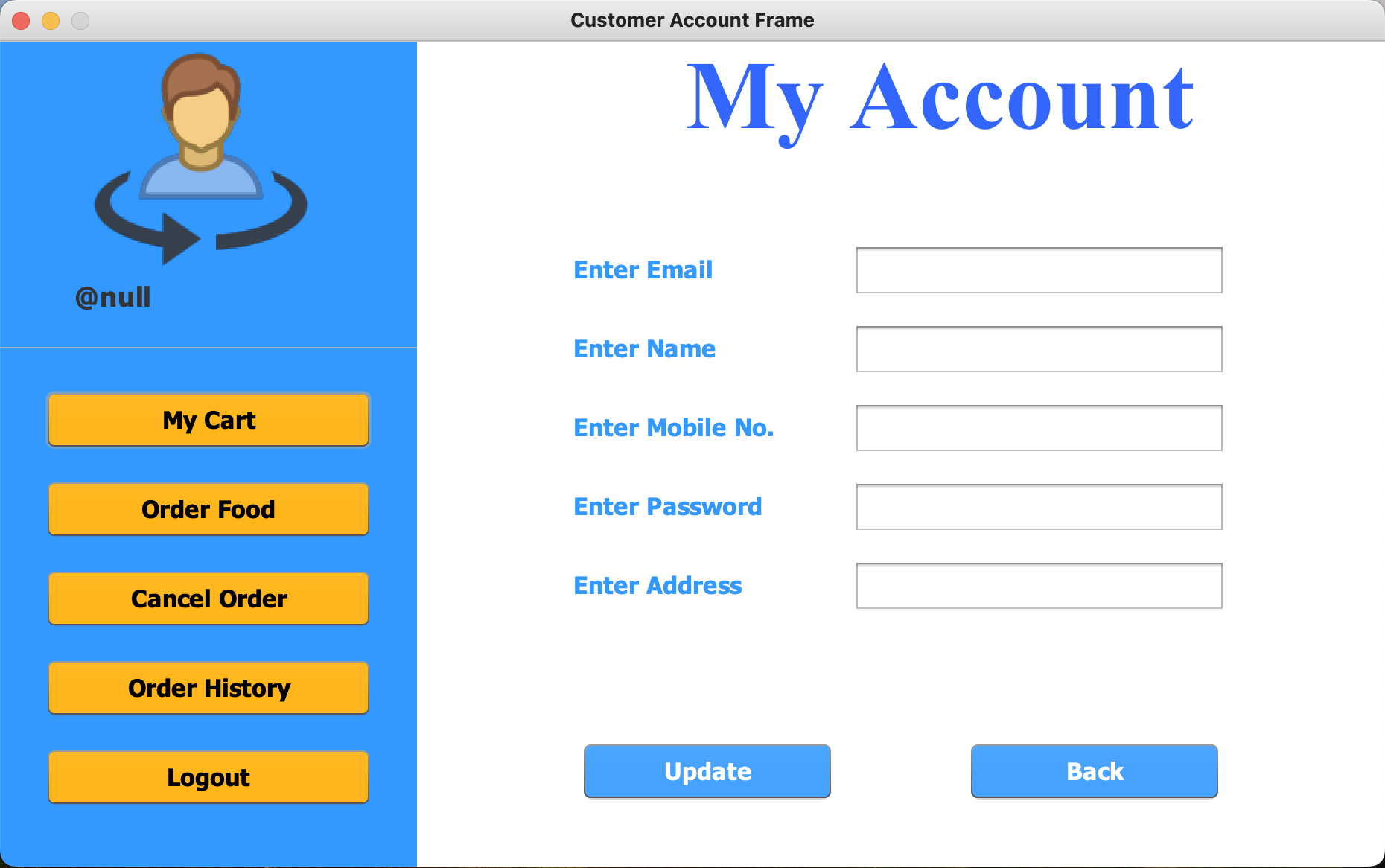
**Java SE Project**

YumXpress Application

***CustomerAccountFrame Documentation***

1. **CustomerAccountFrame –**

****

**STEPS TO BE DONE IN** **CustomerAccountFrame**

1. In the **CustomerDAO** we need define the method **getCustomerDetailsByCustomerId()** which accepts a **customer id** as argument and returns complete details of that customer as **CustomerPojo**.
2. **Even handler for New Customer Register**
3. **validateInputs()**
4. **Event handler for Login button**
5. **validateCustomer() in the CustomerDao class**
6. **Event handler for Back button**

**Coding for getCustomerDetailsByCustomerId() method of CustomerDAO class :**

**public static CustomerPojo getCustomerDetailsByCustomerId(String customerId) throws SQLException {**

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

**String qry = "SELECT customer\_name, mobile\_no, password, address, email\_id FROM customers WHERE c.customer\_id = ?";**

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

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

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

**CustomerPojo customer = null;**

**if (result.next()) {**

**customer = new CustomerPojo();**

**customer.setCustomerId(customerId);**

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

**customer.setMobileNo(result.getString("mobile\_no"));**

**customer.setPassword(result.getString("password"));**

**customer.setAddress(result.getString("address"));**

**customer.setEmailId(result.getString("email\_id"));**

**}**

**return customer;**

**}**

**Coding for updateCustomer() method of CustomerDAO class :**

**public static boolean updateCustomer(CustomerPojo customer) throws SQLException {**

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

**conn.setAutoCommit(false);**

**PreparedStatement ps = conn.prepareStatement("UPDATE customers SET customer\_name = ?, address = ?, password=?,mobile\_no=? where customer\_id = ?");**

**ps.setString(1, customer.getCustomerName());**

**ps.setString(2, customer.getAddress());**

**ps.setString(3, customer.getPassword());**

**ps.setString(4, customer.getMobileNo());**

**ps.setString(5, customer.getCustomerId());**

**return ps.executeUpdate()==1;**

**}**

**Coding for CustomerAccountFrame Constructor :**

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

**private JFrame showFrame;**

**private CustomerPojo customer;**

**public CustomerAccountFrame() {**

**initComponents();**

**this.setLocationRelativeTo(null);**

**lblUserProfile.setText("@"+UserProfile.getUserName());**

**loadCustomerDetailsById(UserProfile.getUserId());**

**}**

**Coding for loadCustomerDetailsById() Method :**

**private void loadCustomerDetailsByCustomerId(String customerId) {**

**try {**

**customer = CustomerDao.getCustomerDetailsByCustomerId(customerId);**

**txtEmail.setText(customer.getEmailId());**

**txtName.setText(customer.getCustomerName());**

**txtMobileNo.setText(customer.getMobileNo());**

**txtPassword.setText(PasswordEncryption.getDecryptedPassword(customer.getPassword()));**

**txtAddress.setText(customer.getAddress());**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**}**

**}**

**Concept for Update Button :**

We have to first **validate** all the **inputs** for that we will have the **validateInput() method**. After **validating** the customer, we will create a **CustomerPojo** object and initialize it with all the data entered by the user . Finally we will call the method **updateCustomer()** , passing it the **CustomerPojo** object for updating the record in the Database.

**Coding for validateInput() method :**

**private boolean validateInput() {**

**if (txtAddress.getText().trim().equals("") || txtName.getText().trim().equals("") || txtMobileNo.getText().trim().equals("") || String.valueOf(txtPassword.getPassword()).trim().equals("")) {**

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

**return false;**

**}**

**if (!Validator.isValidMobile(txtMobileNo.getText().trim())) {**

**JOptionPane.showMessageDialog(null, "Please input a valid mobile number!", "Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**}**

**if (!Validator.isValidEmail(txtEmail.getText().trim())) {**

**JOptionPane.showMessageDialog(null, "Please input a valid Email Id!", "Error", JOptionPane.ERROR\_MESSAGE);**

**return false;**

**}**

**return true;**

**}**

**Coding for Update Button :**

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

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

**return;**

**}**

**CustomerPojo customer = new CustomerPojo();**

**customer.setCustomerId(UserCredentials.getUserId());**

**customer.setEmailId(txtEmail.getText().trim());**

**customer.setCustomerName(txtName.getText().trim().toUpperCase());**

**customer.setMobileNo(txtMobileNo.getText().trim());**

**customer.setPassword(PasswordEncryption.getEncryptedPassword(String.valueOf(txtPassword.getPassword()).trim()));**

**customer.setAddress(txtAddress.getText().trim().toUpperCase());**

**try {**

**boolean isCustomerUpdated = CustomerDao.updateCustomer(customer);**

**if(!isCustomerUpdated) {**

**JOptionPane.showMessageDialog(null, "Can't Update Details!\nPlease Try Again Later", "Error", JOptionPane.ERROR\_MESSAGE);**

**return;**

**}**

**JOptionPane.showMessageDialog(null, "Details Updated Successfully.", "Success", JOptionPane.INFORMATION\_MESSAGE);**

**showFrame = new CustomerOptionFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**} catch (SQLException ex) {**

**ex.printStackTrace();**

**}**

**}**

**Concept for Back Button :**

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

**Coding for Back Button :**

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

**showFrame=new CustomerOptionFrame();**

**showFrame.setVisible(true);**

**this.dispose();**

**}**