Criterion C- Development

Sr No	Name of Complexity	Purpose	Success Criteria
1	Java Database Connectivity (JDBC)	Allows the system to connect to a centralised MySQL database server for systematic retrieval and storage of data such as customer, order, inventory and staff info.	1,2,3,4,5,6,7,8,9
2	Accessing External Libraries	Using external JAR files such as the rs2xml.jar (to transfer database records into a jTable), and jCalender.jar (to access date chooser for forms), to access pre-written code suiting a specific purpose and speeding up the coding process.	1,2,3,4,5,6,7,8
3	Enabling Data Selection using jComboBox	This would restrict repeated user data entry, thus making data entry more efficient in terms of time and limiting data-entry errors.	3,4,5,6,7,9
4	Real-time database updates	This would give real-time updates of data prevailing in tables such as updating current item stock levels based on newly placed orders, and status of orders if 'pending' or 'complete'. This would allow the client to respond accordingly.	1.3, 2, 3.1, 4, 5.1,7, 8, 9
5	Restricting System Access	Granting different access levels to different users based on their roles (Employee/Manager), and using password protection to maintain data privacy and security	1, 8 ,9
6	Simultaneous updation of tables	This is to keep the data consistent in different tables and prevent errors in terms of data	4, 7

		retrieval during concurrent tasks such as inventory updation while placing an order	
7	Auto-Increment Primary Key fields	Assigning unique I'Ds to each record in different tables to keep them unique and maintain data atomicity. This is done by adding the auto-increment functionality to these I'd fields which increment their value by 1 thus creating a new unique I'd for each new record added.	1,2,7,8
8	Data Validation and Confirmation prompts on data entry/update	This is to ensure that the user makes valid data entry with minimal errors, after reviewing the made changes (For example confirmation prompt in 'change password' form)	1,2,7,8
9	Data Abstraction	Making the system more user-friendly in terms of showing the business significance of each functionality and hiding complex processing and calculations through buttons and GUI	1,2,3,4,5,6,7,8,9
10	Searching/ Sorting of database records	This is to allow users view particular records from database such as viewing items from the inventory within a particular stock range. This would aid in easier retrieval of data to make necessary updates.	3,5,6,9
11	PDF Generator	This would enable users to access the PDF copies of reports/receipts generated by the system, to keep a physical record of/ share these copies to clients or other individuals as per requirement.	3.2, 4, 5.2, 6.3, 7.1, 9.1

1. MySQL Connection:

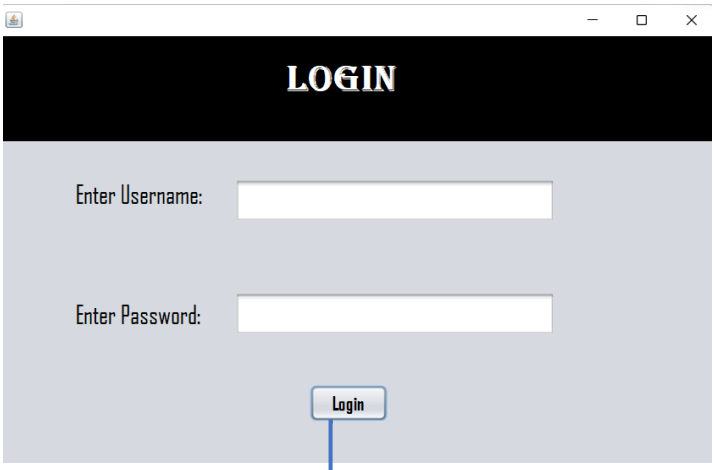
```java
package iaapp;
import java.sql.*;
import javax.swing.*;
public class MySQLConnect {
    Connection conn=null;
    public static Connection ConnectDB(){
        try{ //Connects the database and the program
            Class.forName("com.mysql.cj.jdbc.Driver"); //JDBC driver is used to connect the program and MySQL Database
            Connection conn=DriverManager.getConnection("jdbc:mysql://localhost:3306/ia","root","root");

            return conn;
        }
    catch(ClassNotFoundException | SQLException e){
    JOptionPane.showMessageDialog(null,e);
    return null;
    }
    }
}
```

This ConnectDb() function helps establish connection to the database server. It helps reduce redundancies as this same function can be called in all JFrame Forms to establish database connection

Path to my database server

The above connection to database is obtained through the 'mysql-connection-java' JAR file, which grants me access to several pre-existing libraries and functions such as the .getConnection function which uses the path of my 'IA' database to establish connection.

2. Login Form



Directs user to the system's dashboard. A suitable error message is displayed for incorrect details

Validation to ensure both fields are entered with details and not left blank

```
if (txtUsername.getText().equals("") || txtPassword.getText().equals("")) {
JOptionPane.showMessageDialog(this, "Please fill in all Details!", "Error", JOptionPane.ERROR_MESSAGE);     }
else {
try
{
String sql="select * from Staff where Username=? and password=?";
PreparedStatement pst=conn.prepareStatement(sql);
pst.setString(1,txtUsername.getText());
pst.setString(2,txtPassword.getText());
ResultSet rs=pst.executeQuery();
if(rs.next()){
JOptionPane.showMessageDialog(null,"Logged in successfully..");
String username=txtUsername.getText();
String sql1="select Role from Staff where Username='"+username;
rs=pst.executeQuery();
if(rs.next()){
String role=rs.getString("Role");
if("Manager".equals(role))
{
    DashboardMan dman= new DashboardMan();
dman.setVisible(true);
setVisible(false);
new DashboardMan(username,role).setVisible(true);
      this.setVisible(false);
}
else{
          DashboardEmp demp= new DashboardEmp();
demp.setVisible(true);
setVisible(false);
new DashboardEmp(username,role).setVisible(true);
      this.setVisible(false);
        } }
```

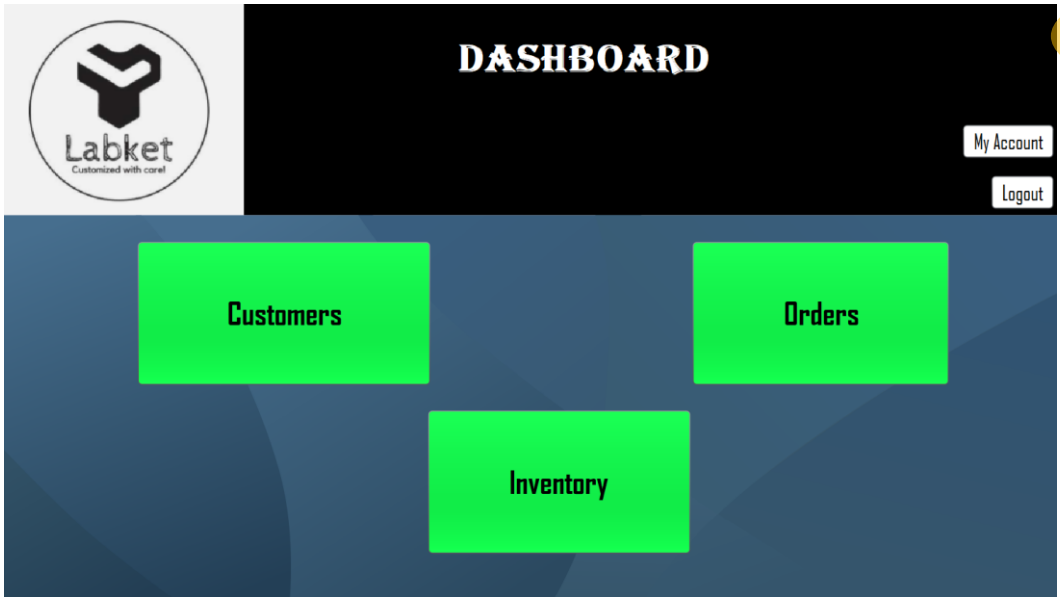
Code to fetch the entered username and password from the 'staff' table

Directs user to the Manager Dashboard if role = "Manager", and to the Employee Dashboard if role = "Employee"

```
    else{
        JOptionPane.showMessageDialog(null,"Incorrect login credentials!");
        txtUsername.setText("");
        txtPassword.setText("");
    }
} catch(Exception e){
    JOptionPane.showMessageDialog(null,e);
}
}
}
```

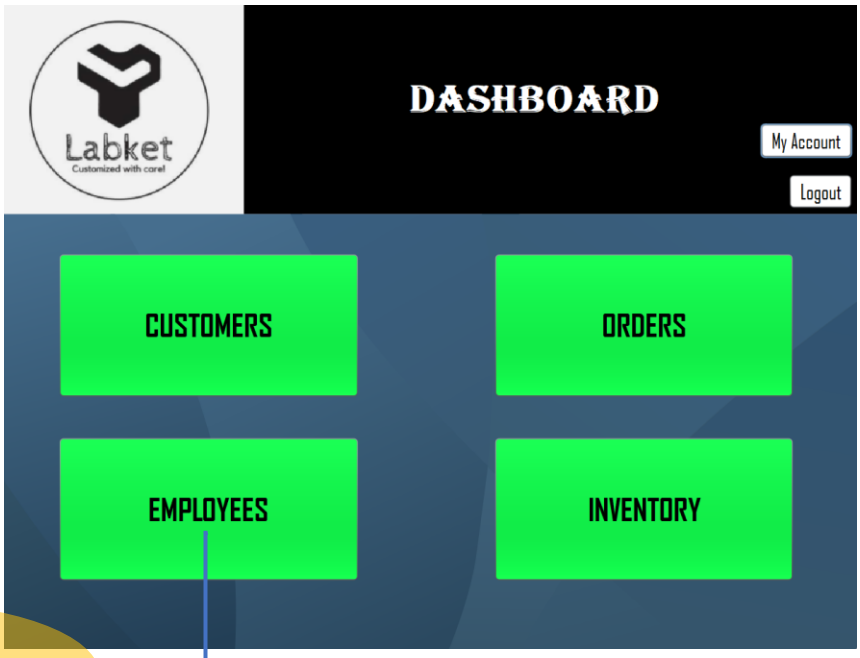
Denies user entry into the system if the Username and Password entered are incorrect

3. Employee Dashboard



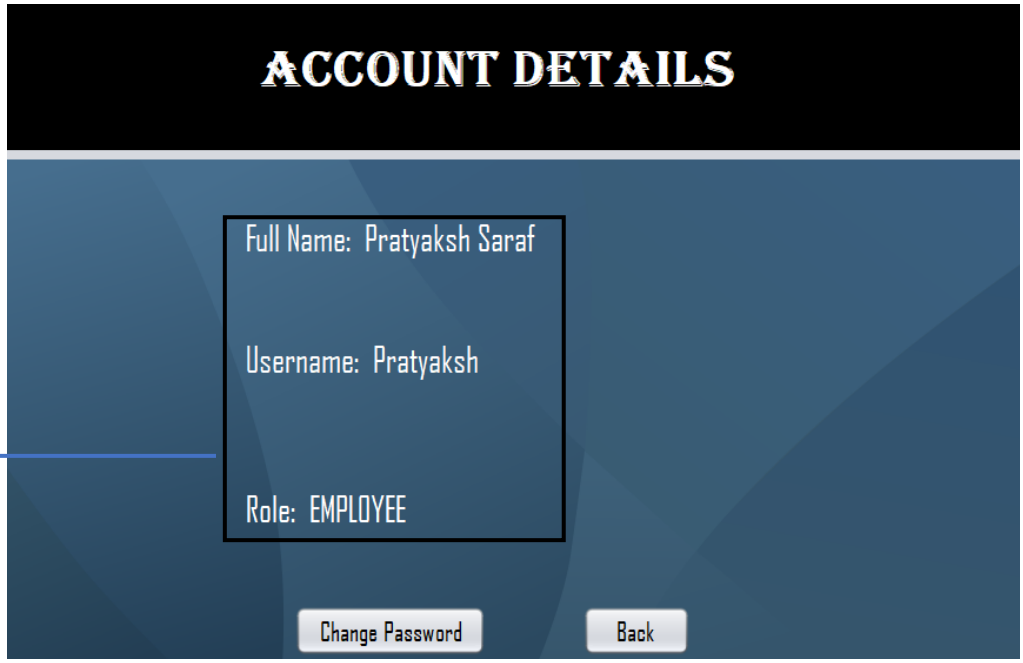
Allows user to view Account details

4. Manager Dashboard



This additional functionally allows Managers to manage the company's employees

5. Staff Details Window



Lists the details of the current logged in user.

```
EmployeeDetails(String username,String role)
{
    initComponents();
    conn=MySQLConnect.ConnectDB();
    Username=username;
    Role=role;
    lblUsername.setText("Username:  "+username);
    try{
        String query="select Staff_Name from staff where Username= '"+username+"'";
        PreparedStatement pst=conn.prepareStatement(query);
        ResultSet rs= (ResultSet) pst.executeQuery();
          if(rs.next()){
              String name=rs.getString("Staff_Name");
              lblName.setText("Full Name:  "+name);
              lblRole.setText("Role:  "+role);
          }
    }
    catch(Exception e){
          JOptionPane.showMessageDialog(null,e);
       }
}
```

Function to display the details of the logged in user

6. Change Password Form

Successfully changes password of the user in the database

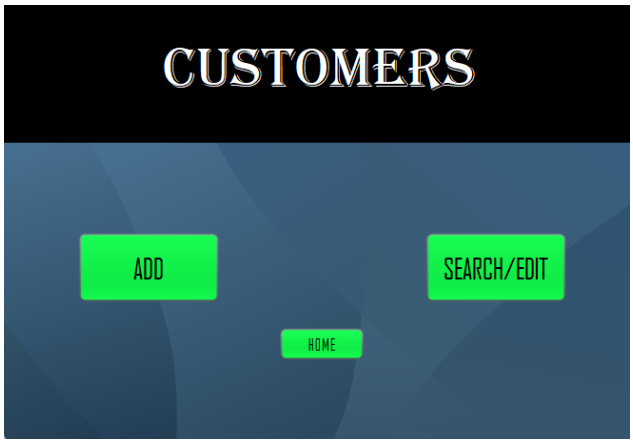
Staff_ID	Staff_Name	Username	Password	Role	Salary
1	Shalabh Agrawal	ShalabhAgra	1234	Manager	150000
3	Pratyaksh Saraf	Pratyaksh	abcd	EMPLOYEE	30000
4	Ashish Bhatt	Ashish	1234	EMPLOYEE	25000
5	ABC	ABC	1234	EMPLOYEE	25000
6	Kunal Saraf	Kunal_1	abcd	EMPLOYEE	25000

Code to update the password in the database

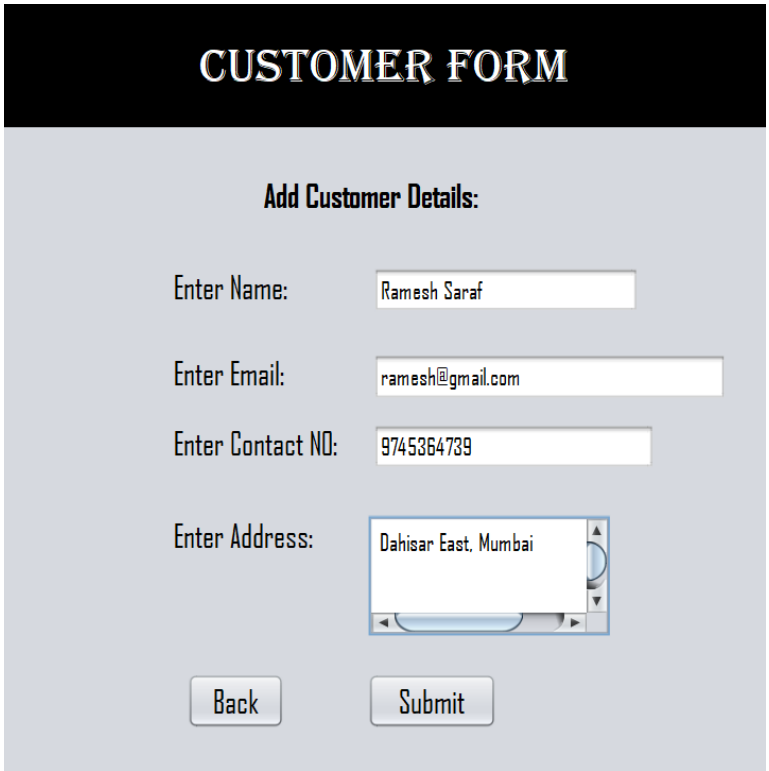
```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int x=JOptionPane.showConfirmDialog(null, "Are you sure you want to make the changes?", "Confirmation", JOptionPane.YES NO OPTION);//Confirmation prompt
    if(x==0)//If the user clicks 'yes' in the confirmation prompt
    {
String CurrentPass=txtCurrentPass.getText();
try{
    String query="select Staff_Name from staff where Username= '"+txtUsername.getText()+"' and Password= '"+CurrentPass+"'";
    System.out.println(query);
    PreparedStatement pst=conn.prepareStatement(query);
    ResultSet rs= (ResultSet) pst.executeQuery();
      if(rs.next()){
          String NewPass=txtNewPass.getText();
          String ConfirmPass=txtConfirmPass.getText();
          if(NewPass.equals(ConfirmPass))
          {
              String query2="Update staff set Password= '"+NewPass+"' where Username= '"+txtUsername.getText()+"'";
              PreparedStatement pst2=conn.prepareStatement(query2);
              int i = pst2.executeUpdate();
        if(i>0){

            JOptionPane.showMessageDialog(null, "Password Changed Successfully");//Confirmation that data is stored in the database
            new Login().setVisible(true);
            this.dispose();
        }
        else{
            JOptionPane.showMessageDialog(null,"Operation Failed");//Notice that data is not saved in the database
        }
          }
          else
          {
              JOptionPane.showMessageDialog(null, "************New and Confirm password do not Match!!**********");
          }
      }
      else
      {
          JOptionPane.showMessageDialog(null, "************INCORRECT PASSWORD!!**********");
      }
}
catch(Exception e){
      JOptionPane.showMessageDialog(null,e);
    }
    }
TODO add your handling code here:
}
```

Corresponding error messages displayed if New and Confirm Password entered do not match, or the user enters incorrect existing login credentials.

7. Customers Window



8. Add Customer Form



Customer_ID	Customer_Name	Email_ID	Contact_NO	Address
3	Akshat Kumbhat	akshatkumbhat@gmail.com	9414132700	1A/7 mayaraj ambascheme, Udaipur
4	Pratyaksh Saraf	pratyaksh00saraf@gmail.com	6356940470	702-Saphire, Royal Gems Vapi
5	Param	parambhala9@gmail.com	7666076019	abjnd
6	John	John@gmail.com	9362748390	Dahisar, Mumbai
7	George	george@gmail.com	9362748390	Bandra, Mumbai
9	Rohan	rohan@gmail.com	4536274859	dgdhf
10	Ramesh Saraf	ramesh@gmail.com	9745364739	Dahisar East, Mumbai

A new Customer record is added to the 'Customers' table with a unique Customer ID

Code to ensure no fields are left blank and a valid email and Contact No are entered

```
    private void btnSubmitActionPerformed(java.awt.event.ActionEvent evt) {
        String Email=txtEmail.getText();
        String ContactNO=txtContactNO.getText();
        int length=ContactNO.length();
        Boolean flag=true;
        Pattern pattern = Pattern.compile("^[A-Za-z0-9._%+-]+@[A-Za-z0-9.-]+\\.[A-Z|a-z]{2,}$");
        Matcher matcher = pattern.matcher(Email);
        for(int i=0;i<length;i++)
        {
            flag=Character.isDigit(ContactNO.charAt(i));
            if(flag==false)
            {
                break;
            }
        }
if(txtName.getText().isEmpty() || txtEmail.getText().isEmpty() || txtContactNO.getText().isEmpty() || txtarAddress.getText().isEmpty())
{
    JOptionPane.showMessageDialog(this, "Please fill in all Details!", "Error", JOptionPane.ERROR_MESSAGE);
}

else if (!matcher.matches()) {
      JOptionPane.showMessageDialog(this, "Invalid Email", "Error", JOptionPane.ERROR_MESSAGE);
      txtEmail.setText("");
}
else if(flag==false || length!=10)
{
      JOptionPane.showMessageDialog(this, "Invalid Contact NO", "Error", JOptionPane.ERROR_MESSAGE);
}
```

```
else{
        String Sql="insert into customers values (?,?,?,?,?)";
        try{
            pst = conn.prepareStatement(Sql);
            pst.setInt(1,0);
            pst.setString(2, this.txtName.getText());
            pst.setString(3,this.txtEmail.getText());
            pst.setString(4,this.txtContactNO.getText());
            pst.setString(5,this.txtarAddress.getText());

            int i = pst.executeUpdate();
            if(i>0){

                JOptionPane.showMessageDialog(null, "Customer Added Successfully");
                new CustomerForm().setVisible(true);
                this.dispose();
            }
            else{
                JOptionPane.showMessageDialog(null,"Customer Addition Failed");
            }
        }
        catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
}
    }
```

Enters the details into the Customers table of the database

9. View Customer Form

Sorts the jTable based on the entered Customer ID/Name

SEARCH CUSTOMER

[SELECT]
[SELECT]
Customer_ID
Customer_Name

Search

Customer_ID	Customer_Name	Email_ID	Contact_NO	Address
3	Akshat Kumbhat	akshatkumbhat@gmail.com	9414132700	1A/7 mayaraj ambascheme, Udaipur
4	Pratyaksh Saraf	pratyaksh00saraf@gmail.com	6356940470	702-Saphire, Royal Gems Vapi
5	Param	parambhala9@gmail.com	7666076019	abjnd
6	John	John@gmail.com	9382748390	Dahisar, Mumbai

Generate PDF

Customer ID: 4

Name: Pratyaksh Saraf

Email: pratyaksh00saraf@gmail.com

Contact NO: 6356940470

Address: 702-Saphire, Royal Gems Vapi

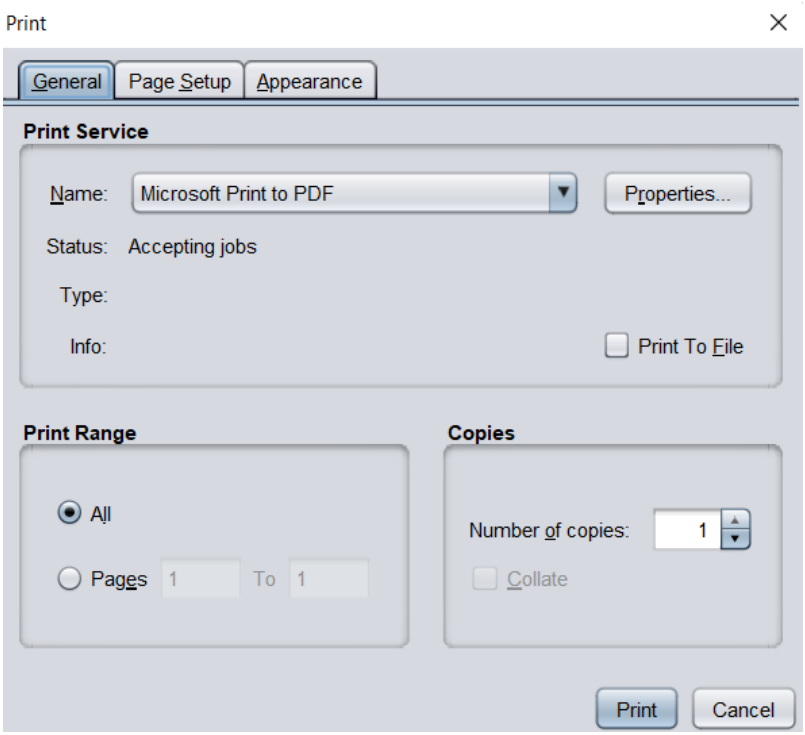
Back

Update

Generates PDF report of the data in the jTable

Updates the database based on the changes made in the jTextFields

Fills the jTextFields below with the corresponding data from the selected row



Customer Report

Customer_ID	Customer_Name	Email_ID	Contact_NO	Address
3	Akshat Kumbhat	akshatkumbhat@gmail.com	9414132700	1A/7 mayaraj ambascheme, Udaipur
4	Pratyaksh Saraf	pratyaksh00saraf@gmail.com	6356940470	702-Saphire, Royal Gems Vapi
5	Param	parambhala9@gmail.com	7666076019	abjnd
6	John	John@gmail.com	9362748390	Dahisar, Mumbai
7	George	george@gmail.com	9362748390	Bandra, Mumbai
9	Rohan	rohan@gmail.com	4536274859	dgdhf
10	Ramesh Saraf	ramesh@gmail.com	9745364739	Dahisar East, Mumbai

```
public SearchCustomer() {
    initComponents();
     conn=MySQLConnect.ConnectDB();
     Table();

}

/**
 * This method is called from within the constructor to initialize the form.
 * WARNING: Do NOT modify this code. The content of this method is always
 * regenerated by the Form Editor.
 */
@SuppressWarnings("unchecked")
Generated Code
```

The Table() function defined below is called by default when the form is invoked to display customer records

```
private void Table()
{
    try{
    String query="select * from customers";
        pst=conn.prepareStatement(query);
        ResultSet rs=(ResultSet) pst.executeQuery();
        tblCustomer.setModel(DbUtils.resultSetToTableModel(rs));

}
catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
}
```

Function to display all records under the Customers table

Rs2xml JAR file is used to display database records in a table format for all forms.

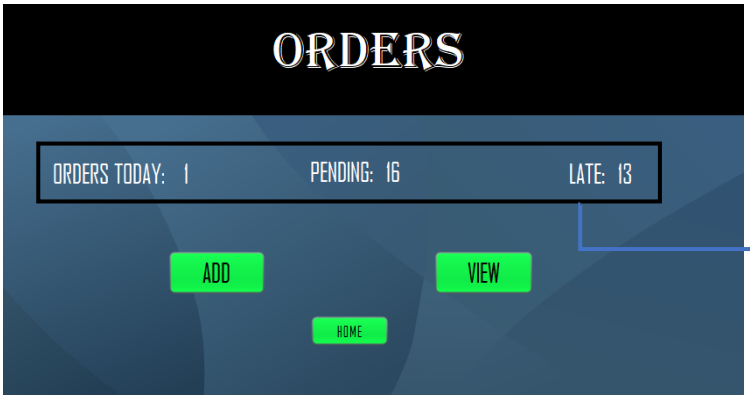
The table_name. setModel(DbUtils.resultSetToTableModel) is a function used to achieve this.

Confirmation prompt to make the changes

```java
private void btnUpdateActionPerformed(java.awt.event.ActionEvent evt) {
    int x=JOptionPane.showConfirmDialog(null, "Are you sure you want to make the changes?", "Confirmation", JOptionPane.YES_NO_OPTION);
    if(x==0)
    {
    int row=tblCustomer.getSelectedRow();
    String Sql="update customers set Customer_Name=?, Email_ID=?, Contact_NO=?, Address=? where Customer_ID="+tblCustomer.getModel().getValueAt(row, 0);
    try{
        pst = conn.prepareStatement(Sql);
        pst.setString(1, this.txtName.getText());
        pst.setString(2,this.txtEmail.getText());
        pst.setString(3,this.txtContact.getText());
        pst.setString(4,this.txtAddress.getText());

        int i = pst.executeUpdate();
        if(i>0){

            JOptionPane.showMessageDialog(null, "Changes Made Successfully");
            new SearchCustomer().setVisible(true);
            this.dispose();
        }
        else{
            JOptionPane.showMessageDialog(null,"Operation Failed");
        }
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
    }
}
```

Code to update the database record with the entered changes

10. Orders Window



Gives real-time updates of the number of orders placed on the current date, are pending and late, from the order's table of the database

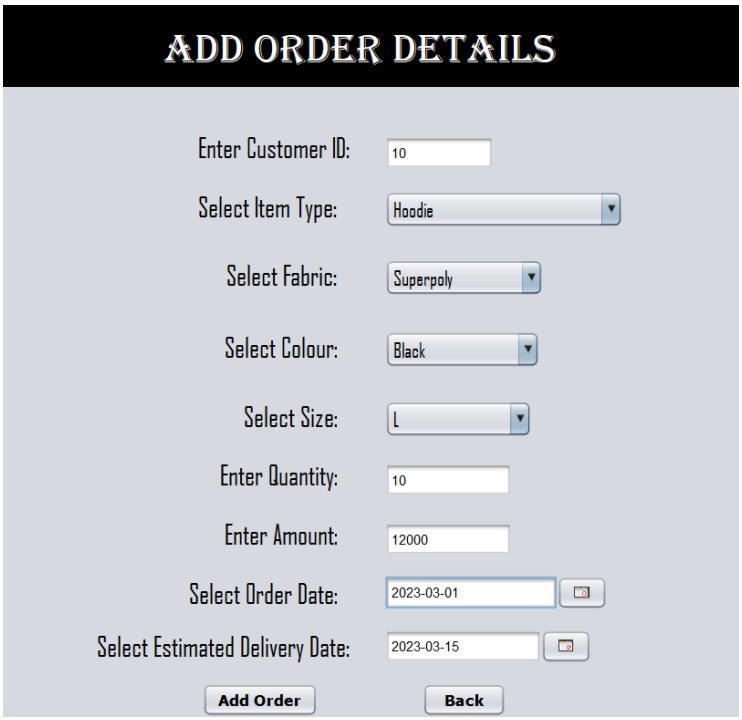
Code to display the number of orders placed on the current day from the 'orders' table

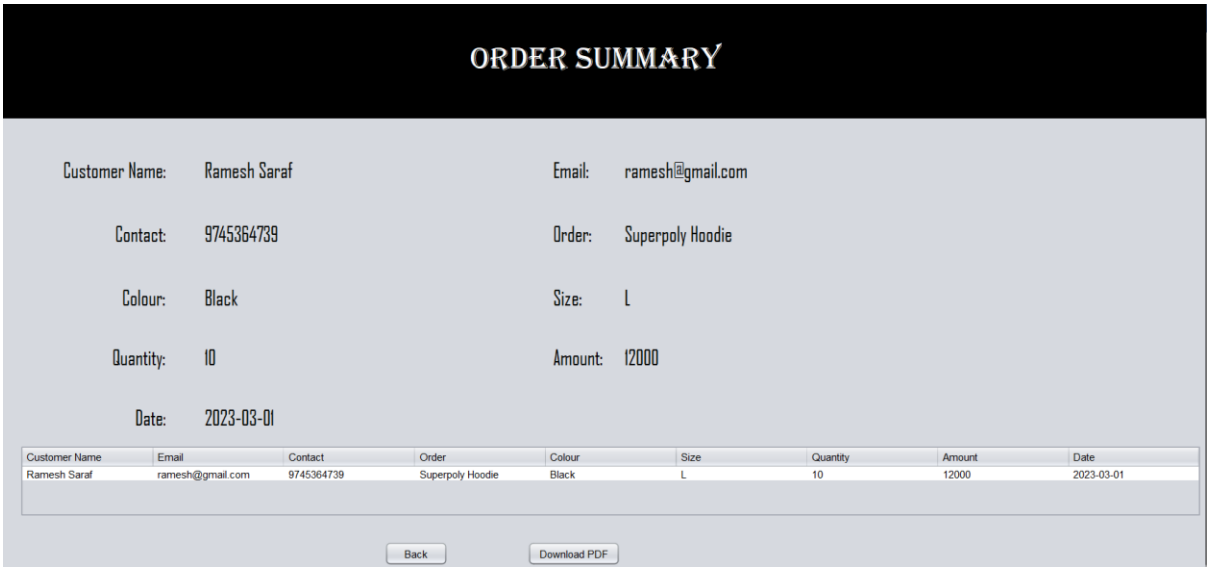
```java
private void OrderDisplay()
{
    try{
        int count;
       DateTimeFormatter dateForm=DateTimeFormatter.ofPattern("YYYY-MM-dd");
        LocalDateTime now=LocalDateTime.now();
    String query="select count(Order_Date) from orders where Order_Date like '%"+dateForm.format(now)+"%'";
    pst=conn.prepareStatement(query);
        ResultSet rs=(ResultSet) pst.executeQuery();
        while(rs.next())
        {
            count=rs.getInt(1);
            lblTdy_Ords.setText(String.valueOf(count));
        }
    String query2="select count(Order_ID) from orders where Order_Status='pending'";
    pst=conn.prepareStatement(query2);
    ResultSet rs2=(ResultSet) pst.executeQuery();
    while(rs2.next())
    {
        count=rs2.getInt(1);
        lblPending.setText(String.valueOf(count));
    }
    String query3="select count(Order_ID) from orders where Date_Estimated<'"+dateForm.format(now)+"'and Order_Status='pending'";
    pst=conn.prepareStatement(query3);
    ResultSet rs3=(ResultSet) pst.executeQuery();
    while(rs3.next())
    {
        count=rs3.getInt(1);
        lblLate.setText(String.valueOf(count));
    }

}
catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);    }}
```

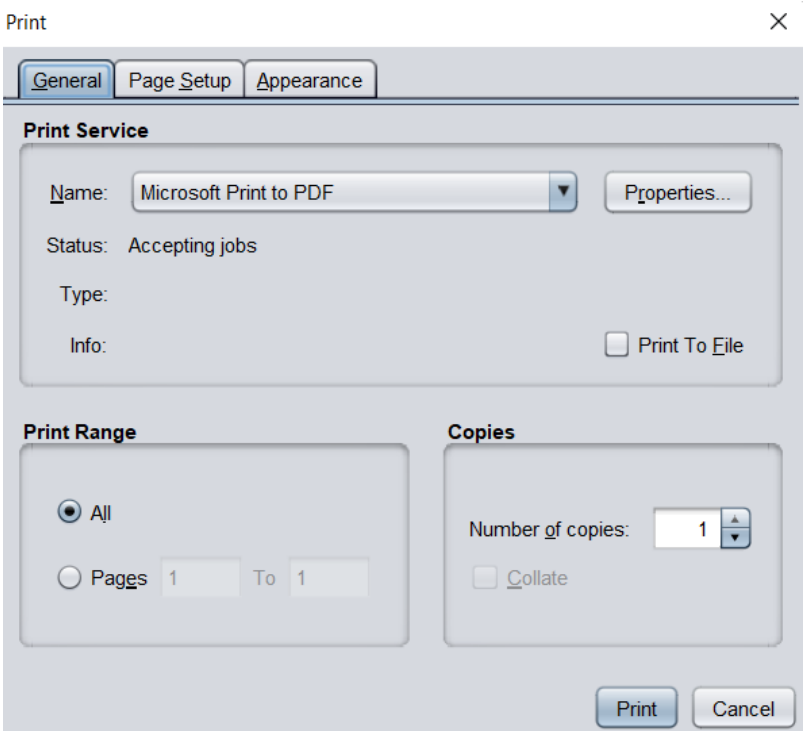
Code to display the number of orders that are 'pending' from the 'orders' table

Code to display the number of orders that are late, from the 'orders' table

11. Add Order Form



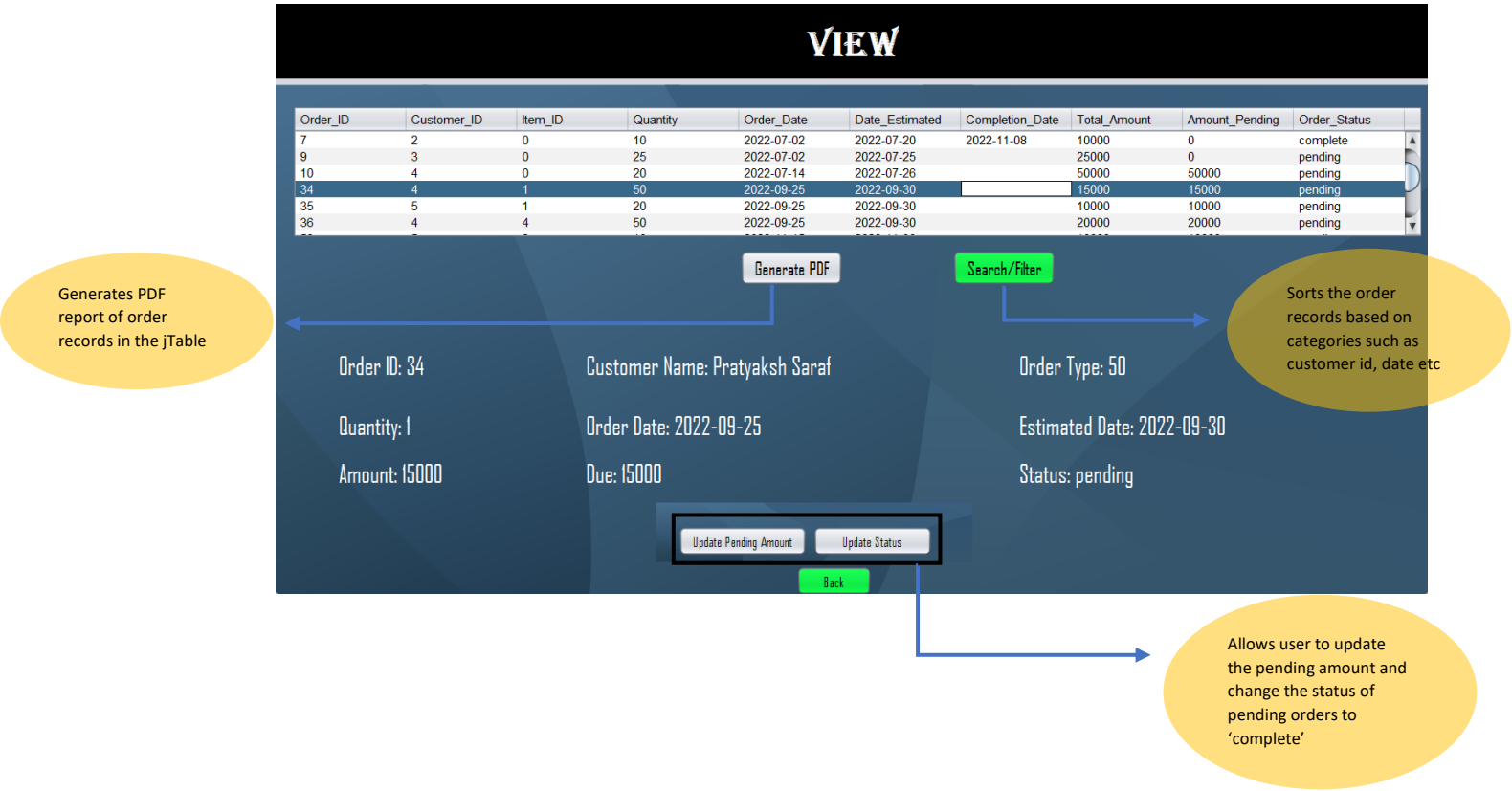


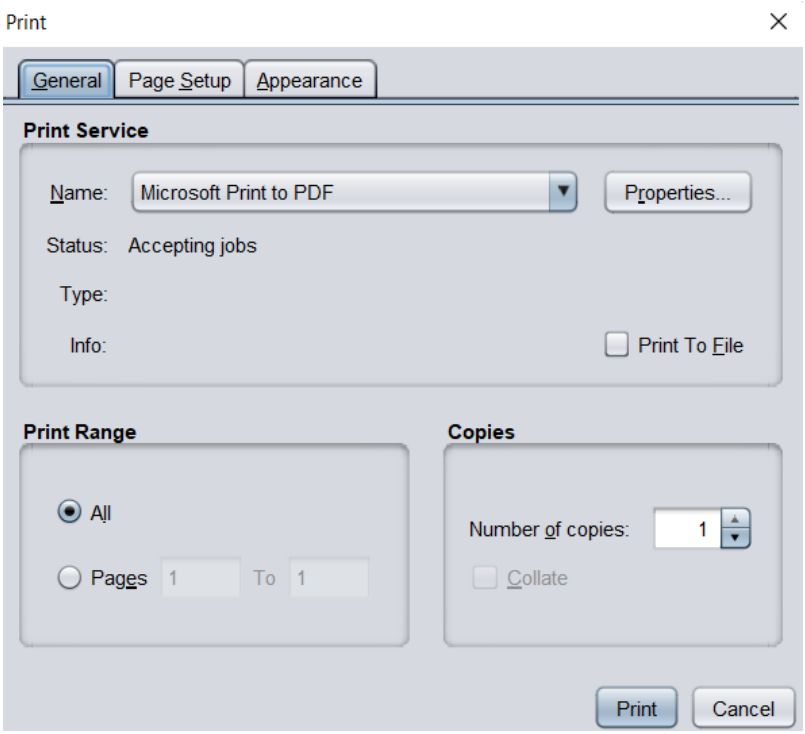


Order Receipt for Ramesh Saraf

Customer Name	Email	Contact	Order	Colour	Size	Quantity	Amount	Date
Ramesh Saraf	ramesh@gmail.com	9745364739	Superpoly Hoodie	Black	L	10	12000	2023-03-01

12. View Orders Window





Order Report

Order_ID	Customer_ID	Item_ID	Quantity	Order_Date	Date_Estimated	Completion_Date	Total_Amount	Amount_Pending	Order_Status
1	4	0	50	2022-04-23	2022-05-20	2022-05-15	30000	0	complete
2	3	0	20	2022-04-23	2022-04-10	2022-05-15	12000	0	complete
3	5	0	25	2022-04-23	2022-05-25	2022-05-26	30000	0	complete
4	4	0	30	2022-04-24	2022-05-10	2022-07-24	20000	0	complete
5	5	0	15	2022-04-18	2022-05-05	2022-09-12	25000	0	complete
6	6	0	20	2022-04-24	2022-05-24	2022-05-23	20000	0	complete
7	2	0	10	2022-07-02	2022-07-20	2022-11-08	10000	0	complete
9	3	0	25	2022-07-02	2022-07-25		25000	0	pending
10	4	0	20	2022-07-14	2022-07-26		50000	50000	pending
34	4	1	50	2022-09-25	2022-09-30		15000	15000	pending
35	5	1	20	2022-09-25	2022-09-30		10000	10000	pending
36	4	4	50	2022-09-25	2022-09-30		20000	20000	pending
58	5	2	10	2022-11-15	2022-11-30		10000	10000	pending
59	4	2	10	2023-01-07	2023-01-31		10000	10000	pending
60	6	2	5	2023-01-07	2023-01-31		5000	5000	pending
62	3	2	10	2023-01-07	2023-01-31		15000	15000	pending
64	7	2	10	2023-01-06	2023-01-19		10000	10000	pending
65	7	2	5	2023-01-01	2023-01-10		5000	5000	pending
66	4	2	50	2023-02-06	2023-02-28		50000	50000	pending
67	5	2	20	2023-02-06	2023-02-28		20000	20000	pending
68	5	2	10	2023-02-28	2023-03-26		10000	10000	pending
69	6	2	10	2023-02-27	2023-03-15		15000	15000	pending
70	7	2	10	2023-03-01	2023-03-15		10000	10000	pending
71	10	5	10	2023-03-01	2023-03-15		12000	12000	pending

```java
public void DisplayOrders()
{
    try{
    String query="select * from orders";
        pst=conn.prepareStatement(query);
        ResultSet rs=(ResultSet) pst.executeQuery();
        tblOrders.setModel(DbUtils.resultSetToTableModel(rs));

}
catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
    MyPanel.hide();

}
```

Code to display the order records from the 'orders' table into a jTable

```java
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
String option[]={"[SELECT]","Order ID","Customer ID","Order Date","Status"};
JComboBox cb=new JComboBox(option);
int input;
input=JOptionPane.showConfirmDialog(this,cb,"SELECT CATEGORY",JOptionPane.DEFAULT_OPTION);
if(input==JOptionPane.OK_OPTION)
{
    String choice=(String)cb.getSelectedItem();
    if("[SELECT]".equals(choice))
    JOptionPane.showMessageDialog(this, "**Select a Category!!**");
    else if (choice.equals("Order ID"))
    {
        try{
        String OrderID=JOptionPane.showInputDialog(this,"Enter Order ID");
        String query = "select * from orders where Order_ID="+OrderID;
        pst=conn.prepareStatement(query);
        ResultSet rs=pst.executeQuery();
        tblOrders.setModel(DbUtils.resultSetToTableModel(rs));
        }
        catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
       }
    }
    else if(choice.equals("Customer ID")){
        try{
        String CustomerID=JOptionPane.showInputDialog(this,"Enter Customer ID");
        String query = "select * from orders where Customer_ID="+CustomerID;
        pst=conn.prepareStatement(query);
        ResultSet rs=pst.executeQuery();
        tblOrders.setModel(DbUtils.resultSetToTableModel(rs));
        }
        catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);} }
```

Code to ensure that the user chooses a valid category

Code to display order records based on the entered 'Order ID'

Code to display order records based on the entered 'Customer ID'

Code to display order records based on the 'status' chosen i.e., 'complete' or 'pending'

```
else if(choice.equals("Status"))
{
    String status[]={"[SELECT]","pending","complete"};
    JComboBox cb2=new JComboBox(status);
    int input2=JOptionPane.showConfirmDialog(this,cb2,"SELECT STATUS",JOptionPane.DEFAULT_OPTION);
    if(input==JOptionPane.OK_OPTION)
    {
        String choice2=(String)cb2.getSelectedItem();
        if("[SELECT]".equals(choice2))
        JOptionPane.showMessageDialog(this, "**Select a Category!!**");
        else{
            try{
    String query = "select * from orders where Order_Status='"+choice2+"'";
    pst=conn.prepareStatement(query);
    ResultSet rs=pst.executeQuery();
    tblOrders.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(HeadlessException | SQLException e){
       JOptionPane.showMessageDialog(null, e);
    }
            }
        }
    }
```

Code to display records in table based on the chosen Order Dates

```
else
{
    JDateChooser startdate=new JDateChooser();//code received from https://stackoverflow.com/questions
    String message ="Choose start date:\n";
    Object[] SDateparams={message,startdate};
    JOptionPane.showConfirmDialog(null,SDateparams,"Start date", JOptionPane.PLAIN_MESSAGE);
    String sdateinput="";
    SimpleDateFormat sdf=new SimpleDateFormat("YYYY-MM-dd");
    sdateinput=sdf.format(((JDateChooser)SDateparams[1]).getDate());

    JDateChooser enddate=new JDateChooser();
    message="Choose end date:\n";
    Object [] EDateparams={message,enddate};
    JOptionPane.showConfirmDialog(null,EDateparams,"End date", JOptionPane.PLAIN_MESSAGE);
    String edateinput="";
    edateinput=sdf.format(((JDateChooser)EDateparams[1]).getDate());
    try{

   String query="select * from orders where Order_Date between '"+sdateinput+"' and '"+edateinput+"'";
   pst=conn.prepareStatement(query);
   ResultSet rs=(ResultSet) pst.executeQuery();
   tblOrders.setModel(DbUtils.resultSetToTableModel(rs));
}
catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Code to get the row selected by the user by Mouse-click and set the JTextFields with its corresponding Customer data

```
    private void tblOrdersMouseClicked(java.awt.event.MouseEvent evt) {
int row=tblOrders.getSelectedRow();
            lblOrderID.setText("Order ID: "+tblOrders.getModel().getValueAt(row, 0).toString());
            try{

        String query="select Customer_Name from customers where Customer_ID="+tblOrders.getModel().getValueAt(row, 1).toString();
        pst=conn.prepareStatement(query);
        ResultSet rs=(ResultSet) pst.executeQuery();
        if(rs.next())
        lblCustomerName.setText("Customer Name: "+rs.getString(1));
    }
    catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
            lblOrderType.setText("Order Type: "+tblOrders.getModel().getValueAt(row, 3).toString());
            lblQuantity.setText("Quantity: "+tblOrders.getModel().getValueAt(row, 2).toString());
            lblOrderDate.setText("Order Date: "+tblOrders.getModel().getValueAt(row, 4).toString());
            lblEstmDate.setText("Estimated Date: "+tblOrders.getModel().getValueAt(row, 5).toString());
            lblAmt.setText("Amount: "+tblOrders.getModel().getValueAt(row, 7).toString());
            lblDue.setText("Due: "+tblOrders.getModel().getValueAt(row, 8).toString());
            lblStatus.setText("Status: "+tblOrders.getModel().getValueAt(row, 9).toString());
            if(tblOrders.getModel().getValueAt(row,9).toString().equals("pending"))
            MyPanel.show();
            else
            MyPanel.hide();
    }
```

Enables users to update the 'status' and the 'price dues' of the order only if the 'order_status' of the chosen record is 'pending'

Code to update the pending amount of the selected order, only if it is 'pending'

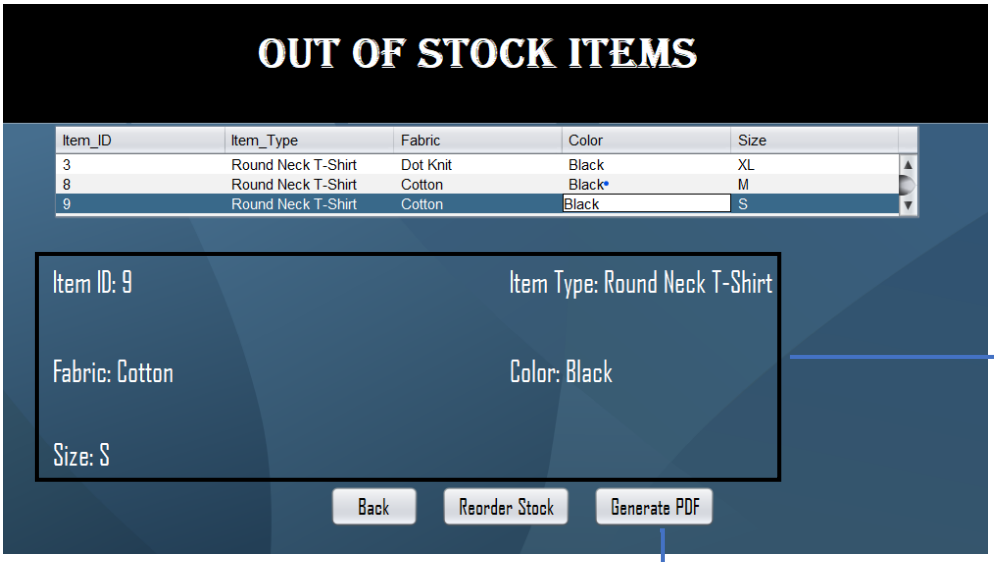
```
    private void btnpenamtActionPerformed(java.awt.event.ActionEvent evt) {
int row=tblOrders.getSelectedRow();
String penamt=JOptionPane.showInputDialog(this, "Enter New Pending Amount");
try{

        String query="update orders set Amount_Pending="+penamt+" where Order_ID="+tblOrders.getModel().getValueAt(row, 0).toString();
        pst=conn.prepareStatement(query);
        int i=pst.executeUpdate();
        if(i>0)
        {
            JOptionPane.showMessageDialog(null,"Pending Amount updated successfully!" );
            new ViewOrders().setVisible(true);
            this.dispose();
        }

    }
    catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
    }
```

```
private void btnStatusActionPerformed(java.awt.event.ActionEvent evt) {
  int row=tblOrders.getSelectedRow();
   try{
    String query="select Amount_Pending from orders where Order_ID="+tblOrders.getModel().getValueAt(row, 0).toString();
    pst=conn.prepareStatement(query);
    ResultSet rs=(ResultSet) pst.executeQuery();
    if(rs.next()){
        if(rs.getString(1).equals("0")) {
            JDateChooser completionDate=new JDateChooser();
     String message ="Choose Order Completion Date:\n";
     Object[] SDateparams={message,completionDate};
     JOptionPane.showConfirmDialog(null,SDateparams,"Completion date", JOptionPane.PLAIN_MESSAGE);
     String compdateinput="";
     SimpleDateFormat sdf=new SimpleDateFormat("YYYY-MM-dd");
     compdateinput=sdf.format(((JDateChooser)SDateparams[1]).getDate());
     String query2="update orders set Order_Status='complete', Completion_Date='"+compdateinput+"' where Order_ID="+tblOrders.getModel().getValueAt(row, 0).toString();
     System.out.print(query2);
    pst=conn.prepareStatement(query2);
    int i=pst.executeUpdate();
    if(i>0)
    {
        JOptionPane.showMessageDialog(null,"Status updated successfully!" );
        new ViewOrders().setVisible(true);
        this.dispose();
    }
        }
        else
        JOptionPane.showMessageDialog(null, "**Please clear the Order's Pending amount first!**");
    }
}
catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
```

Code to update the status of 'pending' orders to 'complete' only if it has 0 dues, prompting the user to enter the order completion date

13. Inventory Window

14. Out of Stock Orders Window



Generates PDF report of the Out-of-Stock items from the jTable

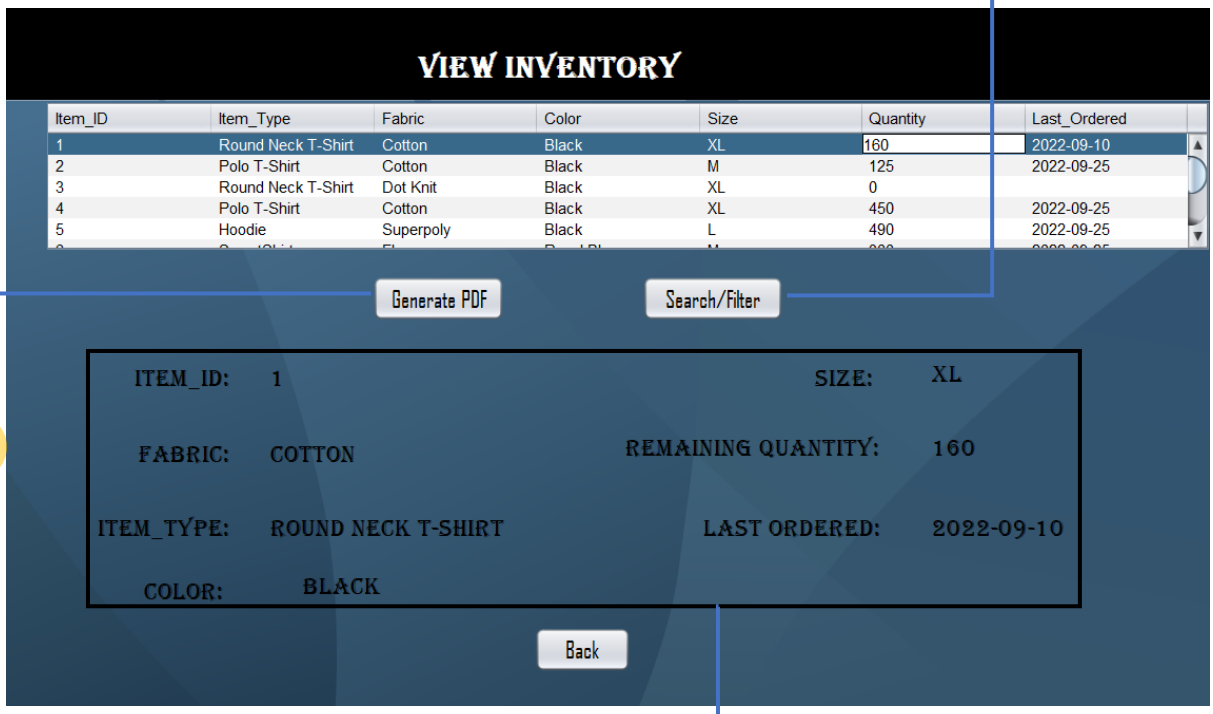
Fills the jTextFields with the corresponding data of selected item in the jTable

```
    private void DisplayOutofStockItems()
    {
        try{
    String query="select Item_ID,Item_Type,Fabric,Color,Size from inventory where Quantity=0";
        pst=conn.prepareStatement(query);
        ResultSet rs=(ResultSet) pst.executeQuery();
        tblOutOfStock.setModel(DbUtils.resultSetToTableModel(rs));

}
catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
    }
```

Code to display the item records from 'inventory' table which are out of stock, i.e., 'Quantity'=0

Sorts the inventory records based on categories such as Item_Id, Fabric, Quantity

15. View Inventory Window



Item_ID	Item_Type	Fabric	Color	Size	Quantity	Last_Ordered
1	Round Neck T-Shirt	Cotton	Black	XL	160	2022-09-10
2	Polo T-Shirt	Cotton	Black	M	125	2022-09-25
3	Round Neck T-Shirt	Dot Knit	Black	XL	0	
4	Polo T-Shirt	Cotton	Black	XL	450	2022-09-25
5	Hoodie	Superpoly	Black	L	490	2022-09-25

Generates PDF report of inventory records in the jTable

Fills the jTextFields with the corresponding data of selected item in the jTable

Inventory Report

Item_ID	Item_Type	Fabric	Color	Size	Quantity	Last_Ordered
1	Round Neck T-Shirt	Cotton	Black	XL	160	2022-09-10
2	Polo T-Shirt	Cotton	Black	M	125	2022-09-25
3	Round Neck T-Shirt	Dot Knit	Black	XL	0	
4	Polo T-Shirt	Cotton	Black	XL	450	2022-09-25
5	Hoodie	Superpoly	Black	L	490	2022-09-25
6	SweatShirt	Fleece	Royal Blue	M	300	2022-09-25
7	SweatShirt	Superpoly	Red	S	400	2022-09-24
8	Round Neck T-Shirt	Cotton	Black	M	0	
9	Round Neck T-Shirt	Cotton	Black	S	0	
10	Polo T-Shirt	Dot Knit	White	S	0	
11	Round Neck T-Shirt	Dot Knit	White	S	0	

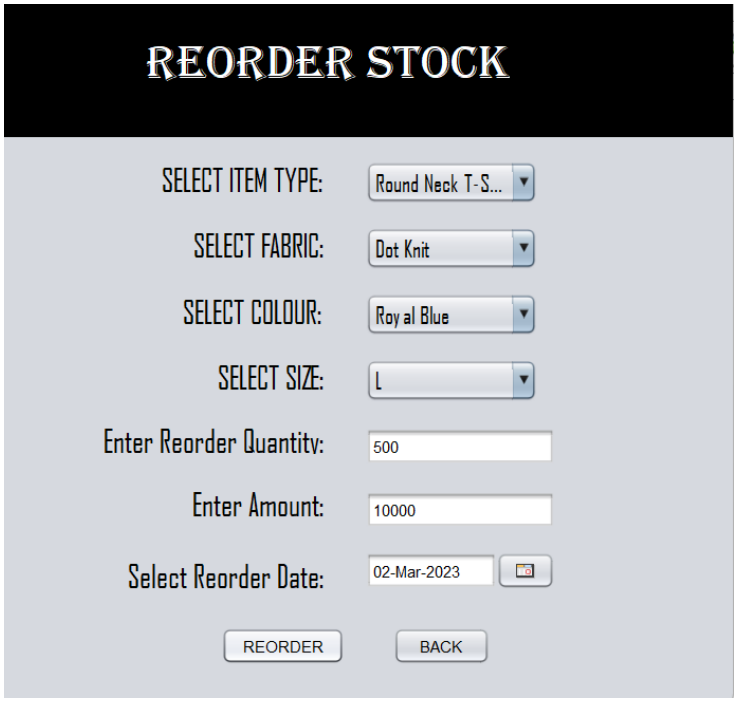
```java
    private void jButton2ActionPerformed(java.awt.event.ActionEvent evt) {
String option[]={"[SELECT]","Item ID","Type","Fabric","Color","Quantity","Last Order Date"};
 JComboBox cb=new JComboBox(option);
 int input;
 input=JOptionPane.showConfirmDialog(this,cb,"SELECT CATEGORY",JOptionPane.DEFAULT_OPTION);          // TODO
if(input==JOptionPane.OK_OPTION)
 {
     String choice=(String)cb.getSelectedItem();

     if("[SELECT]".equals(choice))

     JOptionPane.showMessageDialog(this, "**Select a Category!!**");
     else if (choice.equals("Item ID"))
     {
         try{
         String ItemID=JOptionPane.showInputDialog(this,"Enter Item ID");
         String query = "select * from inventory where Item_ID="+ItemID;
         pst=conn.prepareStatement(query);
         ResultSet rs=pst.executeQuery();
         tblInventory.setModel(DbUtils.resultSetToTableModel(rs));
         }
         catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
     }
 else if(choice.equals("Type"))
 {
     String type[]={"[SELECT]","Round Nect T-Shirt","Polo T-Shirt","SweatShirt","Hoodie"};
     JComboBox cb2=new JComboBox(type);
     int input2=JOptionPane.showConfirmDialog(this,cb2,"SELECT ITEM TYPE",JOptionPane.DEFAULT_OPTION);
     if(input==JOptionPane.OK_OPTION)
     {
         String choice2=(String)cb2.getSelectedItem();
         if("[SELECT]".equals(choice2))
         JOptionPane.showMessageDialog(this, "**Select a Category!!**");
         else{
             try{
     String query = "select * from inventory where Item_Type='"+choice2+"'";
     pst=conn.prepareStatement(query);
     ResultSet rs=pst.executeQuery();
     tblInventory.setModel(DbUtils.resultSetToTableModel(rs));
     }
     catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }

         }
     }
 }
else if(choice.equals("Color"))
{
    String color[]={"[SELECT]","Black","White","Royal Blue","Red","Green","Orange","Navy Blue","Sky Blue","Yellow","Sea Green","Maroon","Grey"};
    JComboBox cb2=new JComboBox(color);
    int input2=JOptionPane.showConfirmDialog(this,cb2,"SELECT COLOR",JOptionPane.DEFAULT_OPTION);
    if(input==JOptionPane.OK_OPTION)
    {
        String choice2=(String)cb2.getSelectedItem();
        if("[SELECT]".equals(choice2))
        JOptionPane.showMessageDialog(this, "**Select a Category!!**");
        else{
            try{
    String query = "select * from inventory where Color='"+choice2+"'";
    pst=conn.prepareStatement(query);
    ResultSet rs=pst.executeQuery();
    tblInventory.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(HeadlessException | SQLException e){
       JOptionPane.showMessageDialog(null, e);
   }

        }
    }
}
```

Sorts the JTable based on the chosen 'Item_ID', 'Item_Type' or 'Color'

```java
else if (choice.equals("Quantity"))
{
    try{
    String MinQuantity=JOptionPane.showInputDialog(this,"Enter Minimum Quantity");
    String MaxQuantity=JOptionPane.showInputDialog(this,"Enter Maximum Quantity");
    String query = "select * from inventory where Quantity >= "+MinQuantity+" and Quantity <= "+MaxQuantity;
    pst=conn.prepareStatement(query);
    ResultSet rs=pst.executeQuery();
    tblInventory.setModel(DbUtils.resultSetToTableModel(rs));
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }
}
}
}
```

Displays the item records in JTable with quantity between the bounds entered by user

16. Reorder Stock Form



Reorder_ID	Item_ID	Quantity	Reorder_Date
6	4	500	2022-09-25
7	5	500	2022-09-25
8	6	300	2022-09-25
9	7	400	2022-09-24
10	2	500	2022-09-25
11	12	500	2023-03-02

Successfully adds the record into the 'Reorders' table, assigning a unique new Reorder_ID.

Validation to ensure no field is left blank

```
private void btnReorderActionPerformed(java.awt.event.ActionEvent evt) {
if(txtQuantity.getText().isEmpty() || txtAmount.getText().isEmpty() || dtReorder.getDate().toString().isEmpty())

JOptionPane.showMessageDialog(this, "Please fill in all Details!", "Error", JOptionPane.ERROR_MESSAGE);

else{
    String type,fabric,colour,size;
Date ReorderDate=dtReorder.getDate();
SimpleDateFormat sdf=new SimpleDateFormat("YYYY-MM-dd");
    boolean check=false;
    type=jComboBox1.getSelectedItem().toString();
    fabric=jComboBox2.getSelectedItem().toString();
    colour=jComboBox3.getSelectedItem().toString();
    size=jComboBox4.getSelectedItem().toString();
    if(type=="[SELECT]"||fabric=="[SELECT]"||colour=="[SELECT]"||size=="[SELECT]")
        JOptionPane.showMessageDialog(null, "****Invalid Selection!!****");
```

Adds the new item into the 'inventory' table if not already present

```
    else{
        int Item_ID = 0,Quantity;
        try{
            String sql="select Item_ID,Quantity from inventory where Item_Type= '"+type+"' and Fabric= '"+fabric+"' and Color= '"+colour+"' and Size= '"+size+"'";
            pst=conn.prepareStatement(sql);
            ResultSet rs=(ResultSet) pst.executeQuery();
     while(rs.next())
     {

         check=true;
         Item_ID=rs.getInt("Item_ID");
         System.out.println("test");
         Quantity=rs.getInt("Quantity");
         String sql3="update inventory set Quantity= "+Quantity+"+"+txtQuantity.getText()+", Last_Ordered= '"+sdf.format(ReorderDate)+"' where Item_ID= "+Item_ID;
         System.out.println(sql3);
         pst=conn.prepareStatement(sql3);
         pst.executeUpdate();
     }

    }
catch(SQLException e){
    JOptionPane.showMessageDialog(null, e);
    }
    if(check==false)
    {
        try{
        String sql4="insert into inventory values (?,?,?,?,?,?,?)";
     pst=conn.prepareStatement(sql4);
     pst.setInt(1,0);
     pst.setString(2,type);
     pst.setString(3, fabric);
     pst.setString(4,colour);
     pst.setString(5,size);
     pst.setInt(6,Integer.parseInt(txtQuantity.getText()));
     pst.setString(7,sdf.format(ReorderDate));
     int j=pst.executeUpdate();
     String sql5="select Item_ID from inventory where Item_Type= '"+type+"' and Fabric= '"+fabric+"' and Color= '"+colour+"' and Size= '"+size+"'";
     pst=conn.prepareStatement(sql5);
     ResultSet rs1=(ResultSet) pst.executeQuery();
     while(rs1.next())
     {
         Item_ID=rs1.getInt("Item_ID");
     }
    }
        catch(SQLException e){
    JOptionPane.showMessageDialog(null, e);
      }
    }
```

Updates the inventory with the newly added quantity, given that the item already exists in the 'inventory' table

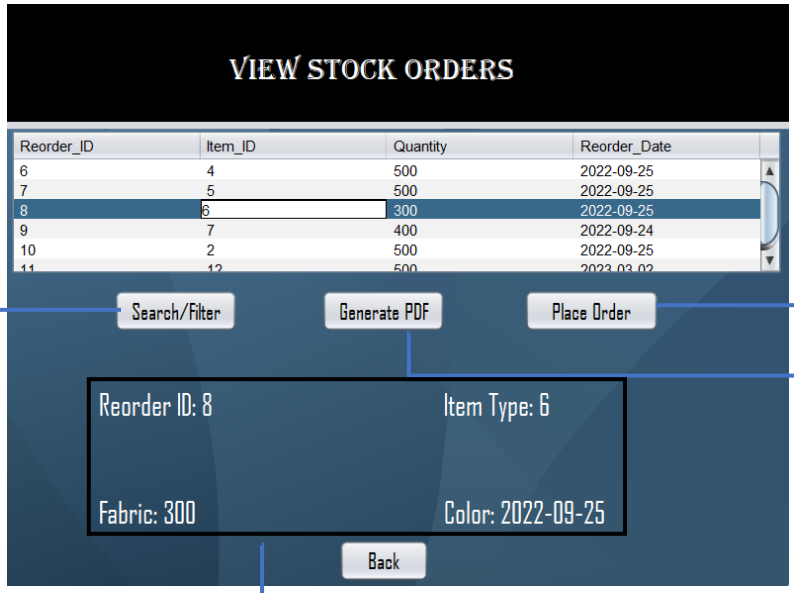
```
        try{
        String sql2="insert into Reorders values(?,?,?,?)";
         pst=conn.prepareStatement(sql2);
         pst.setInt(1,0);
         pst.setInt(2,Item_ID);
         pst.setInt(3,Integer.parseInt(txtQuantity.getText()));
         pst.setString(4,sdf.format(ReorderDate));
         int i=pst.executeUpdate();
         if(i>0)
         {
             JOptionPane.showMessageDialog(null, "Reorder Made Successfully");
             Orders ords= new Orders(null);
       ords.setVisible(true);//Displays the Customers Interface
       setVisible(false);
         }
        }
        catch(SQLException e){
        JOptionPane.showMessageDialog(null, e);
          }

       }
    }

    }
```

Adds the new order record into the 'reorders' table

17. Stock Orders Window

Redirects user to Stock Reorder form

Sorts the stock order records based on categories such as Reorder_ID, Item_Id, Date and Quantity



Generates PDF report of inventory records in the jTable

Fills the jTextFields with the corresponding data of selected item in the jTable

Stock Reorder Report

Reorder_ID	Item_ID	Quantity	Reorder_Date
6	4	500	2022-09-25
7	5	500	2022-09-25
8	6	300	2022-09-25
9	7	400	2022-09-24
10	2	500	2022-09-25
11	12	500	2023-03-02

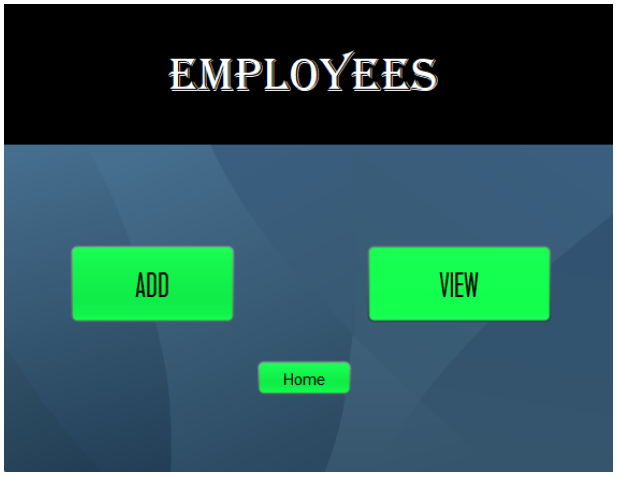
```java
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
String option[]={"[SELECT]","Reorder ID","Item ID","Quantity","Date"};
 JComboBox cb=new JComboBox(option);
 int input;
 input=JOptionPane.showConfirmDialog(this,cb,"SELECT CATEGORY",JOptionPane.DEFAULT_OPTION);
if(input==JOptionPane.OK_OPTION)
 {
     String choice=(String)cb.getSelectedItem();

     if("[SELECT]".equals(choice))

     JOptionPane.showMessageDialog(this, "**Select a Category!!**");
     else if (choice.equals("Reorder ID"))
     {
         try{
         String ReorderID=JOptionPane.showInputDialog(this,"Enter Reorder ID");
         String query = "select * from reorders where Reorder_ID="+ReorderID;
         pst=conn.prepareStatement(query);
         ResultSet rs=pst.executeQuery();
         tblReorders.setModel(DbUtils.resultSetToTableModel(rs));
         }
         catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }
     }

         else if (choice.equals("Item ID"))
         {
             try{
             String ItemID=JOptionPane.showInputDialog(this,"Enter Item ID");
             String query = "select * from reorders where Item_ID="+ItemID;
             pst=conn.prepareStatement(query);
             ResultSet rs=pst.executeQuery();
             tblReorders.setModel(DbUtils.resultSetToTableModel(rs));
             }
             catch(HeadlessException | SQLException e){
                JOptionPane.showMessageDialog(null, e);
            }
         }
         else if (choice.equals("Quantity"))
         {
             try{
             String MinQuantity=JOptionPane.showInputDialog(this,"Enter Minimum Quantity");
             String MaxQuantity=JOptionPane.showInputDialog(this,"Enter Maximum Quantity");
             String query = "select * from reorders where Quantity >= "+MinQuantity+" and Quantity <= "+MaxQuantity;
             pst=conn.prepareStatement(query);
             ResultSet rs=pst.executeQuery();
             tblReorders.setModel(DbUtils.resultSetToTableModel(rs));
             }
             catch(HeadlessException | SQLException e){
                JOptionPane.showMessageDialog(null, e);
            }
         }

           else
          {
             JDateChooser startdate=new JDateChooser();//code received from https://stackoverflow.com/questions/3534
             String message ="Choose start date:\n";
             Object[] SDateparams={message,startdate};
             JOptionPane.showConfirmDialog(null,SDateparams,"Start date", JOptionPane.PLAIN_MESSAGE);
             String sdateinput="";
             SimpleDateFormat sdf=new SimpleDateFormat("YYYY-MM-dd");
             sdateinput=sdf.format(((JDateChooser)SDateparams[1]).getDate());

             JDateChooser enddate=new JDateChooser();
             message="Choose end date:\n";
             Object [] EDateparams={message,enddate};
             JOptionPane.showConfirmDialog(null,EDateparams,"End date", JOptionPane.PLAIN_MESSAGE);
             String edateinput="";
             edateinput=sdf.format(((JDateChooser)EDateparams[1]).getDate());
             try{

            String query="select * from reorders where Reorder_Date between '"+sdateinput+"' and '"+edateinput+"'";
            pst=conn.prepareStatement(query);
            ResultSet rs=(ResultSet) pst.executeQuery();
            tblReorders.setModel(DbUtils.resultSetToTableModel(rs));
          }
          catch(HeadlessException | SQLException e){
                JOptionPane.showMessageDialog(null, e);
            }
         }
         TODO add your handling code here:
     }
```

Sorts the JTable with those items from 'reorders' table, with the entered Reorder/Item ID

Sorts the JTable by displaying items from the 'reorders' table with records within the entered Quantity/Date range

18. Employees Window

This functionality is only available to the Managers, who have access to the Manager's Dashboard.



19. Add Employee Form

Staff_ID	Staff_Name	Username	Password	Role	Salary
1	Shalabh Agrawal	ShalabhAgra	1234	Manager	150000
3	Pratyaksh Saraf	Pratyaksh	pratyaksh	EMPLOYEE	30000
4	Ashish Bhatt	Ashish	1234	EMPLOYEE	25000
5	ABC	ABC	1234	EMPLOYEE	25000
6	Kunal Saraf	Kunal_1	abcd	EMPLOYEE	25000

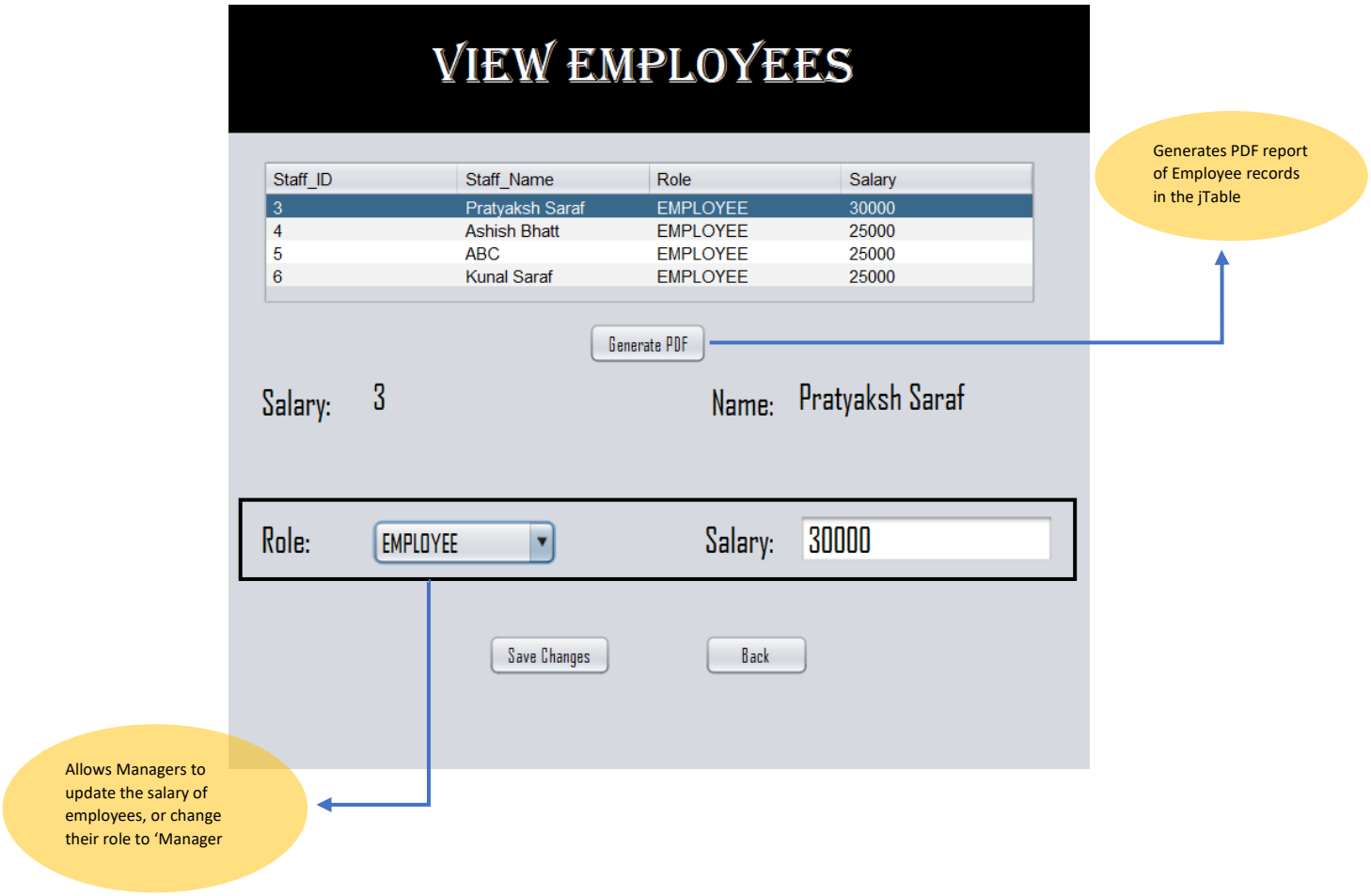
Successfully adds the new Employee record into the 'Staff' Database

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    int length=txtSalary.getText().length();
        Boolean flag=true;
        for(int i=0;i<length;i++)
        {
            flag=Character.isDigit(txtSalary.getText().charAt(i));
            if(flag==false)
            {
                break;
            }
        }
        if(txtEmployeeName.getText().isEmpty() || txtUsername.getText().isEmpty() || txtPassword.getText().isEmpty() || txtSalary.getText().isEmpty() )
{
    JOptionPane.showMessageDialog(this, "Please fill in all Details!", "Error", JOptionPane.ERROR_MESSAGE);
}

        else if(flag==false)
{
     JOptionPane.showMessageDialog(this, "Invalid Salary!", "Error", JOptionPane.ERROR_MESSAGE);
}
        else{
        String Sql="insert into staff values (?,?,?,?,?,?)";
        try{
            pst = conn.prepareStatement(Sql);
            pst.setInt(1,0);
            pst.setString(2, this.txtEmployeeName.getText());
            pst.setString(3,this.txtUsername.getText());
            pst.setString(4,this.txtPassword.getText());
            pst.setString(5,"EMPLOYEE");
            pst.setString(6,this.txtSalary.getText());

        int i = pst.executeUpdate();
        if(i>0){

            JOptionPane.showMessageDialog(null, "Employee Added Successfully");
        }
        else{
            JOptionPane.showMessageDialog(null,"Employee Addition Failed");
        }
    }
    catch(HeadlessException | SQLException e){
        JOptionPane.showMessageDialog(null, e);
    }    // TODO add your handling code here:
    }
}
```

Code to ensure that a valid salary is entered and no field is left bank

Adds the Employee details into the database, creating a new Employee record

18. Add Employee Form



Employee Report

Staff_ID	Staff_Name	Role	Salary
3	Pratyaksh Saraf	EMPLOYEE	30000
4	Ashish Bhatt	EMPLOYEE	25000
5	ABC	EMPLOYEE	25000
6	Kunal Saraf	EMPLOYEE	25000

Updates the Employee record in database based on changes made in salary/role

```
    private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
int x=JOptionPane.showConfirmDialog(null, "Are you sure you want to make the changes?", "Confirmation", JOptionPane.YES_NO_OPTION);//Confirmation prompt
        if(x==0)//If the user clicks 'yes' in the confirmation prompt
        {
        int row=tblEmployees.getSelectedRow();
        String Sql="update staff set Role=?, Salary=? where Staff_ID="+tblEmployees.getModel().getValueAt(row, 0);//query to update the staff table
        try{
            pst = conn.prepareStatement(Sql);
            pst.setString(1,comboRole.getSelectedItem().toString());
            pst.setString(2,this.txtSalary.getText());

            int i = pst.executeUpdate();
            if(i>0){

                JOptionPane.showMessageDialog(null, "Changes Made Successfully");//Confirmation that data is stored in the database
            }
            else{
                JOptionPane.showMessageDialog(null,"Operation Failed");//Notice that data is not saved in the database
            }
        }
        catch(HeadlessException | SQLException e){
            JOptionPane.showMessageDialog(null, e);
        }


    }               // TODO add your handling code here:
    }
```

Word Count: 949

References

"Java Swing Netbeans Ide Gui Tutorial." *YouTube*, YouTube, www.youtube.com/playlist?list=PLNHw_0qv1zy-cVdEX6HqJX2a8ZLHbG3Ox.

"Joptionpane Message Dialog Types in Java Swing." *YouTube*, YouTube, 20 Apr. 2019, www.youtube.com/watch?v=sKgsNPWjuNo&list=LL&index=25.

"Java Prog#26.How to Print Jtable in Java Netbeans." *YouTube*, YouTube, 14 May 2012, www.youtube.com/watch?v=yhqu-NG-AzY&list=LL&index=6.