedStatement.setString(2, productDTO.getProdName());
286.         preparedStatement.setDouble(3, productDTO.getCostPrice());
287.         preparedStatement.setDouble(4, productDTO.getSellPrice());
288.         preparedStatement.setString(5, productDTO.getBrand());
289.
290.         String query2 = "INSERT INTO currentstock VALUES(?,?)";
291.         preparedStatement2 = conn.prepareStatement(query2);
292.         preparedStatement2.setString(1, productDTO.getProdCode());
293.         preparedStatement2.setInt(2, productDTO.getQuantity());
294.
295.         preparedStatement.executeUpdate();
296.         preparedStatement2.executeUpdate();

```

```

297.         JOptionPane.showMessageDialog(null, "Product added and ready for sale.");
298.     } catch (SQLException throwables) {
299.     }
300. }
301.
302. // Method to add a new purchase transaction
303. public void addPurchaseDAO(ProductDTO productDTO) {
304.     try {
305.         String query = "INSERT INTO purchaseinfo VALUES(null,?,?,?,?)";
306.         preparedStatement = conn.prepareStatement(query);
307.         preparedStatement.setString(1, productDTO.getSuppCode());
308.         preparedStatement.setString(2, productDTO.getProdCode());
309.         preparedStatement.setString(3, productDTO.getDate());
310.         preparedStatement.setInt(4, productDTO.getQuantity());
311.         preparedStatement.setDouble(5, productDTO.getTotalCost());
312.
313.         preparedStatement.executeUpdate();
314.         JOptionPane.showMessageDialog(null, "Purchase log added.");
315.     } catch (SQLException throwables) {
316.     }
317.
318.     String prodCode = productDTO.getProdCode();
319.     if(checkStock(prodCode)) {
320.         try {
321.             String query = "UPDATE currentstock SET quantity=quantity+? WHERE
productcode=?";
322.             preparedStatement = conn.prepareStatement(query);
323.             preparedStatement.setInt(1, productDTO.getQuantity());
324.             preparedStatement.setString(2, prodCode);
325.
326.             preparedStatement.executeUpdate();
327.         } catch (SQLException throwables) {
328.         }
329.     }
330.     else if (!checkStock(prodCode)) {
331.         try {
332.             String query = "INSERT INTO currentstock VALUES(?,?)";
333.             preparedStatement = (PreparedStatement) conn.prepareStatement(query);
334.             preparedStatement.setString(1, productDTO.getProdCode());
335.             preparedStatement.setInt(2, productDTO.getQuantity());
336.
337.             preparedStatement.executeUpdate();
338.         } catch (SQLException throwables) {
339.         }
340.     }
341.     deleteStock();

```

```

342.     }
343.
344.     // Method to update existing product details
345.     public void editProdDAO(ProductDTO productDTO) {
346.         try {
347.             String query = "UPDATE products SET
productname=?,costprice=?,sellprice=?,brand=? WHERE productcode=?";
348.             preparedStatement = (PreparedStatement) conn.prepareStatement(query);
349.             preparedStatement.setString(1, productDTO.getProdName());
350.             preparedStatement.setDouble(2, productDTO.getCostPrice());
351.             preparedStatement.setDouble(3, productDTO.getSellPrice());
352.             preparedStatement.setString(4, productDTO.getBrand());
353.             preparedStatement.setString(5, productDTO.getProdCode());
354.
355.             String query2 = "UPDATE currentstock SET quantity=? WHERE
productcode=?";
356.             preparedStatement2 = conn.prepareStatement(query2);
357.             preparedStatement2.setInt(1, productDTO.getQuantity());
358.             preparedStatement2.setString(2, productDTO.getProdCode());
359.
360.             preparedStatement.executeUpdate();
361.             preparedStatement2.executeUpdate();
362.             JOptionPane.showMessageDialog(null, "Product details updated.");
363.         } catch (SQLException throwables) {
364.         }
365.     }
366.
367.     // Methods to handle updating of stocks in Inventory upon any transaction made
368.     public void editPurchaseStock(String code, int quantity) {
369.         try {
370.             String query = "SELECT * FROM currentstock WHERE productcode="
+code+ """;
371.             resultSet = statement.executeQuery(query);
372.             if(resultSet.next()) {
373.                 String query2 = "UPDATE currentstock SET quantity=quantity-? WHERE
productcode=?";
374.                 preparedStatement = conn.prepareStatement(query2);
375.                 preparedStatement.setInt(1, quantity);
376.                 preparedStatement.setString(2, code);
377.                 preparedStatement.executeUpdate();
378.             }
379.         } catch (SQLException throwables) {
380.         }
381.     }
382.     public void editSoldStock(String code, int quantity) {
383.         try {

```

```

384.         String query = "SELECT * FROM currentstock WHERE productcode="
+code+ """;
385.         resultSet = statement.executeQuery(query);
386.         if(resultSet.next()) {
387.             String query2 = "UPDATE currentstock SET quantity=quantity+? WHERE
productcode=?";
388.             preparedStatement = conn.prepareStatement(query2);
389.             preparedStatement.setInt(1, quantity);
390.             preparedStatement.setString(2, code);
391.             preparedStatement.executeUpdate();
392.         }
393.     } catch (SQLException throwables) {
394.     }
395. }
396. public void deleteStock() {
397.     try {
398.         String query = "DELETE FROM currentstock WHERE productcode NOT
IN(SELECT productcode FROM purchaseinfo)";
399.         String query2 = "DELETE FROM salesinfo WHERE productcode NOT
IN(SELECT productcode FROM products)";
400.         statement.executeUpdate(query);
401.         statement.executeUpdate(query2);
402.     } catch (SQLException throwables) {
403.     }
404. }
405.
406. // Method to permanently delete a product from inventory
407. public void deleteProductDAO(String code) {
408.     try {
409.         String query = "DELETE FROM products WHERE productcode=?";
410.         preparedStatement = conn.prepareStatement(query);
411.         preparedStatement.setString(1, code);
412.
413.         String query2 = "DELETE FROM currentstock WHERE productcode=?";
414.         preparedStatement2 = conn.prepareStatement(query2);
415.         preparedStatement2.setString(1, code);
416.
417.         preparedStatement.executeUpdate();
418.         preparedStatement2.executeUpdate();
419.
420.         JOptionPane.showMessageDialog(null, "Product has been removed.");
421.     } catch (SQLException e){
422.     }
423.     deleteStock();
424. }
425.

```

```

426.     public void deletePurchaseDAO(int ID){
427.         try {
428.             String query = "DELETE FROM purchaseinfo WHERE purchaseID=?";
429.             preparedStatement = conn.prepareStatement(query);
430.             preparedStatement.setInt(1, ID);
431.             preparedStatement.executeUpdate();
432.
433.             JOptionPane.showMessageDialog(null, "Transaction has been removed.");
434.         } catch (SQLException e){
435.         }
436.         deleteStock();
437.     }
438.
439.     // Products data set retrieval for display
440.     public ResultSet getQueryResult() {
441.         try {
442.             String query = "SELECT productcode,productname,costprice,sellprice,brand
FROM products ORDER BY pid";
443.             resultSet = statement.executeQuery(query);
444.         } catch (SQLException throwables) {
445.             throwables.printStackTrace();
446.         }
447.         return resultSet;
448.     }
449.
450.     // Purchase table data set retrieval
451.     public ResultSet getPurchaseInfo() {
452.         try {
453.             String query = "SELECT
PurchaseID,purchaseinfo.ProductCode,ProductName,Quantity,Totalcost " +
454.             "FROM purchaseinfo INNER JOIN products " +
455.             "ON products.productcode=purchaseinfo.productcode ORDER BY
purchaseid;";
456.             resultSet = statement.executeQuery(query);
457.         } catch (SQLException throwables) {
458.             throwables.printStackTrace();
459.         }
460.         return resultSet;
461.     }
462.
463.     // Stock table data set retrieval
464.     public ResultSet getCurrentStockInfo() {
465.         try {
466.             String query = ""
467.             SELECT currentstock.ProductCode,products.ProductName,
468.             currentstock.Quantity,products.CostPrice,products.SellPrice

```

```

469.         FROM currentstock INNER JOIN products
470.         ON currentstock.productcode=products.productcode;
471.         """";
472.         resultSet = statement.executeQuery(query);
473.     } catch (SQLException throwables) {
474.         throwables.printStackTrace();
475.     }
476.     return resultSet;
477. }
478.
479. // Search method for products
480. public ResultSet getProductSearch(String text) {
481.     try {
482.         String query = "SELECT productcode,productname,costprice,sellprice,brand
FROM products " +
483.             "WHERE productcode LIKE '%" +text+"%' OR productname LIKE
'" +text+"%' OR brand LIKE '%" +text+"%'";
484.         resultSet = statement.executeQuery(query);
485.     } catch (SQLException e) {
486.     }
487.     return resultSet;
488. }
489.
490. public ResultSet getProdFromCode(String text) {
491.     try {
492.         String query = "SELECT productcode,productname,costprice,sellprice,brand
FROM products " +
493.             "WHERE productcode='" +text+ "' LIMIT 1";
494.         resultSet = statement.executeQuery(query);
495.     } catch (SQLException e) {
496.         e.printStackTrace();
497.     }
498.     return resultSet;
499. }
500.
501. // Search method for purchase logs
502. public ResultSet getPurchaseSearch(String text) {
503.     try {
504.         String query = "SELECT
PurchaseID,purchaseinfo.productcode,products.productname,quantity,totalcost " +
505.             "FROM purchaseinfo INNER JOIN products ON
purchaseinfo.productcode=products.productcode " +
506.             "INNER JOIN suppliers ON
purchaseinfo.suppliercode=suppliers.suppliercode" +
507.             "WHERE PurchaseID LIKE '%" +text+"%' OR productcode LIKE
'" +text+"%' OR productname LIKE '%" +text+"%' " +

```

```

508.         "OR suppliers.fullname LIKE '%" + text + "%' OR
        purchaseinfo.suppliercode LIKE '%" + text + "%' " +
509.         "OR date LIKE '%" + text + "%' ORDER BY purchaseid";
510.         resultSet = statement.executeQuery(query);
511.     } catch (SQLException e) {
512.         e.printStackTrace();
513.     }
514.     return resultSet;
515. }
516.
517. public ResultSet getProdName(String code) {
518.     try {
519.         String query = "SELECT productname FROM products WHERE
        productcode=" + code + "";
520.         resultSet = statement.executeQuery(query);
521.     } catch (SQLException throwables) {
522.         throwables.printStackTrace();
523.     }
524.     return resultSet;
525. }
526.
527. public String getSuppName(int ID) {
528.     String name = null;
529.     try {
530.         String query = "SELECT fullname FROM suppliers " +
531.         "INNER JOIN purchaseinfo ON
        suppliers.suppliercode=purchaseinfo.suppliercode " +
532.         "WHERE purchaseid=" + ID + "";
533.         resultSet = statement.executeQuery(query);
534.         if (resultSet.next())
535.             name = resultSet.getString("fullname");
536.     } catch (SQLException throwables) {
537.         throwables.printStackTrace();
538.     }
539.     return name;
540. }
541.
542. public String getCustName(int ID) {
543.     String name = null;
544.     try {
545.         String query = "SELECT fullname FROM customers " +
546.         "INNER JOIN salesinfo ON
        customers.customercode=salesinfo.customercode " +
547.         "WHERE salesid=" + ID + "";
548.         resultSet = statement.executeQuery(query);
549.         if (resultSet.next())

```

```

550.         name = resultSet.getString("fullname");
551.     } catch (SQLException throwables) {
552.         throwables.printStackTrace();
553.     }
554.     return name;
555. }
556.
557. public String getPurchaseDate(int ID) {
558.     String date = null;
559.     try {
560.         String query = "SELECT date FROM purchaseinfo WHERE purchaseid="
+ID+ """;
561.         resultSet = statement.executeQuery(query);
562.         if (resultSet.next())
563.             date = resultSet.getString("date");
564.     } catch (SQLException throwables) {
565.         throwables.printStackTrace();
566.     }
567.     return date;
568. }
569. public String getSaleDate(int ID) {
570.     String date = null;
571.     try {
572.         String query = "SELECT date FROM salesinfo WHERE salesid=" +ID+ """;
573.         resultSet = statement.executeQuery(query);
574.         if (resultSet.next())
575.             date = resultSet.getString("date");
576.     } catch (SQLException throwables) {
577.         throwables.printStackTrace();
578.     }
579.     return date;
580. }
581.
582.
583. // Method to display product-related data set in tabular form
584. public DefaultTableModel buildTableModel(ResultSet resultSet) throws
SQLException {
585.     ResultSetMetaData metaData = resultSet.getMetaData();
586.     Vector<String> columnNames = new Vector<String>();
587.     int colCount = metaData.getColumnCount();
588.
589.     for (int col=1; col <= colCount; col++){
590.
591.         columnNames.add(metaData.getColumnName(col).toUpperCase(Locale.ROOT));
592.     }

```



```

593.     Vector<Vector<Object>> data = new Vector<Vector<Object>>();
594.     while (resultSet.next()) {
595.         Vector<Object> vector = new Vector<Object>();
596.         for (int col=1; col<=colCount; col++) {
597.             vector.add(resultSet.getObject(col));
598.         }
599.         data.add(vector);
600.     }
601.     return new DefaultTableModel(data, columnNames);
602. }
603. }

```

### Login Page

```

1.  import java.awt.*;
2.  import javax.swing.*;
3.
4.  public class LoginPage extends javax.swing.JFrame {
5.      public LoginPage() {
6.          initComponents();
7.      }
8.
9.      private void initComponents() {
10.
11.         jLabel1 = new javax.swing.JLabel();
12.         jLabel2 = new javax.swing.JLabel();
13.         userText = new javax.swing.JTextField();
14.         passText = new javax.swing.JPasswordField();
15.         jLabel3 = new javax.swing.JLabel();
16.         loginButton = new javax.swing.JButton();
17.
18.         setDefaultCloseOperation(javax.swing.WindowConstants.DISPOSE_ON_CLOSE);
19.         setBounds(new java.awt.Rectangle(400, 100, 0, 0));
20.         setName("loginFrame");
21.
22.         jLabel1.setFont(new java.awt.Font("MV Boli", 0, 14));
23.         jLabel1.setText("Username:");
24.
25.         jLabel2.setFont(new java.awt.Font("MV Boli", 0, 14));
26.         jLabel2.setText("Password:");
27.
28.         jLabel3.setFont(new java.awt.Font("MV Boli", Font.BOLD, 20));
29.         jLabel3.setForeground(Color.BLACK);
30.         jLabel3.setHorizontalAlignment(javax.swing.SwingConstants.CENTER);
31.         jLabel3.setText("MANAGE INVENTORY");
32.
33.         loginButton.setText("LOGIN");

```

```

34.     loginButton.setBackground(new Color(0X05386B));
35.     loginButton.setForeground(new Color(0XEDF5E1));
36.     loginButton.setFocusable(false);
37.     loginButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
38.     loginButton.addActionListener(new java.awt.event.ActionListener() {
39.         public void actionPerformed(java.awt.event.ActionEvent evt) {
40.             loginButtonActionPerformed(evt);
41.         }
42.     });
43.
44.     javax.swing.GroupLayout layout = new
        javax.swing.GroupLayout(getContentPane());
45.     getContentPane().setLayout(layout);
46.     getContentPane().setBackground(new Color(0X8EE4AF));
47.     layout.setHorizontalGroup(
48.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
49.             .addGroup(layout.createSequentialGroup()
50.                 .addGap(47, 47, 47)
51.                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
                    false)
52.                     .addGroup(layout.createSequentialGroup()
53.                         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
74, javax.swing.GroupLayout.PREFERRED_SIZE)
54.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
55.                         .addComponent(userText, javax.swing.GroupLayout.DEFAULT_SIZE,
204, Short.MAX_VALUE))
56.                     .addGroup(layout.createSequentialGroup()
57.                         .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
74, javax.swing.GroupLayout.PREFERRED_SIZE)
58.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
59.                         .addComponent(passText)))
60.                 .addGap(72, 72, 72))
61.             .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
                false)
62.                 .addGroup(layout.createSequentialGroup()
63.                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
64.                         .addGroup(layout.createSequentialGroup()
65.                             .addComponent(loginButton,
                                javax.swing.GroupLayout.PREFERRED_SIZE, 100,
                                javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

66.         .addGap(121, 121, 121))
67.         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
        layout.createSequentialGroup()
68.         .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
        284, javax.swing.GroupLayout.PREFERRED_SIZE)
69.         .addGap(47, 47, 47)))
70.     );
71.     layout.setVerticalGroup(
72.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
73.         .addGroup(layout.createSequentialGroup()
74.         .addGap(53, 53, 53)
75.         .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
        javax.swing.GroupLayout.PREFERRED_SIZE)
76.         .addGap(48, 48, 48)
77.         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
        false)
78.         .addComponent(userText, javax.swing.GroupLayout.DEFAULT_SIZE, 31,
        Short.MAX_VALUE)
79.         .addComponent(jLabel1, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
80.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
81.         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
82.         .addComponent(passText, javax.swing.GroupLayout.PREFERRED_SIZE,
        32, javax.swing.GroupLayout.PREFERRED_SIZE)
83.         .addGroup(layout.createSequentialGroup()
84.         .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
        33, javax.swing.GroupLayout.PREFERRED_SIZE)
85.         .addGap(1, 1, 1)))
86.         .addGap(27, 27, 27)
87.         .addComponent(loginButton, javax.swing.GroupLayout.PREFERRED_SIZE,
        37, javax.swing.GroupLayout.PREFERRED_SIZE)
88.         .addContainerGap(80, Short.MAX_VALUE))
89.     );
90.
91.     pack();
92. }
93.
94. private void loginButtonActionPerformed(java.awt.event.ActionEvent evt) {
95.     String username = userText.getText();
96.     String password = passText.getText();
97.
98.     if (new ConnectionFactory().checkLogin(username, password,
        "ADMINISTRATOR")){

```

```

99.         dispose();
100.         new Dashboard(username, "ADMINISTRATOR");
101.     } else {
102.         JOptionPane.showMessageDialog(
103.             null,
104.             "Invalid username or password.");
105.     }
106.
107. }
108.
109.
110. public static void main(String[] args) {
111.     new LoginPage().setVisible(true);
112. }
113.
114. javax.swing.JLabel jLabel1;
115. javax.swing.JLabel jLabel2;
116. javax.swing.JLabel jLabel3;
117. javax.swing.JButton loginButton;
118. javax.swing.JPasswordField passText;
119. javax.swing.JTextField userText;
120. }

```

## **DASHBOARD**

```

1. import java.awt.CardLayout;
2. import java.awt.Color;
3. import java.awt.event.WindowAdapter;
4. import java.awt.event.WindowEvent;
5.
6. public class Dashboard extends javax.swing.JFrame {
7.
8.     CardLayout layout;
9.
10.     public Dashboard(String username, String userType) {
11.         initComponents();
12.         navPanel.setVisible(true);
13.         menuPanel.setVisible(true);
14.         layout = new CardLayout();
15.
16.         displayPanel.setLayout(layout);
17.         displayPanel.add("Home", new HomePage());
18.         displayPanel.add("Customers", new CustomerPage());
19.         displayPanel.add("Products", new ProductPage(username, this));
20.         displayPanel.add("Current Stock", new CurrentStockPage(username));
21.         displayPanel.add("Sales", new PurchasePage(this));
22.

```

```
23.     this.addWindowListener(new WindowAdapter() {
24.         @Override
25.         public void windowClosing(WindowEvent e) {
26.             super.windowClosing(e);
27.         }
28.     });
29.
30.     setTitle("Inventory Manager");
31.     setVisible(true);
32. }
33.
34. public void addHomePage(){
35.     layout.show(displayPanel, "Home");
36. }
37. public void addCustPage() {
38.     layout.show(displayPanel, "Customers");
39. }
40. public void addProdPage() {
41.     layout.show(displayPanel, "Products");
42. }
43. public void addStockPage() {
44.     layout.show(displayPanel, "Current Stock");
45. }
46. public void addPurchasePage() {
47.     layout.show(displayPanel, "Sales");
48. }
49.
50. private void initComponents() {
51.
52.     mainPanel = new javax.swing.JPanel();
53.     menuPanel = new javax.swing.JPanel();
54.     navPanel = new javax.swing.JPanel();
55.     prodButton = new javax.swing.JButton();
56.     stockButton = new javax.swing.JButton();
57.     custButton = new javax.swing.JButton();
58.     purchaseButton = new javax.swing.JButton();
59.     displayPanel = new javax.swing.JPanel();
60.     userPanel = new javax.swing.JPanel();
61.     jMenuBar1 = new javax.swing.JMenuBar();
62.
63.     mainPanel.setBackground(new Color(0X5CDB95));
64.     menuPanel.setBackground(new Color(0X5CDB95));
65.     navPanel.setBackground(new Color(0X8EE4AF));
66.     userPanel.setBackground(new Color(0X5CDB95));
67.
68.     setDefaultCloseOperation(javax.swing.WindowConstants.EXIT_ON_CLOSE);
```

```

69.     setBounds(new java.awt.Rectangle(200, 100, 0, 0));
70.
71.     menuPanel.setPreferredSize(new java.awt.Dimension(120, 26));
72.
73.     javax.swing.GroupLayout menuPanelLayout = new
    javax.swing.GroupLayout(menuPanel);
74.     menuPanel.setLayout(menuPanelLayout);
75.     menuPanelLayout.setHorizontalGroup(
76.
    menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
77.         .addGroup(
78.             .addGap(0, 125, Short.MAX_VALUE)
79.         );
80.     menuPanelLayout.setVerticalGroup(
81.
    menuPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
82.         .addGroup(
83.             .addGap(0, 50, Short.MAX_VALUE)
84.         );
85.     navPanel.setBorder(new
    javax.swing.border.SoftBevelBorder(javax.swing.border.BevelBorder.RAISED));
86.     prodButton.setText("Products");
87.     prodButton.setBackground(new Color(0X05386B));
88.     prodButton.setForeground(new Color(0XEDF5E1));
89.     prodButton.setFocusable(false);
90.     prodButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
91.     prodButton.addActionListener(new java.awt.event.ActionListener() {
92.         public void actionPerformed(java.awt.event.ActionEvent evt) {
93.             prodButtonActionPerformed(evt);
94.         }
95.     });
96.
97.     stockButton.setText("Current Stock");
98.     stockButton.setBackground(new Color(0X05386B));
99.     stockButton.setForeground(new Color(0XEDF5E1));
100.    stockButton.setFocusable(false);
101.    stockButton.setCursor(new
    java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
102.    stockButton.addActionListener(new java.awt.event.ActionListener() {
103.        public void actionPerformed(java.awt.event.ActionEvent evt) {
104.            stockButtonActionPerformed(evt);
105.        }
106.    });
107.
108.    custButton.setText("Customers");
109.    custButton.setBackground(new Color(0X05386B));

```

```

110.         custButton.setForeground(new Color(0XEDF5E1));
111.         custButton.setFocusable(false);
112.         custButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
113.         custButton.addActionListener(new java.awt.event.ActionListener() {
114.             public void actionPerformed(java.awt.event.ActionEvent evt) {
115.                 custButtonActionPerformed(evt);
116.             }
117.         });
118.
119.         purchaseButton.setText("Sales");
120.         purchaseButton.setBackground(new Color(0X05386B));
121.         purchaseButton.setForeground(new Color(0XEDF5E1));
122.         purchaseButton.setFocusable(false);
123.         purchaseButton.setCursor(new
java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
124.         purchaseButton.addActionListener(new java.awt.event.ActionListener() {
125.             public void actionPerformed(java.awt.event.ActionEvent evt) {
126.                 purchaseButtonActionPerformed(evt);
127.             }
128.         });
129.
130.         javax.swing.GroupLayout navPanelLayout = new
javax.swing.GroupLayout(navPanel);
131.         navPanel.setLayout(navPanelLayout);
132.         navPanelLayout.setHorizontalGroup(
133.
navPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
134.             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
navPanelLayout.createSequentialGroup()
135.                 .addGap(
136.
navPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
137.                     .addComponent(purchaseButton,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
138.                     .addComponent(prodButton,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
Short.MAX_VALUE)
139.                     .addComponent(stockButton,
javax.swing.GroupLayout.Alignment.LEADING,
javax.swing.GroupLayout.DEFAULT_SIZE, 107, Short.MAX_VALUE)
140.                     .addComponent(custButton,
javax.swing.GroupLayout.Alignment.LEADING,

```

```

    javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
141.        .addContainerGap())
142.    );
143.    navPanelLayout.setVerticalGroup(
144.
    navPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
145.        .addGroup(navPanelLayout.createSequentialGroup()
146.            .addGap(44, 44, 44)
147.            .addComponent(prodButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 35,
    javax.swing.GroupLayout.PREFERRED_SIZE)
148.            .addGap(43, 43, 43)
149.            .addComponent(stockButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 35,
    javax.swing.GroupLayout.PREFERRED_SIZE)
150.            .addGap(45, 45, 45)
151.            .addComponent(custButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 35,
    javax.swing.GroupLayout.PREFERRED_SIZE)
152.            .addGap(44, 44, 44)
153.            .addComponent(purchaseButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 35,
    javax.swing.GroupLayout.PREFERRED_SIZE)
154.            .addContainerGap(45, Short.MAX_VALUE))
155.    );
156.
157.    displayPanel.setLayout(new java.awt.CardLayout());
158.
159.    javax.swing.GroupLayout userPanelLayout = new
    javax.swing.GroupLayout(userPanel);
160.    userPanel.setLayout(userPanelLayout);
161.    userPanelLayout.setHorizontalGroup(
162.
    userPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
163.        .addGap(0, 780, Short.MAX_VALUE)
164.    );
165.    userPanelLayout.setVerticalGroup(
166.
    userPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
167.        .addGap(0, 38, Short.MAX_VALUE)
168.    );
169.
170.    javax.swing.GroupLayout mainPanelLayout = new
    javax.swing.GroupLayout(mainPanel);
171.    mainPanel.setLayout(mainPanelLayout);

```



```
172.         mainPanelLayout.setHorizontalGroup(
173.
174.         mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
175.         .addGroup(mainPanelLayout.createSequentialGroup())
176.         .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
177.         .addComponent(navPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
178.         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
179.         .addComponent(menuPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
180.         125, Short.MAX_VALUE))
181.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
182.         .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
183.         .addComponent(displayPanel,
184.         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
185.         Short.MAX_VALUE)
186.         .addComponent(userPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
187.         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)))
188.         );
189.         mainPanelLayout.setVerticalGroup(
190.         mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
191.         .addGroup(mainPanelLayout.createSequentialGroup())
192.         .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
193.         .addComponent(userPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
194.         javax.swing.GroupLayout.DEFAULT_SIZE, javax.swing.GroupLayout.PREFERRED_SIZE)
195.         .addComponent(menuPanel,
196.         javax.swing.GroupLayout.PREFERRED_SIZE, 50,
197.         javax.swing.GroupLayout.PREFERRED_SIZE))
198.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
199.         .addGroup(mainPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
200.         .addComponent(displayPanel,
201.         javax.swing.GroupLayout.PREFERRED_SIZE, 495,
202.         javax.swing.GroupLayout.PREFERRED_SIZE)
203.         .addComponent(navPanel,
204.         javax.swing.GroupLayout.PREFERRED_SIZE, javax.swing.GroupLayout.DEFAULT_SIZE,
205.         javax.swing.GroupLayout.PREFERRED_SIZE))
```

```

193.         .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE))
194.     );
195.
196.     setJMenuBar(jMenuBar1);
197.
198.     javax.swing.GroupLayout layout = new
    javax.swing.GroupLayout(getContentPane());
199.     getContentPane().setLayout(layout);
200.     layout.setHorizontalGroup(
201.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
202.         .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
203.     );
204.     layout.setVerticalGroup(
205.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
206.         .addComponent(mainPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
207.     );
208.
209.     pack();
210. }
211.
212. private void custButtonActionPerformed(java.awt.event.ActionEvent evt) {
213.     addCustPage();
214. }
215.
216. private void stockButtonActionPerformed(java.awt.event.ActionEvent evt) {
217.     addStockPage();
218. }
219.
220. private void prodButtonActionPerformed(java.awt.event.ActionEvent evt) {
221.     addProdPage();
222. }
223.
224. private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {
225.     addPurchasePage();
226. }
227.
228.
229. javax.swing.JButton custButton;
230. javax.swing.JPanel displayPanel;
231. javax.swing.JMenuBar jMenuBar1;
232. javax.swing.JPanel mainPanel;
233. javax.swing.JPanel menuPanel;
234. javax.swing.JPanel navPanel;

```

```

235.     javax.swing.JButton prodButton;
236.     javax.swing.JButton purchaseButton;
237.     javax.swing.JButton stockButton;
238.     javax.swing.JPanel userPanel;
239.
240.     }

```

## HOME PAGE

```

1.  import java.awt.*;
2.
3.  public class HomePage extends javax.swing.JPanel {
4.
5.      public HomePage() {
6.          initComponents();
7.      }
8.
9.      private void initComponents() {
10.
11.          setBackground(new Color(0X5CDB95));
12.
13.          welcomeLabel = new javax.swing.JLabel();
14.          welcomeLabel1 = new javax.swing.JLabel();
15.
16.          welcomeLabel.setFont(new java.awt.Font("MV Boli", Font.BOLD, 45));
17.          welcomeLabel.setText("Welcome");
18.
19.          welcomeLabel1.setFont(new java.awt.Font("MV Boli", Font.BOLD, 45));
20.          welcomeLabel1.setText("To Your Dashboard");
21.
22.          javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
23.          this.setLayout(layout);
24.          layout.setHorizontalGroup(
25.              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
26.                  .addGroup(layout.createSequentialGroup()
27.                      .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
28.                          .addGroup(layout.createSequentialGroup()
29.                              .addGap(170, 170, 170)
30.                              .addComponent(welcomeLabel))
31.                          .addGroup(layout.createSequentialGroup()
32.                              .addGap(71, 71, 71)
33.                              .addComponent(welcomeLabel1)))
34.                      .addGap(83, Short.MAX_VALUE))
35.              );
36.          layout.setVerticalGroup(
37.              layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

38.         .addGroup(layout.createSequentialGroup())
39.         .addGap(96, 96, 96)
40.         .addComponent(welcomeLabel,
    javax.swing.GroupLayout.PREFERRED_SIZE, 48,
    javax.swing.GroupLayout.PREFERRED_SIZE)
41.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
42.         .addComponent(welcomeLabel1,
    javax.swing.GroupLayout.PREFERRED_SIZE, 48,
    javax.swing.GroupLayout.PREFERRED_SIZE)
43.         .addContainerGap(169, Short.MAX_VALUE))
44.     );
45. }
46. javax.swing.JLabel welcomeLabel;
47. javax.swing.JLabel welcomeLabel1;
48. }

```

## **ProductPage**

```

1. import java.awt.Color;
2. import javax.swing.*;
3. import java.sql.SQLException;
4.
5. public class ProductPage extends javax.swing.JPanel {
6.
7.
8.     ProductDTO productDTO;
9.     String username = null;
10.    String supplier = null;
11.    int userID;
12.    Dashboard dashboard;
13.
14.
15.    public ProductPage() {
16.    }
17.
18.    public ProductPage(String username, Dashboard dashboard){
19.        initComponents();
20.        this.username = username;
21.        this.dashboard = dashboard;
22.        loadDataSet();
23.    }
24.
25.    private void initComponents() {
26.        jLabel1 = new javax.swing.JLabel();
27.        jSeparator1 = new javax.swing.JSeparator();
28.        entryPanel = new javax.swing.JPanel();

```

```
29.    jLabel2 = new javax.swing.JLabel();
30.    jLabel3 = new javax.swing.JLabel();
31.    jLabel5 = new javax.swing.JLabel();
32.    jLabel6 = new javax.swing.JLabel();
33.    jLabel7 = new javax.swing.JLabel();
34.    jLabel8 = new javax.swing.JLabel();
35.    codeText = new javax.swing.JTextField();
36.    nameText = new javax.swing.JTextField();
37.    quantityText = new javax.swing.JTextField();
38.    costText = new javax.swing.JTextField();
39.    sellText = new javax.swing.JTextField();
40.    brandText = new javax.swing.JTextField();
41.    addButton = new javax.swing.JButton();
42.    jScrollPane1 = new javax.swing.JScrollPane();
43.    productTable = new javax.swing.JTable();
44.
45.    setBackground(new Color(0X8EE4AF));
46.    jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
47.    jLabel1.setText("PRODUCTS");
48.
49.    entryPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter Product
Details"));
50.
51.    jLabel2.setText("Product Code:");
52.
53.    jLabel3.setText("Product Name:");
54.
55.    jLabel5.setText("Quantity:");
56.
57.    jLabel6.setText("Cost Price:");
58.
59.    jLabel7.setText("Selling Price:");
60.
61.    jLabel8.setText("Brand:");
62.
63.    addButton.setText("Add");
64.    addButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
65.    addButton.setBackground(new Color(0X05386B));
66.    addButton.setForeground(new Color(0XEDF5E1));
67.    addButton.setFocusable(false);
68.    addButton.addActionListener(new java.awt.event.ActionListener() {
69.        public void actionPerformed(java.awt.event.ActionEvent evt) {
70.            addButtonActionPerformed(evt);
71.        }
72.    });
73.
```

```

74.     javax.swing.GroupLayout entryPanelLayout = new
        javax.swing.GroupLayout(entryPanel);
75.     entryPanel.setLayout(entryPanelLayout);
76.     entryPanel.setBackground(new Color(0X8EE4AF));
77.     entryPanelLayout.setHorizontalGroup(
78.
        entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
79.         .addGroup(entryPanelLayout.createSequentialGroup())
80.         .addContainerGap()
81.
        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.L
        EADING)
82.            .addGroup(entryPanelLayout.createSequentialGroup())
83.            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
84. javax.swing.GroupLayout.PREFERRED_SIZE)
84.
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
85.            .addComponent(codeText))
86.        .addGroup(entryPanelLayout.createSequentialGroup())
87.        .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
84. javax.swing.GroupLayout.PREFERRED_SIZE)
88.
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
89.            .addComponent(nameText))
90.        .addGroup(entryPanelLayout.createSequentialGroup())
91.        .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE,
84. javax.swing.GroupLayout.PREFERRED_SIZE)
92.
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
93.            .addComponent(quantityText))
94.        .addGroup(entryPanelLayout.createSequentialGroup())
95.        .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,
84. javax.swing.GroupLayout.PREFERRED_SIZE)
96.
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
97.            .addComponent(costText))
98.        .addGroup(entryPanelLayout.createSequentialGroup())
99.        .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
84. javax.swing.GroupLayout.PREFERRED_SIZE)
100.
        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
101.            .addComponent(sellText))
102.        .addGroup(entryPanelLayout.createSequentialGroup())
103.            .addComponent(jLabel8,
        javax.swing.GroupLayout.PREFERRED_SIZE, 84,
        javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

104.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
105.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
106.                .addComponent(brandText)
107.                .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
108.                    entryPanelLayout.createSequentialGroup()
109.                    .addGap(0, 15, Short.MAX_VALUE)
110.                    .addComponent(addButton,
111.                        javax.swing.GroupLayout.PREFERRED_SIZE, 104,
112.                        javax.swing.GroupLayout.PREFERRED_SIZE)
113.                    .addGap(28, 28, 28))))
114.                .addContainerGap());
115.        entryPanelLayout.setVerticalGroup(
116.            entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
117.                .addGroup(entryPanelLayout.createSequentialGroup()
118.                    .addContainerGap()
119.                    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
120.                        .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,
121.                            26, javax.swing.GroupLayout.PREFERRED_SIZE)
122.                        .addComponent(codeText,
123.                            javax.swing.GroupLayout.PREFERRED_SIZE, 26,
124.                            javax.swing.GroupLayout.PREFERRED_SIZE))
125.                        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
126.                        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
127.                            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
128.                                26, javax.swing.GroupLayout.PREFERRED_SIZE)
129.                            .addComponent(nameText,
130.                                javax.swing.GroupLayout.PREFERRED_SIZE, 26,
131.                                javax.swing.GroupLayout.PREFERRED_SIZE))
132.                            .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
133.                            .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
134.                                .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE,
135.                                    26, javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

127.         .addComponent(quantityText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 26,
    javax.swing.GroupLayout.PREFERRED_SIZE))
128.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
129.
    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
130.         .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
131.         .addComponent(costText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 26,
    javax.swing.GroupLayout.PREFERRED_SIZE))
132.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
133.
    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
134.         .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
135.         .addComponent(sellText, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE))
136.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
137.
    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
138.         .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
139.         .addComponent(brandText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 26,
    javax.swing.GroupLayout.PREFERRED_SIZE))
140.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
141.         .addComponent(addButton, javax.swing.GroupLayout.PREFERRED_SIZE,
    39, javax.swing.GroupLayout.PREFERRED_SIZE)
142.         .addContainerGap(140, Short.MAX_VALUE))
143.     );
144.
145.     productTable.setModel(new javax.swing.table.DefaultTableModel(
146.         new Object [][] {
147.             {null, null, null, null},
148.             {null, null, null, null},
149.             {null, null, null, null},
150.             {null, null, null, null}
151.         },

```



```

152.         new String [] {
153.             "Title 1", "Title 2", "Title 3", "Title 4"
154.         }
155.     ));
156.     productTable.setBackground(new Color(0XEDF5E1));
157.     productTable.setCursor(new
        java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
158.     productTable.addMouseListener(new java.awt.event.MouseAdapter() {
159.         public void mouseClicked(java.awt.event.MouseEvent evt) {
160.             productTableMouseClicked(evt);
161.         }
162.     });
163.     jScrollPane1.setViewportView(productTable);
164.
165.     jSeparator1.setBackground(new Color(0X05386B));
166.     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
167.     this.setLayout(layout);
168.     layout.setHorizontalGroup(
169.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
170.             .addGroup(layout.createSequentialGroup()
171.                 .addContainerGap()
172.
173.                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
174.                     .addGroup(layout.createSequentialGroup()
175.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED, 596,
176.                             Short.MAX_VALUE))
177.                     .addComponent(jSeparator1)
178.                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING,
179.                         false)
180.                         .addGroup(layout.createSequentialGroup()
181.                             .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
182.                             .addComponent(entryPanel,
183.                                 javax.swing.GroupLayout.PREFERRED_SIZE,
184.                                 javax.swing.GroupLayout.DEFAULT_SIZE,
185.                                 javax.swing.GroupLayout.PREFERRED_SIZE)))
186.                         .addContainerGap())

```

```

187.         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
41, javax.swing.GroupLayout.PREFERRED_SIZE)
188.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
189.         .addComponent(jSeparator1,
javax.swing.GroupLayout.PREFERRED_SIZE, 10,
javax.swing.GroupLayout.PREFERRED_SIZE)
190.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
191.         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
false)
192.         .addComponent(entryPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
193.         .addComponent(jScrollPane1,
javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
194.         .addContainerGap())
195.     );
196. }
197.
198. private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
199.     productDTO = new ProductDTO();
200.     if (nameText.getText().equals("") || costText.getText().equals("")
201.         || sellText.getText().equals("") || brandText.getText().equals(""))
202.         JOptionPane.showMessageDialog(null, "Please enter all the required
details.");
203.     else {
204.         productDTO.setProdCode(codeText.getText());
205.         productDTO.setProdName(nameText.getText());
206.         productDTO.setDate("02/01/2020");
207.         productDTO.setQuantity(Integer.parseInt(quantityText.getText()));
208.         productDTO.setCostPrice(Double.parseDouble(costText.getText()));
209.         productDTO.setSellPrice(Double.parseDouble(sellText.getText()));
210.         productDTO.setBrand(brandText.getText());
211.         productDTO.setUserID(userID);
212.
213.         new ProductDAO().addProductDAO(productDTO);
214.         loadDataSet();
215.     }
216. }
217.
218. //static String productName;
219. private void productTableMouseClicked(java.awt.event.MouseEvent evt) {
220.     int row = productTable.getSelectedRow();
221.     int col = productTable.getColumnCount();

```

```

222.
223.     Object[] data = new Object[col];
224.     for (int i=0; i<col; i++)
225.         data[i] = productTable.getValueAt(row, i);
226.
227.         codeText.setText(data[0].toString());
228.         nameText.setText(data[1].toString());
229.         costText.setText(data[2].toString());
230.         sellText.setText(data[3].toString());
231.         brandText.setText(data[4].toString());
232.     }
233.
234.     public void loadDataSet() {
235.         try {
236.             ProductDAO productDAO = new ProductDAO();
237.
238.             productTable.setModel(productDAO.buildTableModel(productDAO.getQueryResult()));
239.         } catch (SQLException throwables) {
240.             throwables.printStackTrace();
241.         }
242.
243.         // Method to display search result in table
244.         public void loadSearchData(String text) {
245.             try {
246.                 ProductDAO productDAO = new ProductDAO();
247.
248.                 productTable.setModel(productDAO.buildTableModel(productDAO.getProductSearch(text)));
249.             } catch (SQLException throwables) {
250.                 throwables.printStackTrace();
251.             }
252.
253.             javax.swing.JButton addButton;
254.             javax.swing.JTextField brandText;
255.             javax.swing.JTextField codeText;
256.             javax.swing.JTextField costText;
257.             javax.swing.JPanel entryPanel;
258.             javax.swing.JLabel jLabel1;
259.             javax.swing.JLabel jLabel2;
260.             javax.swing.JLabel jLabel3;
261.             javax.swing.JLabel jLabel5;
262.             javax.swing.JLabel jLabel6;
263.             javax.swing.JLabel jLabel7;
264.             javax.swing.JLabel jLabel8;

```

```

265.     javax.swing.JScrollPane jScrollPane1;
266.     javax.swing.JSeparator jSeparator1;
267.     javax.swing.JTextField nameText;
268.     javax.swing.JTable productTable;
269.     javax.swing.JTextField quantityText;
270.     javax.swing.JTextField sellText;
271. }

```

### **CurentStockPage**

```

1.  import java.awt.Color;
2.  import java.sql.SQLException;
3.  public class CurrentStockPage extends javax.swing.JPanel {
4.
5.      public CurrentStockPage(String username) {
6.          initComponents();
7.          loadDataSet();
8.      }
9.
10.     private void initComponents() {
11.
12.         jLabel1 = new javax.swing.JLabel();
13.         jSeparator1 = new javax.swing.JSeparator();
14.         jScrollPane1 = new javax.swing.JScrollPane();
15.         stockTable = new javax.swing.JTable();
16.
17.         jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
18.         jLabel1.setText("CURRENT STOCK");
19.         jLabel1.setToolTipText("");
20.
21.         setBackground(new Color(0X8EE4AF));
22.
23.         stockTable.setModel(new javax.swing.table.DefaultTableModel(
24.             new Object [][] {
25.                 {null, null, null, null},
26.                 {null, null, null, null},
27.                 {null, null, null, null},
28.                 {null, null, null, null}
29.             },
30.             new String [] {
31.                 "Title 1", "Title 2", "Title 3", "Title 4"
32.             }
33.         ));
34.         jScrollPane1.setViewportViewView(stockTable);
35.         jSeparator1.setBackground(new Color(0X05386B));
36.         stockTable.setBackground(new Color(0XEDF5E1));
37.

```

```

38.     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
39.     this.setLayout(layout);
40.     this.setBackground(new Color(0X8EE4AF));
41.     layout.setHorizontalGroup(
42.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
43.             .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
44.                 layout.createSequentialGroup()
45.                     .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.TRAILING)
46.                         .addComponent(jSeparator1)
47.                         .addGroup(layout.createSequentialGroup()
48.                             .addComponent(jScrollPane1, javax.swing.GroupLayout.DEFAULT_SIZE,
49.                                 701, Short.MAX_VALUE)
50.                             .addGroup(layout.createSequentialGroup()
51.                                 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
52.                                     165, javax.swing.GroupLayout.PREFERRED_SIZE)
53.                                 .addGap(0, 0, Short.MAX_VALUE)))
54.                             .addGap(0, 0, Short.MAX_VALUE)))
55.                     .addGap(0, 0, Short.MAX_VALUE))
56.             .addGroup(layout.createSequentialGroup()
57.                 .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE, 44,
58.                     javax.swing.GroupLayout.PREFERRED_SIZE)
59.                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
60.                     javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
61.                 .addComponent(jSeparator1, javax.swing.GroupLayout.PREFERRED_SIZE,
62.                     10, javax.swing.GroupLayout.PREFERRED_SIZE)
63.                 .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
64.                 .addComponent(jScrollPane1, javax.swing.GroupLayout.PREFERRED_SIZE,
65.                     330, javax.swing.GroupLayout.PREFERRED_SIZE)
66.                 .addGap(88, 88, Short.MAX_VALUE))
67.             .addGap(0, 0, Short.MAX_VALUE))
68.     );
69. }
70.
71. public void loadDataSet() {
72.     try {
73.         ProductDAO productDAO = new ProductDAO();
74.         stockTable.setModel(productDAO.buildTableModel(productDAO.getCurrentStockInfo())
75.     );
76.     } catch (SQLException e) {
77.     }
78. }

```

```

73.
74. javax.swing.JLabel jLabel1;
75. javax.swing.JScrollPane jScrollPane1;
76. javax.swing.JSeparator jSeparator1;
77. javax.swing.JTable stockTable;
78. }

```

### **CustomerPage**

```

1. import java.awt.Color;
2. import java.sql.SQLException;
3. import javax.swing.JOptionPane;
4.
5. public class CustomerPage extends javax.swing.JPanel {
6.
7.     public CustomerPage() {
8.         initComponents();
9.         loadDataSet();
10.    }
11.
12.    private void initComponents() {
13.
14.        jLabel1 = new javax.swing.JLabel();
15.        jSeparator1 = new javax.swing.JSeparator();
16.        entryPanel = new javax.swing.JPanel();
17.        jLabel2 = new javax.swing.JLabel();
18.        jLabel3 = new javax.swing.JLabel();
19.        jLabel4 = new javax.swing.JLabel();
20.        jLabel5 = new javax.swing.JLabel();
21.        jLabel6 = new javax.swing.JLabel();
22.        jLabel7 = new javax.swing.JLabel();
23.        phoneText = new javax.swing.JTextField();
24.        locationText = new javax.swing.JTextField();
25.        codeText = new javax.swing.JTextField();
26.        nameText = new javax.swing.JTextField();
27.        creditText = new javax.swing.JTextField();
28.        debitText = new javax.swing.JTextField();
29.        addButton = new javax.swing.JButton();
30.        deleteButton = new javax.swing.JButton();
31.        jScrollPane1 = new javax.swing.JScrollPane();
32.        custTable = new javax.swing.JTable();
33.
34.        setBackground(new Color(0X8EE4AF));
35.        jSeparator1.setBackground(new Color(0X05386B));
36.        jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
37.        jLabel1.setText("CUSTOMERS");
38.

```

```

39.     entryPanel.setBorder(javax.swing.BorderFactory.createTitledBorder("Enter
Customer Details"));
40.
41.     jLabel2.setText("Customer Code:");
42.
43.     jLabel3.setText("Full Name:");
44.
45.     jLabel4.setText("Location:");
46.
47.     jLabel5.setText("Contact:");
48.
49.     jLabel6.setText("Debit Amount:");
50.
51.     jLabel7.setText("Credit Amount:");
52.
53.     addButton.setText("Add");
54.     addButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
55.     addButton.setBackground(new Color(0X05386B));
56.     addButton.setForeground(new Color(0XEDF5E1));
57.     addButton.setFocusable(false);
58.     addButton.addActionListener(new java.awt.event.ActionListener() {
59.         public void actionPerformed(java.awt.event.ActionEvent evt) {
60.             addButtonActionPerformed(evt);
61.         }
62.     });
63.
64.     deleteButton.setText("Delete");
65.     deleteButton.setCursor(new java.awt.Cursor(java.awt.Cursor.HAND_CURSOR));
66.     deleteButton.setBackground(new Color(0X05386B));
67.     deleteButton.setForeground(new Color(0XEDF5E1));
68.     deleteButton.setFocusable(false);
69.     deleteButton.addActionListener(new java.awt.event.ActionListener() {
70.         public void actionPerformed(java.awt.event.ActionEvent evt) {
71.             deleteButtonActionPerformed(evt);
72.         }
73.     });
74.
75.     javax.swing.GroupLayout entryPanelLayout = new
javax.swing.GroupLayout(entryPanel);
76.     entryPanel.setLayout(entryPanelLayout);
77.     entryPanel.setBackground(new Color(0X8EE4AF));
78.     entryPanelLayout.setHorizontalGroup(
79.         entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
80.             .addGroup(entryPanelLayout.createSequentialGroup()
81.                 .addContainerGap()

```

```

82.      .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
83.          .addGroup(entryPanelLayout.createSequentialGroup()
84.              .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING, false)
85.                  .addComponent(jLabel2, javax.swing.GroupLayout.DEFAULT_SIZE,
86.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
87.                  .addComponent(jLabel4, javax.swing.GroupLayout.DEFAULT_SIZE,
88.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
89.                  .addComponent(jLabel5, javax.swing.GroupLayout.DEFAULT_SIZE,
90.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
91.                  .addComponent(jLabel6, javax.swing.GroupLayout.DEFAULT_SIZE,
92.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
93.                  .addComponent(jLabel7, javax.swing.GroupLayout.DEFAULT_SIZE,
94.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
95.                  .addComponent(jLabel3, javax.swing.GroupLayout.DEFAULT_SIZE,
96.                      javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
97.              .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
98.          .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
99.              .addComponent(phoneText,
100.                  javax.swing.GroupLayout.DEFAULT_SIZE, 137, Short.MAX_VALUE)
101.              .addComponent(creditText)
102.              .addComponent(debitText)
103.              .addComponent(locationText)
104.              .addComponent(codeText)
105.              .addComponent(nameText)))
106.          .addGroup(entryPanelLayout.createSequentialGroup()
107.              .addGap(41, 41, 41)
108.              .addComponent(addButton)
109.              .addGap(18, 18, 18)
110.              .addComponent(deleteButton)
111.              .addGap(0, 0, Short.MAX_VALUE)))
112.      .addContainerGap()
113.  );
114.  entryPanelLayout.setVerticalGroup(
115.      entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
116.          .addGroup(entryPanelLayout.createSequentialGroup()
117.              .addContainerGap()

```



```
111.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
        BASELINE)  
112.            .addComponent(jLabel2, javax.swing.GroupLayout.PREFERRED_SIZE,  
        29, javax.swing.GroupLayout.PREFERRED_SIZE)  
113.            .addComponent(codeText,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 29,  
        javax.swing.GroupLayout.PREFERRED_SIZE))  
114.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
115.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
        BASELINE)  
116.            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,  
        29, javax.swing.GroupLayout.PREFERRED_SIZE)  
117.            .addComponent(nameText,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 29,  
        javax.swing.GroupLayout.PREFERRED_SIZE))  
118.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
119.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
        BASELINE)  
120.            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,  
        29, javax.swing.GroupLayout.PREFERRED_SIZE)  
121.            .addComponent(locationText,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 29,  
        javax.swing.GroupLayout.PREFERRED_SIZE))  
122.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
123.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
        BASELINE)  
124.            .addComponent(jLabel5, javax.swing.GroupLayout.PREFERRED_SIZE,  
        29, javax.swing.GroupLayout.PREFERRED_SIZE)  
125.            .addComponent(phoneText,  
        javax.swing.GroupLayout.PREFERRED_SIZE, 29,  
        javax.swing.GroupLayout.PREFERRED_SIZE))  
126.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)  
127.        .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.  
        BASELINE)  
128.            .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,  
        29, javax.swing.GroupLayout.PREFERRED_SIZE)
```

```

129.         .addComponent(debitText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 29,
    javax.swing.GroupLayout.PREFERRED_SIZE))
130.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
131.
    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
132.         .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
    29, javax.swing.GroupLayout.PREFERRED_SIZE)
133.         .addComponent(creditText,
    javax.swing.GroupLayout.PREFERRED_SIZE, 29,
    javax.swing.GroupLayout.PREFERRED_SIZE))
134.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.UNRELATED)
135.
    .addGroup(entryPanelLayout.createParallelGroup(javax.swing.GroupLayout.Alignment.
    BASELINE)
136.         .addComponent(addButton)
137.         .addComponent(deleteButton))
138.        .addContainerGap(54, Short.MAX_VALUE))
139.    );
140.
141.    custTable.setModel(new javax.swing.table.DefaultTableModel(
142.        new Object [][] {
143.            {null, null, null, null},
144.            {null, null, null, null},
145.            {null, null, null, null},
146.            {null, null, null, null}
147.        },
148.        new String [] {
149.            "Title 1", "Title 2", "Title 3", "Title 4"
150.        }
151.    ));
152.    custTable.setBackground(new Color(0XEDF5E1));
153.    custTable.addMouseListener(new java.awt.event.MouseAdapter() {
154.        public void mouseClicked(java.awt.event.MouseEvent evt) {
155.            custTableMouseClicked(evt);
156.        }
157.    });
158.    jScrollPane1.setViewportViewView(custTable);
159.
160.    javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
161.    this.setLayout(layout);
162.    layout.setHorizontalGroup(
163.        layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)

```

```

164.         .addGroup(layout.createSequentialGroup())
165.         .addContainerGap()
166.         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
167.         .addGroup(layout.createSequentialGroup()
168.         .addComponent(jLabel1,
169.         javax.swing.GroupLayout.PREFERRED_SIZE, 122,
170.         javax.swing.GroupLayout.PREFERRED_SIZE)
171.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED,
172.         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
173.         .addComponent(jSeparator1)
174.         .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
175.         layout.createSequentialGroup()
176.         .addComponent(jScrollPane1,
177.         javax.swing.GroupLayout.DEFAULT_SIZE, 451, Short.MAX_VALUE)
178.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
179.         .addComponent(entryPanel,
180.         javax.swing.GroupLayout.PREFERRED_SIZE,
181.         javax.swing.GroupLayout.DEFAULT_SIZE,
182.         javax.swing.GroupLayout.PREFERRED_SIZE)))
183.         .addContainerGap())
184.         );
185.         layout.setVerticalGroup(
186.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
187.         .addGroup(layout.createSequentialGroup()
188.         .addContainerGap()
189.         .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
190.         40, javax.swing.GroupLayout.PREFERRED_SIZE)
191.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
192.         .addComponent(jSeparator1,
193.         javax.swing.GroupLayout.PREFERRED_SIZE, 10,
194.         javax.swing.GroupLayout.PREFERRED_SIZE)
195.         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
196.         .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
197.         false)
198.         .addComponent(entryPanel, javax.swing.GroupLayout.DEFAULT_SIZE,
199.         javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE)
200.         .addComponent(jScrollPane1,
201.         javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE))
202.         .addContainerGap(67, Short.MAX_VALUE))
203.         );

```

```

190.     }
191.
192.     private void addButtonActionPerformed(java.awt.event.ActionEvent evt) {
193.         if (codeText.getText().equals("") || nameText.getText().equals("")
194.             || locationText.getText().equals("") || phoneText.getText().equals(""))
195.             JOptionPane.showMessageDialog(this, "Please enter all the required
                details.");
196.         else {
197.             CustomerDTO customerDTO = new CustomerDTO();
198.             customerDTO.setCustCode(codeText.getText());
199.             customerDTO.setFullName(nameText.getText());
200.             customerDTO.setLocation(locationText.getText());
201.             customerDTO.setPhone(phoneText.getText());
202.             new CustomerDAO().addCustomerDAO(customerDTO);
203.             loadDataSet();
204.         }
205.     }
206.
207.     private void deleteButtonActionPerformed(java.awt.event.ActionEvent evt) {
208.         new
            CustomerDAO().deleteCustomerDAO(custTable.getValueAt(custTable.getSelectedRow()
                ,0).toString());
209.         loadDataSet();
210.     }
211.
212.     private void custTableMouseClicked(java.awt.event.MouseEvent evt) {
213.         int row = custTable.getSelectedRow();
214.         int col = custTable.getColumnCount();
215.         Object[] data = new Object[col];
216.
217.         for (int i=0; i<col; i++)
218.             data[i] = custTable.getValueAt(row, i);
219.         codeText.setText((String) data[0]);
220.         nameText.setText((String) data[1]);
221.         locationText.setText((String) data[2]);
222.         phoneText.setText((String) data[3]);
223.     }
224.
225.     public void loadDataSet() {
226.         try {
227.             CustomerDAO customerDAO = new CustomerDAO();
228.             custTable.setModel(customerDAO.buildTableModel(customerDAO.getQueryResult()));
229.         } catch (SQLException e) {
230.         }
231.     }

```

```

232.     public void loadSearchData(String text) {
233.         try {
234.             CustomerDAO customerDAO = new CustomerDAO();
235.             custTable.setModel(customerDAO.buildTableModel(customerDAO.getCustomerSearch(t
ext)));
236.         } catch (SQLException e) {
237.         }
238.     }
239.
240.     javax.swing.JButton addButton;
241.     javax.swing.JTextField codeText;
242.     javax.swing.JTextField creditText;
243.     javax.swing.JTable custTable;
244.     javax.swing.JTextField debitText;
245.     javax.swing.JButton deleteButton;
246.     javax.swing.JPanel entryPanel;
247.     javax.swing.JLabel jLabel1;
248.     javax.swing.JLabel jLabel2;
249.     javax.swing.JLabel jLabel3;
250.     javax.swing.JLabel jLabel4;
251.     javax.swing.JLabel jLabel5;
252.     javax.swing.JLabel jLabel6;
253.     javax.swing.JLabel jLabel7;
254.     javax.swing.JScrollPane jScrollPane1;
255.     javax.swing.JSeparator jSeparator1;
256.     javax.swing.JTextField locationText;
257.     javax.swing.JTextField nameText;
258.     javax.swing.JTextField phoneText;
259. }

```

### **SalesPage**

```

1. import java.awt.Color;
2. import java.sql.ResultSet;
3. import java.sql.SQLException;
4. import javax.swing.JOptionPane;
5.
6. public class PurchasePage extends javax.swing.JPanel {
7.
8.     ProductDTO productDTO;
9.
10.    public PurchasePage(Dashboard dashboard) {
11.        initComponents();
12.        loadDataSet();
13.    }
14.

```

```
15.     private void initComponents() {
16.
17.         jLabel1 = new javax.swing.JLabel();
18.         jSeparator1 = new javax.swing.JSeparator();
19.         jPanel1 = new javax.swing.JPanel();
20.         jLabel3 = new javax.swing.JLabel();
21.         jLabel4 = new javax.swing.JLabel();
22.         jLabel6 = new javax.swing.JLabel();
23.         jLabel7 = new javax.swing.JLabel();
24.         jLabel8 = new javax.swing.JLabel();
25.         codeText = new javax.swing.JTextField();
26.         nameText = new javax.swing.JTextField();
27.         quantityText = new javax.swing.JTextField();
28.         costText = new javax.swing.JTextField();
29.         sellText = new javax.swing.JTextField();
30.         purchaseButton = new javax.swing.JButton();
31.         jScrollPane1 = new javax.swing.JScrollPane();
32.         purchaseTable = new javax.swing.JTable();
33.
34.         setBackground(new Color(0X8EE4AF));
35.         jSeparator1.setBackground(new Color(0X05386B));
36.
37.         jLabel1.setFont(new java.awt.Font("MV Boli", 0, 15));
38.         jLabel1.setText("SALES");
39.
40.         jPanel1.setBorder(javax.swing.BorderFactory.createTitledBorder("Sales Info"));
41.
42.         jLabel3.setText("Product Code:");
43.
44.         jLabel4.setText("Product Name:");
45.
46.         jLabel6.setText("Quantity:");
47.
48.         jLabel7.setText("Cost Price:");
49.
50.         jLabel8.setText("Selling Price:");
51.
52.         codeText.addKeyListener(new java.awt.event.KeyAdapter() {
53.             public void keyReleased(java.awt.event.KeyEvent evt) {
54.                 codeTextKeyReleased(evt);
55.             }
56.         });
57.
58.         purchaseButton.setText("Sales");
59.         purchaseButton.setBackground(new Color(0X05386B));
60.         purchaseButton.setForeground(new Color(0XEDF5E1));
```

```

61.     purchaseButton.setFocusable(false);
62.     purchaseButton.addActionListener(new java.awt.event.ActionListener() {
63.         public void actionPerformed(java.awt.event.ActionEvent evt) {
64.             purchaseButtonActionPerformed(evt);
65.         }
66.     });
67.
68.     javax.swing.GroupLayout jPanel1Layout = new javax.swing.GroupLayout(jPanel1);
69.     jPanel1.setLayout(jPanel1Layout);
70.     jPanel1.setBackground(new Color(0X8EE4AF));
71.     jPanel1Layout.setHorizontalGroup(
72.         jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
73.             .addGroup(jPanel1Layout.createSequentialGroup()
74.                 .addContainerGap()
75.                 .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
76.                     .addGroup(jPanel1Layout.createSequentialGroup()
77.                         .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
102, javax.swing.GroupLayout.PREFERRED_SIZE)
78.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
79.                         .addComponent(codeText, javax.swing.GroupLayout.DEFAULT_SIZE,
173, Short.MAX_VALUE))
80.                     .addGroup(jPanel1Layout.createSequentialGroup()
81.                         .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,
102, javax.swing.GroupLayout.PREFERRED_SIZE)
82.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
83.                         .addComponent(nameText))
84.                     .addGroup(jPanel1Layout.createSequentialGroup()
85.                         .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,
102, javax.swing.GroupLayout.PREFERRED_SIZE)
86.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
87.                         .addComponent(quantityText))
88.                     .addGroup(jPanel1Layout.createSequentialGroup()
89.                         .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
102, javax.swing.GroupLayout.PREFERRED_SIZE)
90.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
91.                         .addComponent(costText))
92.                     .addGroup(jPanel1Layout.createSequentialGroup()
93.                         .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE,
102, javax.swing.GroupLayout.PREFERRED_SIZE)

```

```

94.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
95.            .addComponent(sellText)))
96.        .addContainerGap())
97.        .addGroup(javax.swing.GroupLayout.Alignment.TRAILING,
    jPanel1Layout.createSequentialGroup())
98.        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE)
99.        .addComponent(purchaseButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 113,
    javax.swing.GroupLayout.PREFERRED_SIZE)
100.        .addGap(34, 34, 34))
101.    );
102.    jPanel1Layout.setVerticalGroup(
103.        jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
104.        .addGroup(jPanel1Layout.createSequentialGroup())
105.        .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
    Short.MAX_VALUE)
106.        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
    Seline)
107.            .addComponent(jLabel3, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
108.            .addComponent(codeText,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
109.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
110.        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
    Seline)
111.            .addComponent(jLabel4, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
112.            .addComponent(nameText,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
113.        .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
114.        .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BA
    Seline)
115.            .addComponent(jLabel6, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)

```



```

116.         .addComponent(quantityText,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
117.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
118.
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
119.         .addComponent(jLabel7, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
120.         .addComponent(costText,
    javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
121.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
122.
    .addGroup(jPanel1Layout.createParallelGroup(javax.swing.GroupLayout.Alignment.BASELINE)
123.         .addComponent(jLabel8, javax.swing.GroupLayout.PREFERRED_SIZE,
    26, javax.swing.GroupLayout.PREFERRED_SIZE)
124.         .addComponent(sellText, javax.swing.GroupLayout.PREFERRED_SIZE,
    javax.swing.GroupLayout.DEFAULT_SIZE,
    javax.swing.GroupLayout.PREFERRED_SIZE))
125.
    .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
126.
    .addComponent(purchaseButton,
    javax.swing.GroupLayout.PREFERRED_SIZE, 31,
    javax.swing.GroupLayout.PREFERRED_SIZE)
127.
    .addGap(29, 29, 29))
128.    );
129.
130.    purchaseTable.setModel(new javax.swing.table.DefaultTableModel(
131.        new Object [][] {
132.            {null, null, null, null},
133.            {null, null, null, null},
134.            {null, null, null, null},
135.            {null, null, null, null}
136.        },
137.        new String [] {
138.            "Title 1", "Title 2", "Title 3", "Title 4"
139.        }
140.    ));
141.
142.    purchaseTable.setBackground(new Color(0XEDF5E1));

```

```

143.
144.     purchaseTable.addMouseListener(new java.awt.event.MouseAdapter() {
145.         public void mouseClicked(java.awt.event.MouseEvent evt) {
146.             purchaseTableMouseClicked(evt);
147.         }
148.     });
149.     jScrollPane1.setViewportView(purchaseTable);
150.
151.     javax.swing.GroupLayout layout = new javax.swing.GroupLayout(this);
152.     this.setLayout(layout);
153.     layout.setHorizontalGroup(
154.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
155.             .addComponent(jSeparator1)
156.             .addGroup(layout.createSequentialGroup()
157.                 .addContainerGap()
158.                 .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
159.                     .addGroup(layout.createSequentialGroup()
160.                         .addComponent(jLabel1,
161.                             javax.swing.GroupLayout.PREFERRED_SIZE, 112,
162.                             javax.swing.GroupLayout.PREFERRED_SIZE)
163.                         .addGap(0, 0, Short.MAX_VALUE))
164.                     .addGroup(layout.createSequentialGroup()
165.                         .addComponent(jScrollPane1,
166.                             javax.swing.GroupLayout.DEFAULT_SIZE, 459, Short.MAX_VALUE)
167.                         .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
168.                         .addComponent(jPanel1,
169.                             javax.swing.GroupLayout.PREFERRED_SIZE,
170.                             javax.swing.GroupLayout.DEFAULT_SIZE,
171.                             javax.swing.GroupLayout.PREFERRED_SIZE)))
172.                 .addContainerGap())
173.         );
174.     layout.setVerticalGroup(
175.         layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING)
176.             .addGroup(layout.createSequentialGroup()
177.                 .addGroup(layout.createSequentialGroup()
178.                     .addContainerGap()
179.                     .addComponent(jLabel1, javax.swing.GroupLayout.PREFERRED_SIZE,
180.                         43, javax.swing.GroupLayout.PREFERRED_SIZE)
181.                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)
182.                     .addComponent(jSeparator1,
183.                         javax.swing.GroupLayout.PREFERRED_SIZE, 10,
184.                         javax.swing.GroupLayout.PREFERRED_SIZE)
185.                     .addPreferredGap(javax.swing.LayoutStyle.ComponentPlacement.RELATED)

```

```

176.        .addGroup(layout.createParallelGroup(javax.swing.GroupLayout.Alignment.LEADING,
        false)
177.            .addComponent(jScrollPane1,
        javax.swing.GroupLayout.PREFERRED_SIZE, 0, Short.MAX_VALUE)
178.            .addComponent(jPanel1, javax.swing.GroupLayout.DEFAULT_SIZE,
        javax.swing.GroupLayout.DEFAULT_SIZE, Short.MAX_VALUE))
179.            .addContainerGap(javax.swing.GroupLayout.DEFAULT_SIZE,
        Short.MAX_VALUE))
180.    );
181.    }
182.
183.    private void purchaseTableMouseClicked(java.awt.event.MouseEvent evt) {
184.        int row = purchaseTable.getSelectedRow();
185.        int col = purchaseTable.getColumnCount();
186.
187.        Object[] data = new Object[col];
188.        for (int i=0; i<col; i++)
189.            data[i] = purchaseTable.getValueAt(row, i);
190.
191.        Integer.parseInt(data[3].toString());
192.        data[1].toString();
193.    }
194.
195.    private void purchaseButtonActionPerformed(java.awt.event.ActionEvent evt) {
196.        productDTO = new ProductDTO();
197.        productDTO.setSuppCode("12");
198.        productDTO.setProdCode(codeText.getText());
199.        try {
200.            ResultSet resultSet = new ProductDAO().getProdName(codeText.getText());
201.            if (resultSet.next()) {
202.                productDTO.setDate("02/01/2020");
203.                productDTO.setQuantity(Integer.parseInt(quantityText.getText()));
204.                Double costPrice = Double.parseDouble(costText.getText());
205.                Double totalCost = costPrice * Integer.parseInt(quantityText.getText());
206.                productDTO.setTotalCost(totalCost);
207.
208.                new ProductDAO().addPurchaseDAO(productDTO);
209.                loadDataSet();
210.            } else
211.                JOptionPane.showMessageDialog(null, "Please add this product in the
        \"Products\" section before proceeding.");
212.        } catch (SQLException e) {
213.        }
214.    }
215.

```

```

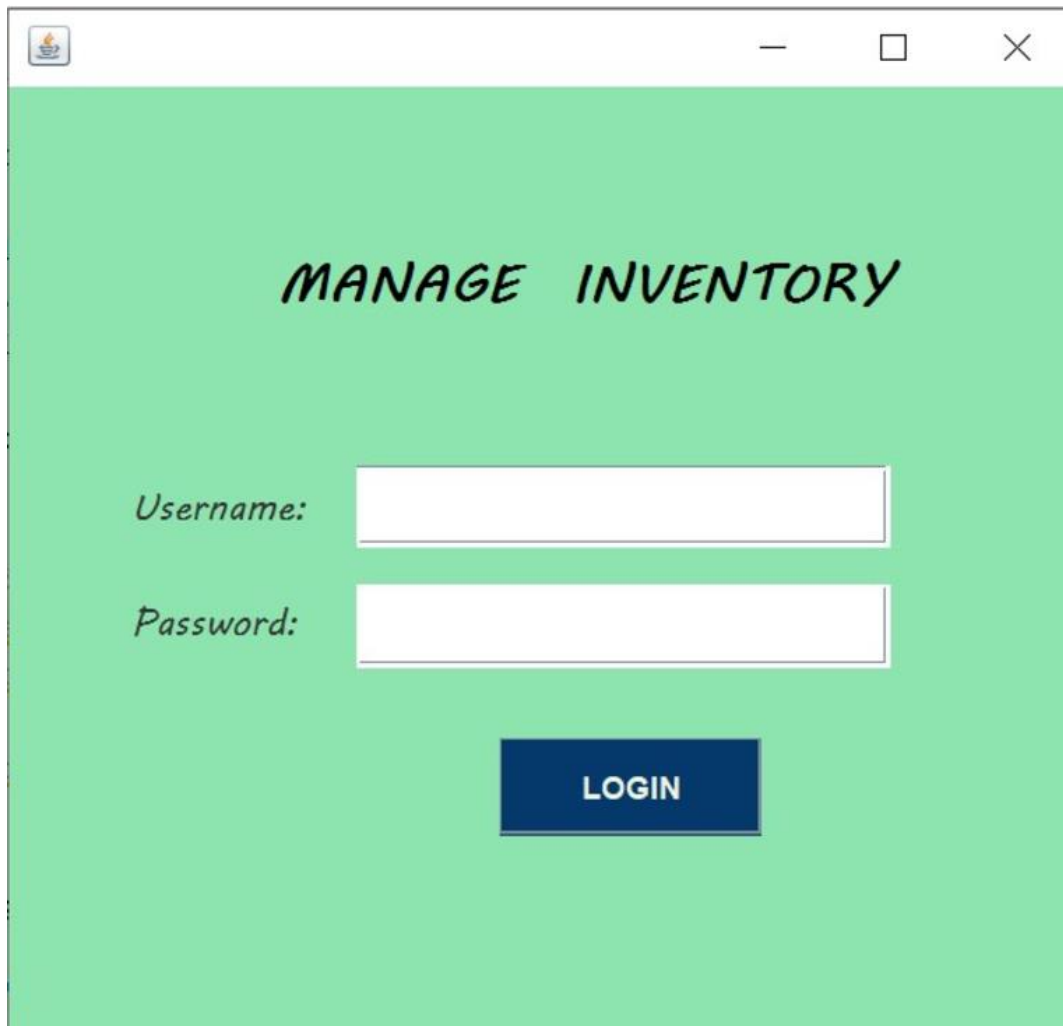
216.     private void codeTextKeyReleased(java.awt.event.KeyEvent evt) {
217.         try {
218.             ResultSet resultSet = new
                ProductDAO().getProdFromCode(codeText.getText());
219.             if (resultSet.next()) {
220.                 nameText.setText(resultSet.getString("productname"));
221.                 costText.setText(String.valueOf(resultSet.getDouble("costprice")));
222.                 sellText.setText(String.valueOf(resultSet.getDouble("sellprice")));
223.             } else {
224.                 nameText.setText("");
225.                 costText.setText("");
226.                 sellText.setText("");
227.             }
228.         } catch (SQLException e) {
229.         }
230.     }
231.
232.     public void loadDataSet() {
233.         try {
234.             ProductDAO productDAO = new ProductDAO();
235.             purchaseTable.setModel(productDAO.buildTableModel(productDAO.getPurchaseInfo()))
                ;
236.         } catch (SQLException throwables) {
237.         }
238.     }
239.
240.     public void loadSearchData(String text) {
241.         try {
242.             ProductDAO productDAO = new ProductDAO();
243.             purchaseTable.setModel(productDAO.buildTableModel(productDAO.getPurchaseSearch(
                text)));
244.         } catch (SQLException e) {
245.         }
246.     }
247.
248.     javax.swing.JTextField codeText;
249.     javax.swing.JTextField costText;
250.     javax.swing.JLabel jLabel1;
251.     javax.swing.JLabel jLabel3;
252.     javax.swing.JLabel jLabel4;
253.     javax.swing.JLabel jLabel6;
254.     javax.swing.JLabel jLabel7;
255.     javax.swing.JLabel jLabel8;
256.     javax.swing.JPanel jPanel1;

```

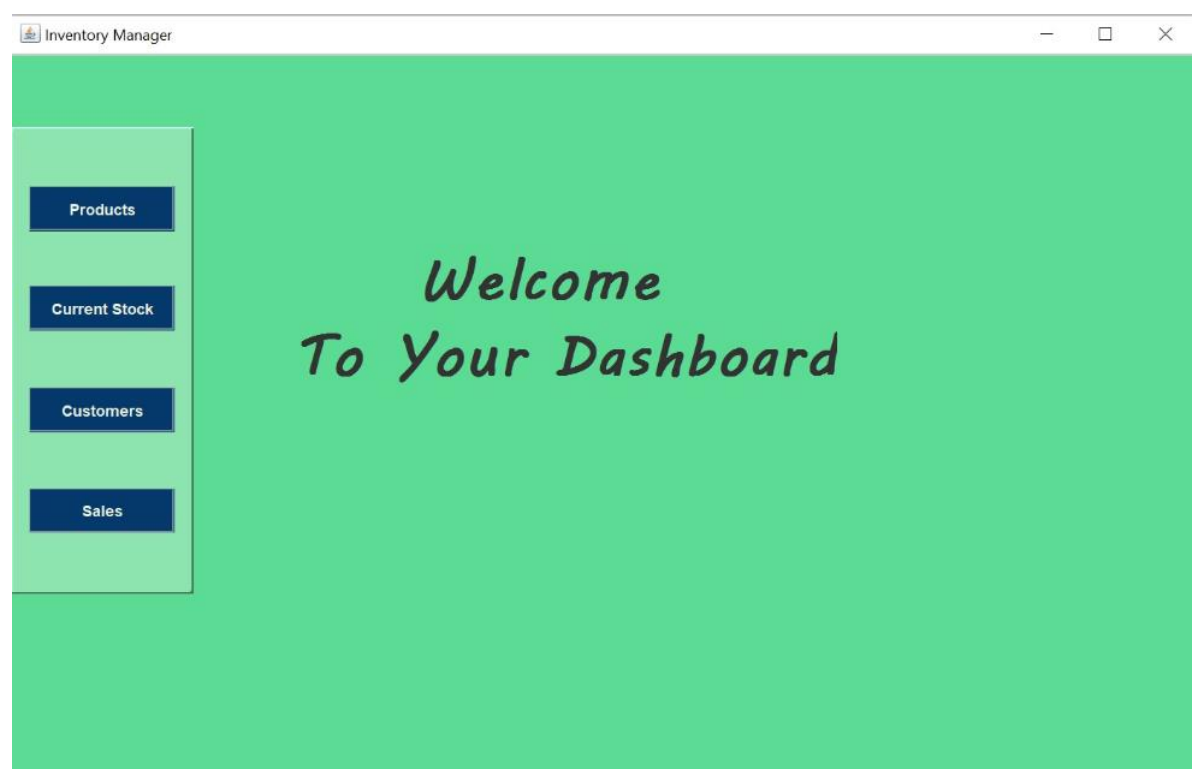
```
257.     javax.swing.JScrollPane jScrollPane1;  
258.     javax.swing.JSeparator jSeparator1;  
259.     javax.swing.JTextField nameText;  
260.     javax.swing.JButton purchaseButton;  
261.     javax.swing.JTable purchaseTable;  
262.     javax.swing.JTextField quantityText;  
263.     javax.swing.JTextField sellText;  
264. }
```

## OUTPUT:

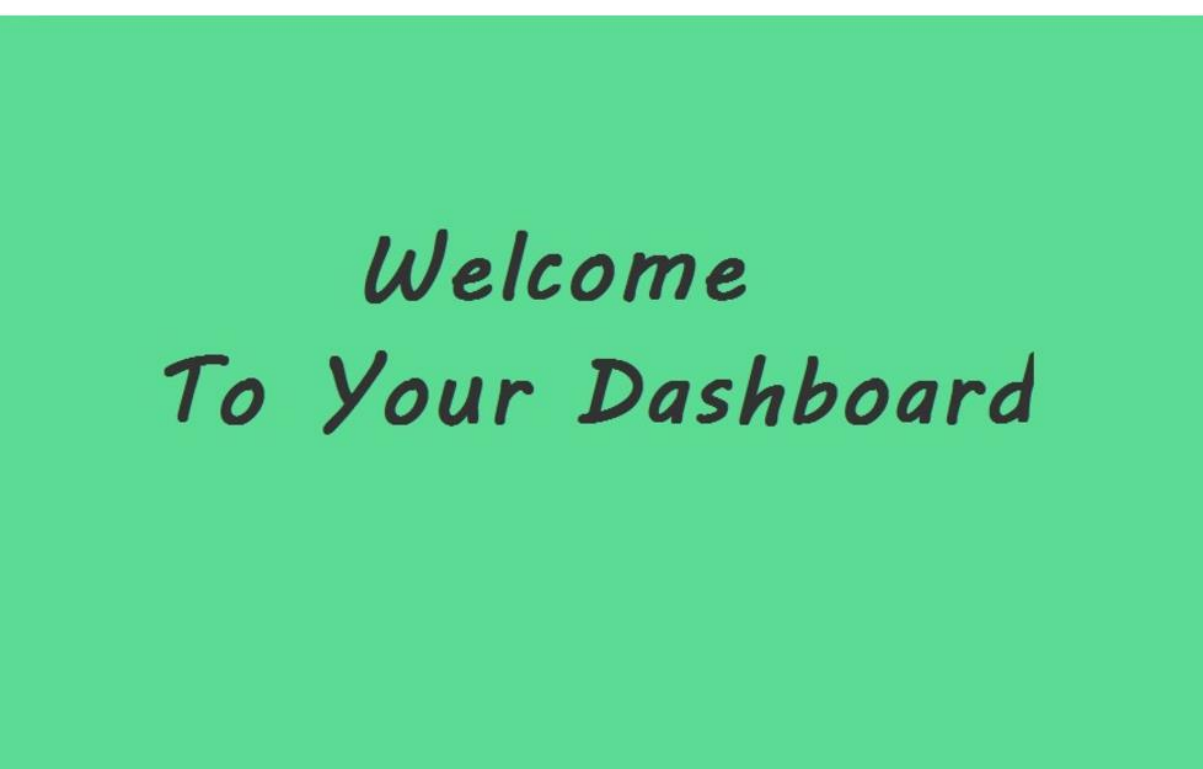
### Login Page



**Dashboard:**



**Home Page:**



Product Page:

PRODUCTS

PRODUCTCO...	PRODUCTNAME	COSTPRICE	SELLPRICE	BRAND
prod1	Laptop	85000.0	90000.0	Dell
prod2	Laptop	70000.0	72000.0	HP
prod3	Mobile	60000.0	64000.0	Apple
prod4	Mobile	50000.0	51000.0	Samsung
prod5	Charger	2000.0	2100.0	Apple
prod6	Mouse	1700.0	1900.0	Dell
prod7	Power Adapter	3000.0	3500.0	Dell
prod8	Smart Watch	15000.0	17000.0	Apple

Enter Product Details

Product Code:

Product Na...

Quantity:

Cost Price:

Selling Price:

Brand:

Add

Current Stock Page:

CURRENT STOCK

PRODUCTCODE	PRODUCTNAME	QUANTITY	COSTPRICE	SELLPRICE
prod1	Laptop	146	85000.0	90000.0
prod2	Laptop	100	70000.0	72000.0
prod3	Mobile	202	60000.0	64000.0
prod4	Mobile	172	50000.0	51000.0
prod5	Charger	500	2000.0	2100.0
prod6	Mouse	500	1700.0	1900.0
prod7	Power Adapter	10	3000.0	3500.0
prod8	Smart Watch	20	15000.0	17000.0

Customer Page:

*CUSTOMERS*

CUSTOMERCODE	FULLNAME	LOCATION	PHONE
vip1	John Seed	New York	9818562354
vip2	Jacob Seed	Texas	9650245489
std1	Ajay Kumar	Mumbai	9236215622
std2	Astha Walla	Chandigarh	8854612478
vip3	Madhu Chitkara	Chandigarh	9826546182

Enter Customer Details

Customer Code:

Full Name:

Location:

Contact:

Debit Amount:

Credit Amount:

AddDelete

Sales Page:

*SALES*

PURCHASEID	PRODUCTC...	PRODUCTN...	QUANTITY	TOTALCOST
1001	prod1	Laptop	10	850000.0
1002	prod6	Mouse	20	34000.0
1003	prod3	Mobile	5	300000.0
1004	prod5	Charger	5	10000.0
1005	prod2	Laptop	2	140000.0
1006	prod4	Mobile	2	100000.0
1009	prod3	Mobile	2	120000.0
1010	prod7	Power Adapter	10	30000.0
1011	prod8	Smart Watch	20	300000.0

Sales Info

Product Code:

Product Name:

Quantity:

Cost Price:

Selling Price:

Add



**RESULT:**

The Inventory Management System successfully automates and streamlines the process of managing stock, providing efficient features for adding, updating, and removing items. It enhances data accuracy, reduces manual errors, and allows real-time inventory tracking, improving overall operational efficiency.