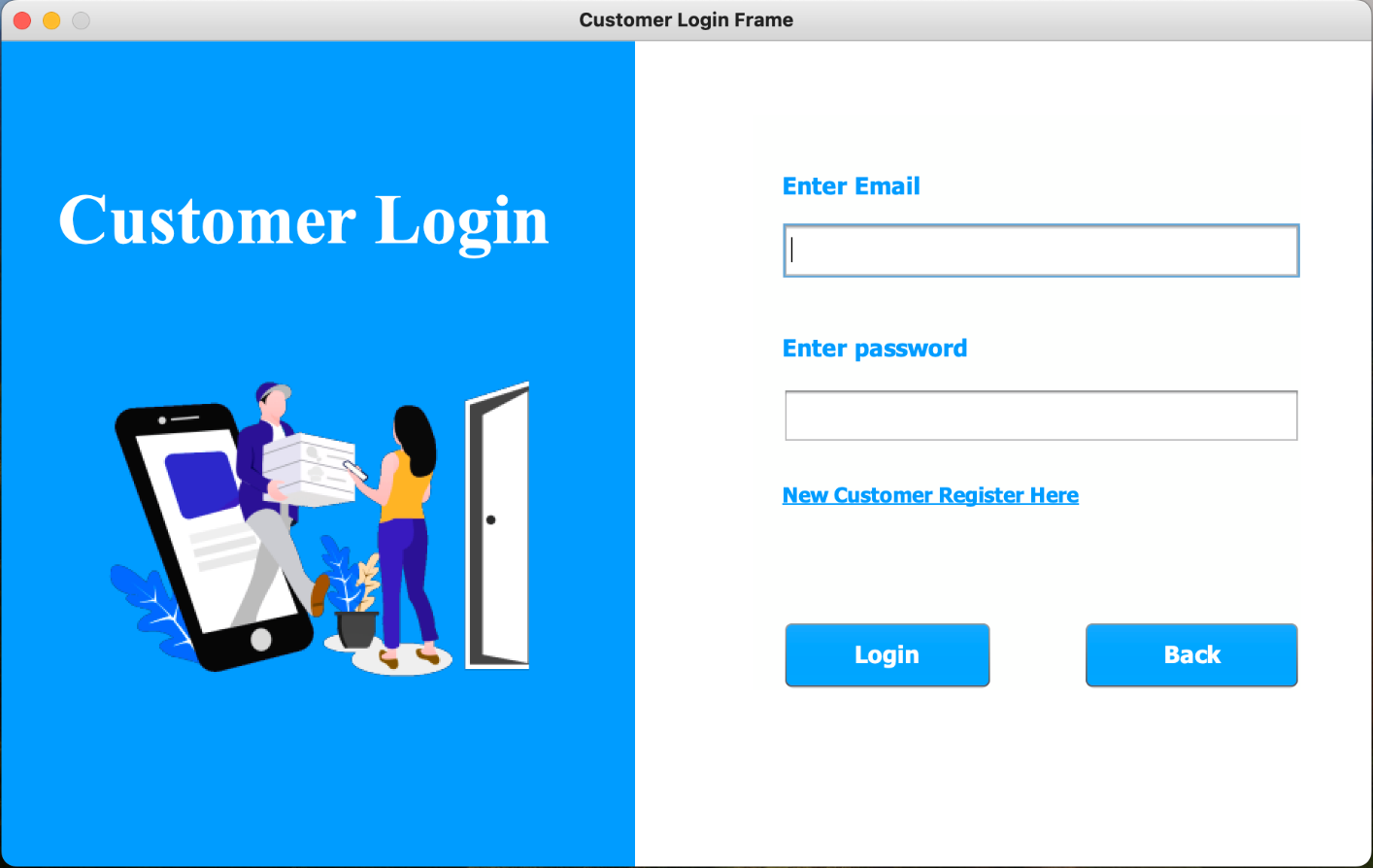
**Java SE Project**

YumXpress Application

***Customer Login Frame Documentation***

1. **Customer Login Frame –**



**STEPS TO BE DONE IN** **CustomerLoginFrame**

1. **Creating UserProfile POJO class**
2. In the **CustomerLoginFrame** we need define following methods:
3. **Even handler for New Customer Register**
4. **validateInputs()**
5. **Event handler for Login button**
6. **validateCustomer() in the CustomerDao class**
7. **Event handler for Back button**

**Creating UserProfile**

**public class UserProfile {**

**private static String userName;**

**private static String userId;**

**private static String emailId;**

**public static String getUserName() {**

**return userName;**

**}**

**public static void setUserName(String userName) {**

**UserProfile.userName = userName;**

**}**

**public static String getUserId() {**

**return userId;**

**}**

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

**UserProfile.userId = userId;**

**}**

**public static String getEmailId() {**

**return emailId;**

**}**

**public static void setEmailId(String emailId) {**

**UserProfile.emailId = emailId;**

**}**

**}**

**Coding for validateSeller() method of CompanyDao class :**

**public static CustomerPojo validateCustomer(String emailId, String password) throws SQLException {**

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

**String qry = "SELECT customer\_id, customer\_name FROM customers WHERE email\_id = ? AND password = ?";**

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

**ps.setString(1, emailId);**

**ps.setString(2, password);**

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

**CustomerPojo customer = null;**

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

**customer = new Customer();**

**customer.setEmailId(emailId);**

**customer.setCustomerId(rs.getString("customer\_id"));**

**customer.setCustomerName(rs.getString("customer\_name"));**

**}**

**return customer;**

**}**

**Coding for New Customer Register**

**private void lblRegisterMouseClicked(java.awt.event.MouseEvent evt) {**

**showFrame = new CustomerRegisterFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**}**

**Concept for Login Button :**

We have to first **validate** all the **inputs** for that we will have the **validateInput() metho**.After **validating** the customer, we will check the given **emailId** and **password** in the **database**. If the customer does not exist then we will generate the dialog box with the error **"Login Denied"** followed by the error message **"Invalid id or password! Try Again"** or else we will get customer details from the database like **emailId**, **customerId** and **customerName** and will store in the **UserProfile** POJO and then we will open **CustomerOptionFrame** and dispose this frame.

**Coding for Login Button :**

**private void btnLoginActionPerformed(java.awt.event.ActionEvent evt) {**

**if (!validateInput()) {**

**JOptionPane.showMessageDialog(this, "Please Fill All Fields!", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**String emailId = txtEmailId.getText().trim().toUpperCase();**

**char[] password = txtPassword.getPassword();**

**String pwd = PasswordEncryption.getEncryptedPassword(String.valueOf(password).trim());**

**System.out.println("emial" + emailId);**

**System.out.println("pass" + pwd);**

**try {**

**CustomerPojo customer = UserDao.validateCustomer(emailId, pwd);**

**if (customer == null) {**

**System.out.println("customer is not valid.");**

**JOptionPane.showMessageDialog(null, "Invalid Email Id or Password!", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**UserProfile.setEmailId(result.get("emailId"));**

**UserProfile.setUserName(result.get("userName"));**

**UserProfile.setUserId(result.get("customerId"));**

**showFrame = new CustomerOptionFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**}**

**}**

**Coding for validateInput() method :**

**private boolean validateInput() {**

**if (txtEmailId.getText().trim().equals("") || String.valueOf(txtPassword.getPassword()).trim().equals("")) {**

**return false;**

**}**

**return true;**

**}**

**Concept for Back Button :**

On the click of back button, we will redirect to the previous Frame i.e. UserVerifyFrame and dispose this Frame.

**Coding for Back Button :**

**private void btnBackActionPerformed(java.awt.event.ActionEvent evt) {**

**showFrame=new UserVerifyFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**}**