Contents

[1. Logo Screen 3](#_Toc439543824)

[2. Login Signup Screen 3](#_Toc439543825)

[2.1 Forgot Password OTP: 4](#_Toc439543826)

[2.2 Forgot Password New Password: 5](#_Toc439543827)

[3. Sign up Screen: 5](#_Toc439543828)

[3.1 Retailer/Customer 6](#_Toc439543829)

[4. Buyer Home 7](#_Toc439543830)

[4.1 Products detailed 9](#_Toc439543831)

[5. SplitActionBar 10](#_Toc439543832)

[5.1 Cart 10](#_Toc439543833)

[5.1.1 Cart Home 11](#_Toc439543834)

[5.1.2 Cart OTP 11](#_Toc439543835)

[5.1.3 Cart Order Status 12](#_Toc439543836)

[5.2 Chat 13](#_Toc439543837)

[5.3 Settings 13](#_Toc439543838)

[5.3.1 About Distributor 14](#_Toc439543839)

[5.3.2 Update Profile 14](#_Toc439543840)

[5.3.3 Privacy Policy: 16](#_Toc439543841)

[5.3.4 Terms and Conditions: 16](#_Toc439543842)

[5.3.4 Logout 16](#_Toc439543843)

[5.3.6 About Developer 17](#_Toc439543844)

[5.4 Feedback 17](#_Toc439543845)

[5.5 Automated Call 18](#_Toc439543846)

[5.6 Home 18](#_Toc439543847)

[6. Info\_store 18](#_Toc439543848)

[6.1 Methods: 19](#_Toc439543849)

[7. OTP\_Handler 19](#_Toc439543850)

[7.1 Methods: 19](#_Toc439543851)

[8. Webservice\_Handler 20](#_Toc439543852)

[8.1 Methods: 21](#_Toc439543853)

[9. CartOperation 22](#_Toc439543854)

[9.1 Methods 22](#_Toc439543855)

[10. sUtility 24](#_Toc439543856)

[10.1 Methods 24](#_Toc439543857)

[11. fileHandler 27](#_Toc439543858)

[11.1 Methods 27](#_Toc439543859)

[12. XMLHandler 28](#_Toc439543860)

[12.1 Methods 28](#_Toc439543861)

[13.Item 29](#_Toc439543862)

# Logo Screen

**Type:** Activity

**Sequence:** 1

**Description:** Will display the logo of the customer for 5 seconds.

**Startup Functionality:**

This activity should call the method **isLoginAvailable()** of **sUtility** class. If it returns true, then skip **Login SIgnupScreen** to go directly to “**Buyer Home**” activity. Else it should go to **Login Signup Screen.**

# Login Signup Screen

**Type:** Activity

**Sequence:** 2

**Description:** Login signup screen will allow the user to input Phonenumber/TIN and password to log in to the application.

**Elements:**

1. Editbox1 –TIN/Phonenumber
2. Editbox2 - Password
3. Button1 – Login
4. Button2 - Signup
5. Button3 – Forgot password

**Requirements:**

1. **Editbox1:**

This will have a length of 10 digits. It will only get NUMERIC characters. It will automatically invoke number keypad when focused. Upon double tapping, the contents of the Editbox will be cleared.

1. **Editbox2:**

This will have a minimum length of 6 characters and a max length of 50 characters, characters can be alphanumeric. Upon double tapping, the contents of the Editbox will be cleared.

1. **Button1:**

When login button is clicked, it should send the content of Editbox1 and Editbox2 to Webservice and if the result is Success it should invoke the next activity “**Buyer home**” It will update the **Info\_Store** class with information obtained from **Accounts** table. Also if the **acc\_user\_info** details are not available in Info\_Store then it will be obtained and saved locally by calling **save\_details\_toLocal()** method of **sUtility** class.

If the result is Fail, it should prompt that “login failed” and sit on the same screen.

1. **Button2:**

When signup button is clicked, it should start a new activity “**Signup Screen**”

1. **Button3:**

When forgot password button is clicked it should start a new activity “**forgot password**” .

## Forgot Password OTP:

**Type:** Activity

**Sequence :** 2.1

**Description:** This activity will allow user to reset any lost password .

**Startup Functionality:**

Upon invoking this activity, it should call **generate\_OTP()** method of **OTP\_Handler** class. Then it should call the **Send\_OTP()** method of **OTP\_Handler** class.

**Elements:**

1. Label1-Desc
2. Editbox1-OTP
3. Button1-Reset
4. Button2-Resend OTP

**Requirements:**

1. **Label1:**

Label1 will display the following message in Center text format.

“We have sent a reset code in to your mail. Please enter it below.”

1. **Editbox1:**

Editbox1 will have 10 character max length, and the characters can be alphanumeric. Upon double tapping the Editbox1, the content of Editbox1 will be cleared.

1. **Button1:**

Clicking the Button1 will send the content of the Editbox1 to **compare\_OTP** method of **OTP\_Handler** class. If the return is TRUE, then it will open the fragment “Forgot password new password”

If the return is False, a Toast message will appear saying “Please enter correct OTP or click Resend OTP”

1. **Button2:**

Clicking the Button2 will call the **generate\_OTP()** method **of OTP\_Handler** class and then call **Send\_OTP()** method of the same class .

A toast message will be displayed stating “OTP sent to mail”

## Forgot Password New Password:

**Type:** Fragment

**Sequence:** 2.2

**Description:** This fragment allows the user to enter new password and set it.

**Elements:**

1. Editbox1 – New password
2. Editbox2- Retype password
3. Label1 - Status
4. Button1 – Submit

**Requirements:**

1. **Editbox1:**

Editbox1 will have a minimum of 6 characters and a maximum of 50 characters. The characters can be alphanumeric. Upon double tapping the Editbox1, the content of Editbox1 should be cleared.

1. **Editbox2:**

Same as Editbox1.

1. **Label1:**

Upon entering the Editbox2, for each character typed, the content of Editbox2 is matched with the content of Editbox1 and if they are same, the Label1 will display “Matches” in green. If not the Label1 will show “Does not match” in red.

1. **Button1:**

Upon clicking on Submit button, it should verify if status is “Matches”. If the status is “matches”, the content of Editbox1 along with the accounted to the method **send\_new\_password()** of **Webservice\_Handler** Class. If the method **send\_new\_password()** returns true it will start a new activity “**Login Signup Screen**”. Before calling **send\_new\_password()** method, progress dialog is invoked and ends when the function returns a value.

# Sign up Screen:

**Type:** Activity

**Sequence:** 3

**Description:** This screen allows user to sign up with their details to create accounts in order to use the application. This Activity has two Fragments- **Customer and Retailer**.

**Startup Functionality:**

Upon invoking this activity, RadioButton1 will be checked by default thereby displaying **Fragment1-Customer.**

**Common Elements:**

1. Radio Button 1- Customer
2. Radio Button 2- Retailer
3. Button1- Signup
4. Fragment1 – Customer/Retailer

**Requirements:**

1. **RadioButton1:**

RadioButton1 will be checked by default. Upon clicking it , **Fragment1** will be displayed with Editbox4(TIN) disabled/hidden.

1. **RadioButton2:**

When RadioButton2 is clicked **Fragment1** will be loaded and displayed.

1. **Button1:**

When Button 1 is clicked, it will obtain all the information from the elements of **Fragment 1** and the status of RadioButton1 and RadioButton2 Pass it to the method **signup\_FormSend()** of **Webservice\_Handler** class.

It will start a progress Dialog before calling the **signup\_FormSend()** method. The progress Dialog will end when the method returns Either TRUE or FALSE. If the above method returns TRUE, then a toast message will be displayed stating ”Signup Successful”, else “Signup failed! Please try again”.

1. **Fragment1:**

See 3.1

## Retailer/Customer

**Type:** Fragment

**Sequence:** 3.1

**Description:** **Retailer Fragment** will display a group of Editboxes depending on the Radiobutton checked.

**Elements:**

1. Editbox1 – Firm Name
2. Editbox2 - Address
3. Editbox 3 – PINcode
4. Editbox4 – TIN
5. Editbox5 – Password
6. Editbox6 – Phone number
7. Editbox7 – Email

**Requirements:**

1. **Editbox1**  
   Editbox1 should have a maximum character of 50. Characters must be Alphabets. Upon double tapping on Editbox, it’s contents should be cleared.
2. **Editbox2**

Once clicked or focused, it should start a new activity “**update profile**” with mobile number Editbox and PIN code Editbox disabled or hidden. Once the user finishes entering all the information. The information is stored on **Info\_Store** class once update button is clicked.

Once the update button is clicked, the EditBox2 is updated with the information stored in **Info\_Store** class.

1. **Editbox3**

Editbox3 has maximum character length of 6 and a minimum character length of 6. It should only contain Numeric characters.

1. **Editbox4**

Editbox4 has maximum character length of 10 and a minimum character length of 10. It should only contain Numeric characters.

1. **Editbox5**

Editbox5 has maximum character length of 50 and minimum character length of 6. It can have alphanumeric characters.

1. **Editbox6**

Same as Editbox4.

1. **Editbox7**

Editbox7 has maximum character length of 50 and minimum character length of 7. It can have alphanumeric characters.

# Buyer Home

**Type:** Activity

**Sequence:** 4

**Description:** The Home screen of the application. This is where the user can browse for the brands, category and products.

**Elements:**

1. Editbox1 – Search Field
2. Listbox1 – Criteria
3. Listbox2 – Criteria value
4. Button1 – Search
5. Listview – Product,Categories,Brands
   1. View1-Brands, Categories
      1. ImageView1-Brand/Category image
      2. Label1 – Brand /Category name
      3. ImageView2 – Right Arrow
   2. View2- Products
      1. ImageView1-Product Image
      2. Label1- Title
      3. Label2 - Price
      4. ImageView2(clickable) –Add to cart
6. Fragment1-Home
7. Fragment2-Products

**Requirements:**

1. **Editbox1**

The Editbox1 will have a max character of 50 and could contain alphanumeric characters.

1. **Listbox1**

Listbox1 will contain three items: Price, capacity and Usage.

1. **Listbox2**

Listbox2 will contain items depending on the selected value of Listbox1.

1. **Button1**

When Button1 is clicked, it will first determine whether the Editbox1 is empty or not. If it is empty then It will check whether the Listbox1 has a value. If Listbox1 has a value, then it will check if Listbox2 has a value, If it has, then It will call **Search\_query\_Constructor()** method of **sUtility** Class. Before calling the above method, a progress dialog is invoked. The progress dialog is closed when the method returns true. The method will update the Listview data in **Info\_Store.** Update the Listview with the data from **Info\_store**.

1. **ListView**

When the Activity is called, ListView is populated with the Brand Info, when any of the listItems are clicked, Categories for that Brand is displayed on the same listview. When a category list item is clicked, the same listview is updated with Listitems related to Product. The listitem will have two types of layout, one for brands and categories and another for products.

Upon clicking the product list item, “**Fragment2-Products**” fragment will be opened.

1. **Fragment1**

When the activity is called**, Fragment1** is loaded as default. It will contain the main elements of this activity.

1. **Fragment2**

This fragment replaces the main container of **the Buyer Home activity** when the list item is clicked during the product scope.

## Products detailed

**Type:** Fragment

**Sequence:** 4.1

**Description:** This fragment gives a detailed description and a large picture of the product.

**Startup Functionality:**

Upon invoking this fragment, The Product ID is passed to the method **fill\_productDetails()** of **sUtility** Class. This method will invoke progress Dialog before calling **fill\_productDetails()** and Close the dialog when **fill\_productDetails()** returns true.

**Elements:**

1. ImageView1
2. Label1
3. Label2
4. Editbox1
5. Button1

**Requirements:**

1. **ImageView1**

ImageView displays the Image of the Product being displayed. The image is fetched from the Info\_store.

1. **Label1**

Label1 will display the detailed description

1. **Label2**

Label2 will display the price of the item

1. **Editbox1**

Editbox1 will have max of 999 as value and minimum of 1 as value. It will always be Numeric character. Upon double tapping the Editbox1, it’s content should be cleared.

1. **Button1**

When Button1 is clicked, it should add the product ID and content of editbox in to the cart implementation singleton class Cart. And returns to the Fragment1.

# 5. SplitActionBar

**Type:** ActionBar

**Sequence:** 4

**Description:** This actionbar has various shortcuts leading to different activity. One of the Main purpose is to make avail the cart button.

**Elements:**

1. Home Button
2. Chat Button
3. Settings
4. Feedback
5. Automated Call
6. Cart

**Requirements:**

1. **Home Button**

Home button should call the “**BuyerHome**” Activity when clicked.

1. **Chat Button**

Chat button should call the “**chat**” Activity when clicked.

1. **Settings Button**

Settings button should call the “**Settings**” Activity when clicked.

1. **Feedback Button**

Feedback Button should call the “**Feedback**” Activity when clicked.

1. **Automated Call Button**

It should call the “**Call**” the number of the Arihant when clicked.

1. **Cart Button**

It should invoke “**Cart**” activity when clicked.

## 5.1 Cart

**Type:** ActionBarItem & Activity

## 5.1.1 Cart Home

**Sequence :** 0.1

**Description:** Allows user to check which products they have added to cart and gives a brief summary of order.

**Elements:**

1. ListView – Cart Items
   1. ImageView-Picture
   2. Label1-Title
   3. Label2-Price
   4. Label3-Quantity
   5. Button1-Edit
   6. Button2-Remove
2. View- Order Details
   1. Label1-DeliverBy
   2. Editbox1-DeliveryDate
   3. Label2-Sub total
   4. Label3-Tax
   5. Label4-total
3. Button1-Checkout

**Requirements:**

1. **ListView**

ListView contains the items the user has chosen to checkout. It reflects the singleton class “**Cart**” which implements the cart functionality. It has one view. It has six inner elements . When remove button of one of its items are pressed, the index of the element is obtained and that item is removed from the “**cart**” class and **setnotifydatasetchanged()** is called.

Clicking on the edit button will just display a small dialog with Editbox and a label saying “Change quantity to “. That Editbox will have minimum value of 1 and maximum value of 999.

1. **View**

This is a simple rectangle which will contain the information about order. There will be a editbox1 or **Date Selector** which will be used to obtain the delivery date.

The labels will reflect the respective functions.

1. **Button1**

When Button1 is clicked, it will call “**Cart OTP**” Actvity.

## 5.1.2 Cart OTP

**Sequence :** 0.1

**Description:** Accepts OTP from User and proceeds to checkout.

**Startup Functionality:**

Upon invoking this activity, it should call **generate\_OTP()** method of **OTP\_Handler** class. Then it should call the **Send\_OTP()** method of **OTP\_Handler** class.

**Elements:**

1. Label1
2. Editbox1
3. Button1
4. Button2

**Requirements:**

1. **Label1**

Label1 will display the following message “We have sent you an One Time Password. Please enter it to confirm your order”

1. **Editbox1**

Editbox1 will have maximum length of 4 digit. It will only contain alphanumeric characters.

1. **Button1:**

Clicking the Button1 will send the content of the Editbox1 to **compare\_OTP** method of **OTP\_Handler** class. If the return is TRUE, then it will open the fragment “Order Status”

If the return is False, a Toast message will appear saying “Please enter correct OTP or click Resend OTP”

1. **Button2:**

Clicking the Button2 will call the **generate\_OTP()** method of **OTP\_Handler** class and then call **Send\_OTP()** method of the same class .

A toast message will be displayed stating “OTP sent to your number”

## 5.1.3 Cart Order Status

**Sequence :** 0.3

**Description:** Displays order status of the screen.

**Startup Functionality:**

Upon invoking this activity, it should call compute the order details and call **create\_orders()** method of **sUtility** class.It should first invoke **progressDialog** before calling **create\_orders().** When the **create\_orders()** method returns a value the **progressDialog** will exit and based on the return value, the order status will be displayed.

If the return value is true, then the message “Order placed successfully” in **green** text

Else if the retun value is false, the message “Order not placed! Please try again” **in red text**.

**Elements:**

1. Label1

**Requirements:**

1. **Label1**

Label1 will be updated with Order status.

## 5.2 Chat

**Type:** ActionBarItem & Activity

**Sequence :** 0.2

**Description:** Will allow the Users to chat with Arihant Marketing .

**Elements:**

1. ListView1 – Chat Screen
2. Editbox1 – Type to chat
3. Button1- Send

**Requirements:**

1. **ListView1**

This listview displays the chat contents between the user and Arihant Marketing.

It gets its content from the local database. It has limit of 50 messages. It will only display the latest 50 messages.

1. **Editbox1**

Editbox1 can have any type of characters and has a max length of 150characters.

1. **Button1**

Upon clicking Button1, The content of Editbox1 will be added to ListView as new Item, it will be added to the local database. It will then be passed along with AccountID to **chat()** method of **Webservice\_handler** class. The method returns true if the message is commited to the **database**. Once the value is return the message which is added as listview item is marked sent.

## 5.3 Settings

**Type:** ActionBarItem & Activity

**Sequence :** 0.3

**Description:** Displays Settings Activity.

**Elements:**

1. ListView1
   1. About Us
   2. Update Profile
   3. Privacy Policy
   4. Terms and conditions
   5. Logout
   6. About Developer

**Requirements:**

1. **ListView1**

The ListView will list the above items in the Elements section. Upon clicking on these items, respective Fragments and Activities will be started.

## 5.3.1 About Distributor

**Type:** Fragment

**Sequence :** 0.3.1

**Description:** Displays Information about Distributor.

**Elements:**

1. ImageView1
2. Label1

**Requirements:**

1. **ImageView1:**

ImageView1 will contain the Company Logo

1. **Label1**

Label1 will contain the company Description.

## 5.3.2 Update Profile

**Type:** Activity

**Sequence :** 0.3.2

**Description:** Used to update information about customer/retailer.

**Startup Functionality:**

The Elements displayed on this form varies according to Scope.

#### 5.3.2.1 Scope – Settings

All elements will be displayed. The elements will be pre-populated from the data available at **Info\_Store** Class.

#### 5.3.2.2 Scope – Signup Form

All elements except EditText5 and EditText6 will be displayed.

**Elements:**

1. Editbox1 –Door/Appartment
2. Editbox2-Street2
3. Editbox3-Area
4. EditBox4-City
5. Listbox1-District
6. Listbox2-State
7. Editbox5-PINcode
8. Editbox6-MobileNo
9. Button1-Update

**Requirements:**

1. **Editbox1**
2. **Editbox2**
3. **Editbox3**
4. **Editbox4**

The above Editboxes will have minimum character of 1 and Maximum Character of 50, They can contain alphanumeric characters.

1. **Listbox1**

Listbox1 will display the State from drop down.

1. **Listbox2**

Once Listbox1 has a value the District will be displayed from dropdown. It will remain disabled until listbox1 has value.

1. **Editbox5**

Editbox5 will have a maximum character length of 6 and minimum character length of 6.

1. **Editbox6**

Editbox6 will have a maximum character length of 10 and minimum character length of 10.

1. **Button1**

Upon clicking Button1, the contents of editboxes and listboxes will be copied to **Info\_Store** class. The progressDialog will be invoked. Then the method **Construct\_update\_detail()** of **sUtility** Class will be called. ProgressDialog will be closed once the above method returns a value. A toast message will be displayed depending upon the return value. If the return value is true then the message”Update successful” will be displayed. Else the message “ Update failed, please try again” will be displayed.

## 5.3.3 Privacy Policy:

**Type:** Fragment

**Sequence :** 0.3.3

**Description:** Displays Information about Privacy policy.

**Elements:**

1. Label1

**Requirements:**

1. **Label1**

Label1 will display the privacy policy of this app.

## 5.3.4 Terms and Conditions:

**Type:** Fragment

**Sequence :** 0.3.4

**Description:** Displays Information about Privacy policy.

**Elements:**

1. Label1

**Requirements:**

1. **Label1**

Label1 will display the terms and condition of this app.

## 5.3.4 Logout

**Type:** Dialog

**Sequence :** 0.3.4

**Description:** Logs out the user from the application

**Elements:**

1. Dialogue box

**Requirements:**

1. **Dialogue box**

Once the logout list item is clicked, it should invoke a dialogue box prompting “ Are you sure?”

With two buttons “yes” and “no” . If the user presses “yes” it will go to **Login Signup form**  else, it will remain on the same activity.

When the user clicks “yes” it should invoke the method, **clear\_Login\_data()** of **sUtility** class.

## 5.3.6 About Developer

**Type:** Fragment

**Sequence :** 0.3.6

**Description:** Displays Information about Develoepr.

**Elements:**

1. ImageView1
2. Label1

**Requirements:**

1. **ImageView1:**

ImageView1 will contain the Company Logo

1. **Label1**

Label1 will contain the company Description.

## 5.4 Feedback

**Type:** Activity

**Sequence :** 0.4

**Description:** Allows user to input feedback regarding app and Service.

**Elements:**

1. EditBox1-Content
2. RadioButton1-App
3. RadioButton2-Service
4. Button1- Submit

**Requirements:**

1. **Editbox1**

Editbox1 will contain the message. It has the maximum character length of 150 characters.

1. **RadioButton1**

When RadioButton1 is checked, the message will be sent with “App” as about.

1. **RadioButton2**

When RadioButton2 is checked, the message will be sent with “Service” as about.

1. **Button1**

When Button1 is clicked, it would invoke progressDialogue. It will then check which radiobutton is checked. Then it will call the method **Create\_Feedback()** of **sUtility** class and pass its output to the method **Non\_query()** method **of webservice\_handler** class. The progressDialog is exited when the Non\_query() method returns a value. Depending on the value the following messages are displayed. “Feedback submitted!” for true, and “Submission failed! Please try again” for false.

## 5.5 Automated Call

**Type:** ActionbarItem/Intent

**Sequence :** 0.5

**Description:** Sends intent to dialer with the number of Arihant Marketing

**Requirements:**

1. Once the Call button is pressed, it should send an intent with the phone number of Arihant marketing to the dialer.

## 5.6 Home

**Type:** ActionbarItem/Intent

**Sequence :** 0.6

**Description:** Hops to Buyer Home Activity

**Requirements:**

1. Home button should start the **Buyer Home** activity and close all the other activity.

# 6. Info\_store

**Type:** Singleton Class

**Description:**

A singleton class used to hold Context Independent Data at a central place.

**Global Members:**

1. AccountID(String)
2. Name(String)
3. Street\_1(String)
4. Street\_2(String)
5. Area(String)
6. City(String)
7. District(String)
8. State(String)
9. Pincode(String)
10. Mobile No(String)
11. Email(String)

## Methods:

#### 6.1.1 getInstance()

1. **Input Parameters:**

None

1. **Return:**

Info\_store

1. **Description:**

This static method will return the static instance of the Info\_Store class.

**Constructors: None**

**Destructors: None**

**Parent Class: None**

# 7. OTP\_Handler

**Type:** Class

**Description:**

A Class containing Commonly used OTP related Logic.

**Global Members:**

OTP\_Holder(String)

## 7.1 Methods:

#### 7.1.1 generateOTP()

1. **Input Parameters:**

None

1. **Return:**

None

1. **Description:**

This Method will generate a four digit Random Number and assigns it to Global variable OTP\_Holder

#### 7.1.2 sendOTP()

1. **Input Parameters:**

String(AccountID)

1. **Return:**

Boolean

1. **Description:**

This method will call sendOTPtoSMS() method of Webservice\_Handler. It will return TRUE for Success and FALSE for Failure. It will pass account id and OTP/content.

#### 7.1.3 compareOTP()

1. **Input Parameters:**

String(Editbox-OTP)

1. **Return:**

Boolean

1. **Description:**

This Method will compare the entered string with OTP\_Holder to check whether they match. Returns True if match, else returns False.

**Constructors: None**

**Destructors: None**

**Parent Class: None**

# 8. WebserviceHandler

**Type:** Class

**Description:**

A Class used to communicate with Web services.

**Global Members:**

* 1. SOAP\_NAMESPACE(String)
  2. SOAP\_METHOD\_NAME(String)
  3. Request(SoapObject)
  4. Pgd(ProgressDialog)
  5. C(Context)

## 8.1 Methods:

#### Send\_New\_Password()

* 1. **Input Parameters:**

String(New Password)

String(AccountID)

* 1. **Return :**

Boolean

* 1. **Description:**

This method will call the webservice **resetpass.php.** Return TRUE for successful operation, else returns FALSE.

#### Signup\_FormSend()

* 1. **Input Parameters:**

None

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **signup.php** using the information from **Info\_Store**

#### Search\_Query()

* 1. **Input Parameters:**

String(Query)

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **query.php** and returns TRUE for successful operation, else returns FALSE.

#### Non\_Query()

* 1. **Input Parameters:**

String(Query)

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **query.php** and returns TRUE for successful operation, else returns FALSE.

#### Chat()

* 1. **Input Parameters:**

String(Message)

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **Chat.php** and passes Message and AccountID from Info\_Store and returns TRUE for successful operation, else returns FALSE.

#### Insert\_Orders()

* 1. **Input Parameters:**

String(Query –Orders table)  
String(Query1-Order\_item table)

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **Commit\_orders.php** and passes Query and Query1, returns TRUE for successful operation, else returns FALSE.

#### sendOTPtoSMS()

* 1. **Input Parameters:**

String(AccountID)  
String(OTP)

* 1. **Return :**

Boolean

* 1. **Description:**

This Method will call the webservice **Send\_mail.php** and passes AccountID and OTP, returns TRUE for successful operation, else returns FALSE.

This function will call the webservice to send the OTP.

**Constructors: None**

**Destructors: None**

**Parent Class:** **AsyncTask<Object, Object, Object>**

# 9. CartOperation

**Type:**  Class

**Description:**

A Class used to store Cart contents

**Global Members:**

1.Cart (ArrayList)<Item> (Static)

## 9.1 Methods

#### 9.1.1 addtocart()

1. **Input Parameters:**

Item

1. **Return :**

None

1. **Description:**

Adds the Item in to Cart.

#### 9.1.2 removefromcart ()

1. **Input Parameters:**

Item

1. **Return :**

None

1. **Description:**

removes the Item from the Cart.

#### 9.1.3 copyCartContents ()

1. **Input Parameters:**

None

1. **Return :**

List<Item>

1. **Description:**

Returns Cart contents.

#### 9.1.4 setCart ()

1. **Input Parameters:**

List<Item>

1. **Return :**

None

1. **Description:**

Sets Content to Cart.

#### 9.1.5 chkifalrdyextsinlst ()

1. **Input Parameters:**

Item

1. **Return :**

int

1. **Description:**

Checks if the Item is already present in Cart, if yes returns the index and if no returns -1

Can be used also to get the index of the item in cart stack.

#### 9.1.7 updatecart ()

1. **Input Parameters:**

Int(position), int(Quantity)

1. **Return :**

None

1. **Description:**

Gets the position and adds the quantity to already present quantity of the Item in Cart

**Constructors: CartOperation()**

**Destructors: None**

**Parent Class: None**

# 10. sUtility

**Type:**  Class

**Description:**

A Class used to do minor operations related to this particular app

**Global Members:**

None

## 10.1 Methods

#### 10.1.1 Search\_query\_Constructor()

1. **Input Parameters:**

String(Criteria\_Name)

String[](Criteria\_Value)

1. **Return :**

Boolean

1. **Calls:**

**Search\_query()** method of **Webservice\_Handler** class

1. **Description:**

This method will create SQL Statements based on its parameter. Then pass the String to **Search\_query()** method.

#### 10.1.2 Clear\_login\_data()

1. **Input Parameters:**

None

1. **Return :**

Boolean

1. **Calls:**

**Clean\_file()** of **fileHandler** class

1. **Description:**

This method will erase all the login data from both the local file by passing **usr\_data** as parameter to the method **Clean\_file()** of **fileHandler** class and set all the login data in **Info\_store** class to " ".

#### Create\_Feedback()

1. **Input Parameters:**

String(Message Content)

String(AccountID)

String(MessageType)

1. **Return :**

String

1. **Calls:**

None

1. **Description:**

This method is used to create Insert SQL Query to be passed to a webservice.

#### isLoginAvailable()

* 1. **Input Parameters:**

None

* 1. **Return :**

Boolean

* 1. **Calls:**

**Read\_file**() of **fileHandler**, parseXML() **of XMLHandler** classes

* 1. **Description:**

Calls **read**\_**file**() method of **fileHandler** with **usr**\_**data** as parameter and passes its output to **parseXML**() method of **XMLHandler** class. If the **parseXML** Returns Empty String, then return false, else return true.

#### Fill\_productDetails()

1. **Input Parameters:**

String(productID)

1. **Return :**

Boolean

1. **Calls:**

**Search\_query()** method of  **Webservice\_Handler** class

1. **Description:**

This method will get one input parameter **ProductID** and creates a SQL Statement

**Query =Select \* from products where productID = input parameter**. It will then pass the value of **Query** to **search\_query()** method of **webservice\_handler** and obtain the results. The results is later populated to **info\_store** class. This returns **true** when the information is populated in to the **info\_store**.

#### Create\_orders()

1. **Input Parameters:**

None

1. **Return :**

Boolean

1. **Calls:**

**insert\_orders()** method of **webservice\_handler**

1. **Description:**

This method will take the information from the "Cart" Singleton class and construct SQL Queries to insert Orders in the database. **Query 1**= Insert x,y,z into **orders** **Query**+=Insert x,y,z into **order\_items**. It will then pass the two strings, **query1**, **query** in to **insert\_orders()** method of **webservice\_handler**. Return the return of **insert\_orders()**

#### Construct\_update\_detail()

1. **Input Parameters:**

None

1. **Return :**

Boolean

1. **Calls:**

**Non\_Query()**  method of  **Webservice\_Handler** class.

1. **Description:**

This method will construct a SQL query to update User info(all the contents displayed in updateProfile Screen).

Query = [UPDATE](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/update.html) `arihant`.`acc\_user\_info` [SET](http://localhost/phpmyadmin/url.php?url=http://dev.mysql.com/doc/refman/5.5/en/set.html) `Street\_1` = 'sfsd' WHERE `acc\_user\_info`.`ID` = 1;

This method will pass Query string to **NON\_query()** method of **webservice\_handler** which will return a value **TRUE** or **FALSE**, which will be return by this Method.

#### Save\_details\_toLocal()

1. **Input Parameters:**

None

1. **Return :**

Boolean

1. **Calls:**

**makeXML()** method of **XMLHandler** class, **save\_file**() method of **fileHandler** class

1. **Description:**

Call **makeXML**() method of **XMLHandler** class. Pass the output of **makeXML**() to **save\_file**() method of **fileHandler** class. With **usr\_data** as Filename.

**Constructors: None**

**Destructors: None**

**Parent Class: None**

# 11. FileHandler

**Type:**  Class

**Description:**

A Class used to do common file operations.

**Global Members:**

Context c

## 11.1 Methods

#### 11.1.1 saveToFile()

1. **Input Parameters:**

String(Filename)

String(Content)

1. **Return :**

Boolean

1. **Calls:**

None

1. **Description:**

Save the content with the spec ified file name.

#### 11.1.2 readFromFile()

1. **Input Parameters:**

String(Filename)

1. **Return :**

String(Content)

1. **Calls:**

None

1. **Description:**

Reads data from the file and returns content as a string.

#### cleanFile()

1. **Input Parameters:**

String(Filename)

1. **Return :**

Boolean

1. **Calls:**

None

1. **Description:**

Removes the content of the file.

**Constructors: None**

**Destructors: None**

**Parent Class: None**

# 12. XMLHandler

**Type:**  Class

**Description:**

A Class used to do parse and construct XML based Storage.

**Global Members:**

Context c

## 12.1 Methods

#### 12.1.1 parseXML()

1. **Input Parameters:**

String(Content)

1. **Return :**

Arraylist

1. **Calls:**

None

1. **Description:**

Parses and extracts information from XML Style document. Ps. That arraylist is used to populate info\_store class

#### 12.1.2 makeXMLfromInfoStore()

1. **Input Parameters:**

Object Info\_Store

1. **Return :**

String(Content)

1. **Calls:**

None

1. **Description:**

Constructs XML Data from the singleton class Info\_Store

#### 12.1.3 makeXML()

1. **Input Parameters:**

List<String>

1. **Return :**

String(Content)

1. **Calls:**

None

1. **Description:**

Constructs XML Data from the Arraylist<string>.

**Constructors: None**

**Destructors: None**

**Parent Class: None**

# 13.Item

**Type:**  POJO Class

**Description:**

A Class used to describe Item as an entity with name and Price and quantity as attributes.

**Global Members:**

Item\_name

Item\_Price

Item\_Quantity

**Constructors: None**

**Destructors: None**

**Parent Class: None**