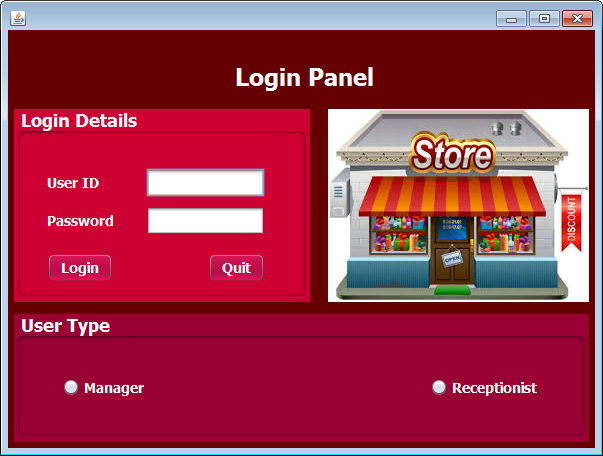
**The LoginFrame**

****

**THE TABLE USED IN LoginFrame**

**USERS:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Column Name** | | **Data Type** | **Description** |
| **USERID** | **Varchar2(10)** | | **Contains id of the user** |
| **EMPID** | **Varchar2(10)** | | **Contains the employees id** |
| **PASSWORD** | **Varchar2(10)** | | **Contains the Password** |
| **USERTYPE** | **Varchar2(10)** | | **Contains the type of user ‘Manager’ or ‘Receptionist’** |
| **USERNAME** | **Varchar2(10)** | | **Contains username of the user** |

**THE POJO CLASSES USED IN LoginFrame**

1. The **UserPojo** POJO

2. The **UserProfile** POJO

**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 **UserPojo** object as argument,search the **USER** TABLE for the given **UserID, Password** and **Type**  and return **true** or **false** , based upon whether the user is found or not.

Following is the prototype of this method:

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

Following are it's steps:

**a. It will accept an UserPojo 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 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 UserPojo object.**

**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 true otherwise it will return false.**

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

**WRITING THE CODE FOR BUTTON 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 "Manager" or "Receptionist" option has been selected or not.**

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

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

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

**g. Otherwise , it will also 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 Manager it will open the ManagerOptionsFrame and if the user is Receptionist , it will open the ReceptionistOptionsFrame**

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