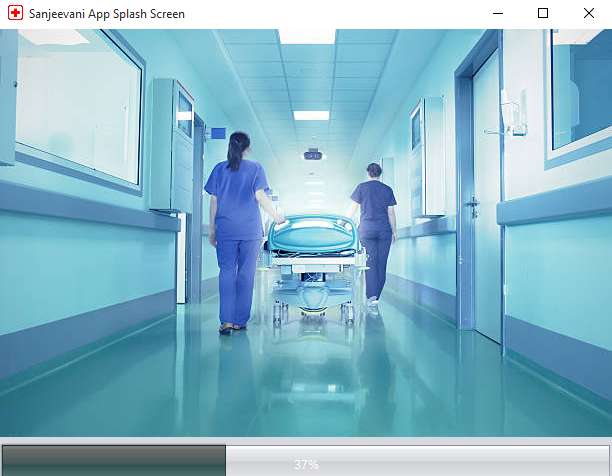
**Designing The SplashScreenFrame**

****

**STEPS TO BE DONE IN** **SplashScreenFrame**

A Splash Screen is the first screen displayed by the app and contains app title . It stays open for some duration and then disappears giving way to the next screen.

In the **SplashScreenFrame** we need to do following steps:

1. Create a **JProgressBar.**

2. It's important methods are:

**setStringPainted( )**

**setValue( )**

**getValue( )**

**getMaximum( )**

3. Call it's **setStringPainted ( )** method passing it the argument true which enables the Progress Bar to show progress string.

3. Create a class which extends **Thread**

4. Override it's **run( )** method

5. Launch the thread

6. When **run()** method finishes then load the **LoginFrame**

***public class SplashScreenFrame extends javax.swing.JFrame {***

***SplashThread sp;***

***public SplashScreenFrame() {***

***initComponents();***

***this.setLocationRelativeTo(null);***

***jProgressBar1.setStringPainted(true);***

***jProgressBar1.setForeground(Color.white);***

***Color bkColor=new Color(47,79,79);***

***ColorUIResource colorResource = new ColorUIResource(bkColor);***

***UIManager.put("nimbusOrange",colorResource);***

***sp=new SplashThread();***

***sp.start();***

***}***

***class SplashThread extends Thread***

***{***

***public void run()***

***{***

***int count=1;***

***Random r=new Random();***

***while(jProgressBar1.getValue()<jProgressBar1.getMaximum()){***

***try***

***{***

***jProgressBar1.setValue(count);***

***Thread.sleep(1200);***

***count=count+r.nextInt(100);***

***}***

***catch(InterruptedException e)***

***{***

***JOptionPane.showMessageDialog(null, "Exception In Thread:"+e,"Error!",JOptionPane.ERROR\_MESSAGE);***

***}***

***}***

***dispose();***

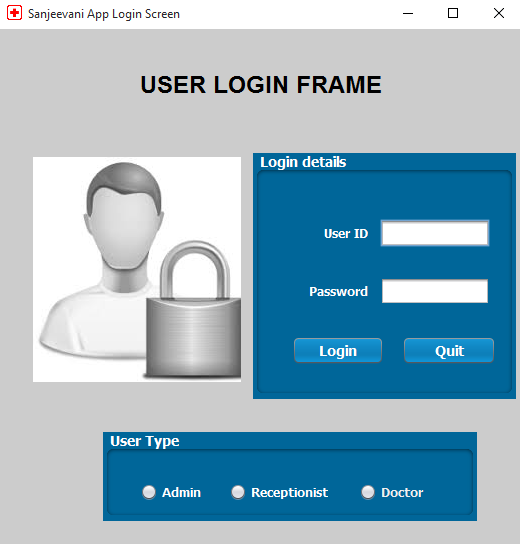
***LoginFrame loginFrame=new LoginFrame();***

***loginFrame.setVisible(true);***

***}***

***}***

**The LoginFrame**



**THE TABLES USED IN LoginFrame**

**1. EMPLOYEES**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| **EMPID** | **Varchar2(20)** | **Contains Id Of Ihe Employee** |
| **ENAME** | **Varchar2(45)** | **Contains The Employee Name** |
| **ROLE** | **Varchar2(20)** | **Contains Job Title** |
| **SAL** | **Number(7)** | **Contains Salary Of The Employee** |

**2. USERS**

|  |  |  |
| --- | --- | --- |
| **Column Name** | **Data Type** | **Description** |
| **USERID** | **Varchar2(20)** | **Contains Login Id Of the Employee** |
| **USERNAME** | **Varchar2(45)** | **Contains The Employee Name** |
| **EMPID** | **Varchar2(20)** | **Contains Id Of The Employee .This column is the Foreign Key of employee table.** |
| **PASSWORD** | **Varchar2(25)** | **Contains Password** |
| **USERTYPE** | **Varchar2(20)** | **Contains "Admin" or "Receptionist" or “Doctor”** |

**THE POJO CLASSES USED IN LoginFrame**

1. The **User** POJO

2. The **UserProfile** POJO

**CODE FOR "User" POJO**

***public class User {***

***private String userId;***

***private String password;***

***private String userType;***

***public String getUserId() {***

***return userId;***

***}***

***public void setUserId(String userId) {***

***this.userId = userId;***

***}***

***public String getPassword() {***

***return password;***

***}***

***public void setPassword(String password) {***

***this.password = password;***

***}***

***public String getUserType() {***

***return userType;***

***}***

***public void setUserType(String userType) {***

***this.userType = userType;***

***}***

***}***

**CODE FOR "UserProfile" POJO**

***public class UserProfile {***

***private static String username;***

***private static String usertype;***

***public static String getUsertype() {***

***return usertype;***

***}***

***public static void setUsername(String username) {***

***UserProfile.username = username;***

***}***

***public static void setUsertype(String usertype) {***

***UserProfile.usertype = usertype;***

***}***

***public static String getUSername(){***

***return username;***

***}***

***}***

**THE DAO CLASSES USED IN LoginFrame**

1. The **UserDao**

**HOW TO VERIFY USERID/PASSWORD**

To do this we need to create a method in **UserDao** called **validateUser()** which will accept a **User POJO** object as argument **,** search the **USER** TABLE for the given **User ID,Password** and **Type**  and return the **USERNAME**

Following is the prototype of this method:

**public static String validateUser(User user)throws SQLException**

Following are it's steps:

**a. It will accept an User POJO object as argument containing all the fields of data**

**b. It will get a Connection object from DBConnection class using the method getConnection( )**

**c. It will then frame a SELECT query for retrieving USERNAME with placeholders for the given USERID ,PASSWORD and USERTYPE**

**d. It will then create a PreparedStatement object passing it the SELECT query and use setters of the PreparedStatement to replace question marks with actual values of USER POJO**

**e. Then it will execute the query by calling the method executeQuery( ) of PreparedStatement and receive the result in a ResultSet object**

**f. Now , it will check whether USERNAME is found or not . If it is found then the method will return the username otherwise it will return null**

**e. It will not handle any SQLException and will simply pass it on to it's caller**

3. Following is it's code:

***public static String validateUser(User user)throws SQLException***

***{***

***Connection conn=DBConnection.getConnection();***

***String qry="Select username from Users where userid=? and password=? and usertype=?";***

***PreparedStatement ps=conn.prepareStatement(qry);***

***ps.setString(1,user.getUserId());***

***ps.setString(2,user.getPassword());***

***ps.setString(3, user.getUserType());***

***ResultSet rs=ps.executeQuery();***

***String username=null;***

***if(rs.next())***

***{***

***username=rs.getString(1);***

***}***

***return username;***

***}***

**WRITING THE CODE FOR BUTTON Button.png IN LoginFrame**

1. When the user will click the **Login** button then it will verify the login details and accordingly open the next screen

2. Following are it's important points:

**a. It will first validate whether all the data has been properly filled all or not.**

**b. If not , then it will generate an error message and return.**

**c. Then it will verify whether "Admin" or "Cashier" option has been selected or not.**

**d. If not , then it will generate an error message and return.**

**e. Otherwise , it will create an User pojo object , fill all the values in it and pass it to the method validateUser( ) of the UserDao.**

**f. If the method validateUser( ) returned null then it will display an error message.**

**g. Otherwise , it will store username and usertype as static fields in another class called UserProfile. This class will be used throughout the app to display the username on every frame.**

**h. Then if the user is Admin it will open the AdminOptionsFrame , if the user is Receptionist , it will open the ReceptionistOptionsFrame and if the user is Doctor then it will open the DoctorOptionsFrame.**

**i. It will also handle any SQLException that will be thrown by the method validateUser( )**