



THEMEDIMENSION

IvoryShop Android
Documentation v2.3.0

CONFIDENTIAL



ABOUT

Ivory is a mobile application that shows the products sold by a shop to its clients online. The app takes a JSON feed of products and displays it to its users in an easy and interactive way, allowing them to choose their favorites products and place orders according to their needs.

Installation

In order to install the app on a physical Android OS running device you will need to locate and decompress the Apk.zip file from the base package, connect an Android device (API 19 or higher) to your computer, and drag the ***Ivory.apk*** file to the device's storage.

Once you have added the file to the mobile device internal storage, using the device, navigate to the location where you have previously copied the APK file and click **Ivory**. You will be prompted with an installation button, and after clicking it, a message that the file comes from potentially harmful sources might appear because the apk is not downloaded from the Google Play app. In order to continue with the installation, you will have to approve with the installation and let the device instal the app. This should take around 3-5 minutes, based on the device used.



Customization

How to run the project:

In order to compile the project, you will need to have a JDK installed.

If you don't have JDK installed, please download it from [here](#).

If you want to run the app directly on a device, you need to install the APK on your device. For further guidance on how to install the APK, click on this [link](#).

In order to run the project, you will need to install the [Android Studio](#) IDE.

For further instructions on how to install the IDE and a quick tutorial about it, please follow the link below:

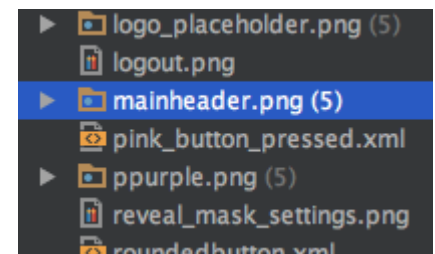
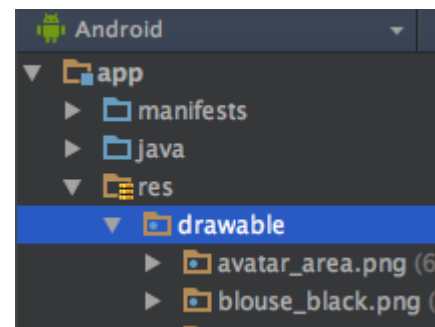
<https://developer.android.com/studio/intro/index.html>

How to change the promo image in the Main Screen:

In order to replace the promo image with one of your own, make sure to have the *Android View* selected in the project tree, and navigate to ... -> **main** -> **res** -> **drawable**.

Next, just replace the file **mainheader.png** with the desired one, but make sure to name it **mainheader**.

Note that the extension of the file can be **.jpg**, **.jpeg**, or **.png**.

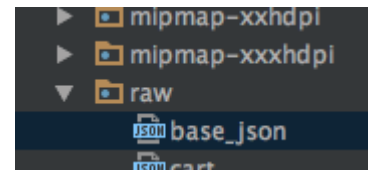


How to change the source data to be displayed by the app:

The app supports source data to be read from a json file located in the **raw** directory and it follows the JSON format ({ "productList": [jsonProduct, ..., jsonProduct] }). The data must be inserted in the file **JSON**, in the **RAW** directory, that can be found in the project tree inside the Android Studio, and it reads the list of products in a JSON format as it follows:

Product:

```
{
1  "productName": "Name Of The Product",
2  "productId": IdentificationNumber,
3  "productColor": "#CoLorX",
4  "productImage": "Product_Image_File_Name.png",
5  "productPrice": PriceOfTheProductInDecimalFormat,
6  "Sizes": [
7      {
          "id":IdentificationNumberOfTheSize,
          "name":"Size Name",
          "quantity":QuantityForThisProductSize
      },
8      {
          "id":IdentificationNumberOfTheSize,
          "name":"Size Name",
          "quantity":QuantityForThisProductSize
      },
      ...
9      {
          "id":IdentificationNumberOfTheSize,
          "name":"Size Name",
          "quantity":QuantityForThisProductSize
      }
10 ],
11 "productCategory": "The Product's Target Client",
12 "productAddedDate": "MM-DD-YYYY",
13 "productGender": "The Gender Of The Target Client"
},
...
```



```
{
  "productName":"WomenClothes",
  "productId":1,
  "productColor":"#00aabb",
  "productPrice":20,
  "sizes":
  [
    {
      "id":1,
      "name":"XS",
      "quantity":0
    },
    {
      "id":2,
      "name":"S",
      "quantity":10
    },
    {
      "id":3,
      "name":"M",
      "quantity":25
    },
    {
      "id":4,
      "name":"L",
      "quantity":15
    },
    {
      "id":5,
      "name":"XL",
      "quantity":25
    },
    {
      "id":6,
      "name":"XXL",
      "quantity":25
    }
  ],
  "productAddedDate":"05-11-2015",
  "productGender":"Women",
  "productCategory":"Clothes",
  "productImage":"blouse_blue"
}
```



Note: The example above represents a single product in **JSON** format. In order for the app to work as expected, you should follow the instructions as stated above.

Note: If you have a server that you want the app to use, please check the server-side communication documentation further down this document.

Note: In order to change all the source data at once, be sure to follow the following instructions:

1. All the data should be enclosed between `{` and `}`.

```
Ex:  {
      ...data...
    }
```

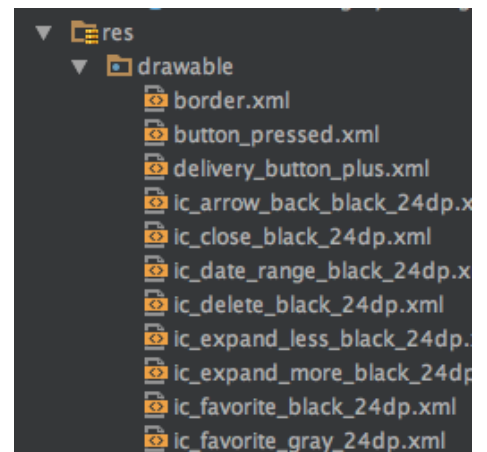
2. The first line between the `{}` symbols must be **"productList":** and immediately after it, a block of products in JSON format should follow between the square brackets `[...]`. You must be sure to close all the brackets used and state where a new product is entered, using the `,` symbol.

```
Ex:  {
      "productList":
      [
        {
          Product1InJSONFormat
        },
        {
          Product2InJSONFormat
        },
        ...
        {
          ProductNInJSONFormat
        }
      ]
    }
```

Note: Make sure that there is a `.jpg`, `.png` or `.jpeg` file with the same name available for the **"productImage"**: to be considered valid. In case not, the app will load its data successfully, but no image of the product will be available to the user.

For further information about the **JSON** format and how to use it, please follow the link below:

<http://json.org/example.html>





If you already have knowledge about the format stated above, here are some helpful links to format the data using JSON:

[JSONLint](#)

[JSON Formater & Validator](#)

[JSON Viewer](#)

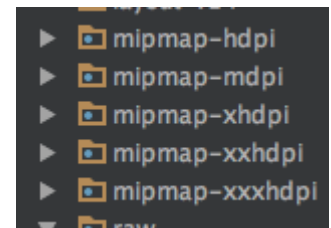
Note that any image resource the app should use must be included in the **res/drawable** directory. What is more, any icon used across the app can be found in this folder and changed accordingly just by adding a new resource to this directory.

How to change the app's icon:

Open **Ivory folder -> app -> src -> main -> res -> mipmap**

In the *mipmap* folder, replace the file with the name *app_logo.png* with a new one of an adequate size.

In case you have multiple sizes for the logo, you can delete *app_logo.png* and add your new file under the same name.



How to change the text across app:

Ivory offers great flexibility when you want to customize its elements to suit your needs. Any string used across the app can be changed from just one place in order to give its client time to deal with ideas rather than endless search.

In order to change the value of a button's text, you just have to navigate in the project's file tree to the **IvoryAndroid -> app -> src -> main -> res -> values** directory. In the *values* directory you have access to all the strings, dimensions, colors and styles used by the app.

Moreover, for a user-friendly experience, the string resources are sorted by Screens (Ex: Delivery Strings, Payment Strings etc) , Alerts and Common strings.

Note: If the user decides to change one or more strings from the **“Login”** screen he has to search in the **strings.xml** file for the **“Login”** title.

Note: If the user changes the strings in the **“Common”** section, all the strings that were common through the app will change. Example: if the string for the **“email_address_hint”** key is changed, the strings will be changed in Delivery, Login, Register, Reset Password, Settings Screens.





How to change the colors used by different elements across app:

In order to change a color across the app, the steps to follow are similar to changing a text value. The difference is that you need to navigate to the ... -> **res** -> **values** -> **colors.xml** file and change the value stored with the desired one.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 ~ Copyright © 2017 Theme Dimension, a Mobile Web America, Inc. venture
4 -->
5
6 <resources>
7
8 <!-- Cart screen color. -->
9 <color name="colorPrimary">#3F51B5</color>
10 <color name="colorPrimaryDark">#303F9F</color>
11 <color name="colorAccent">#FF4081</color>
12
13 <color name="purple">#d580ff</color>
14 <color name="red">#ff0000</color>
15 <color name="colorGreen">#2cdca0</color>
16 <color name="colorGreenDark">#21c58d</color>
17 <color name="colorPink">#a883b9</color>
18 <color name="colorPinkDark">#996dad</color>
19 <color name="testWhite">#FFFFFF</color>
20 <color name="trans">#0000</color>
21 <color name="black">#000000</color>
22 <color name="pickNumberGrey">#FFB688BF</color>
23
24 <!-- Backgrounds -->
25 <color name="background_white">#ffffffff</color>
26 <color name="background_black">#000000</color>
27 <color name="background_f5_grey">#f5f5f5</color>
28 <color name="background_fc_grey">#f0f0f0</color>
29 <color name="background_e8_grey">#f8f8f8</color>
30 <color name="background_transparent">#0000</color>
```

Ex: We will change the **"colorGreen"** (#2cdca0) with a warmer **"orangeColor"** (#f4af09). We just have to change the value stored at **"colorGreen"** with the value **#f4af09**. Now, anywhere the app uses the former green color, will now be displayed a nice orange color.

Once you have changed the value of a color, you will notice that the small color marker will change its background to the color you have just set for that resource.

```
1 <?xml version="1.0" encoding="utf-8"?>
2 <!--
3 ~ Copyright © 2017 Theme Dimension, a Mobile Web America, Inc. venture
4 -->
5
6 <resources>
7
8 <!-- Cart screen color. -->
9 <color name="colorPrimary">#3F51B5</color>
10 <color name="colorPrimaryDark">#303F9F</color>
11 <color name="colorAccent">#FF4081</color>
12
13 <color name="purple">#d580ff</color>
14 <color name="red">#ff0000</color>
15 <color name="colorGreen">#f4af09</color>
16 <color name="colorGreenDark">#21c58d</color>
17 <color name="colorPink">#a883b9</color>
18 <color name="colorPinkDark">#996dad</color>
19 <color name="testWhite">#FFFFFF</color>
20 <color name="trans">#0000</color>
21 <color name="black">#000000</color>
22 <color name="pickNumberGrey">#FFB688BF</color>
23
24 <!-- Backgrounds -->
25 <color name="background_white">#ffffffff</color>
26 <color name="background_black">#000000</color>
27 <color name="background_f5_grey">#f5f5f5</color>
28 <color name="background_fc_grey">#f0f0f0</color>
29 <color name="background_e8_grey">#f8f8f8</color>
30 <color name="background_transparent">#0000</color>
```

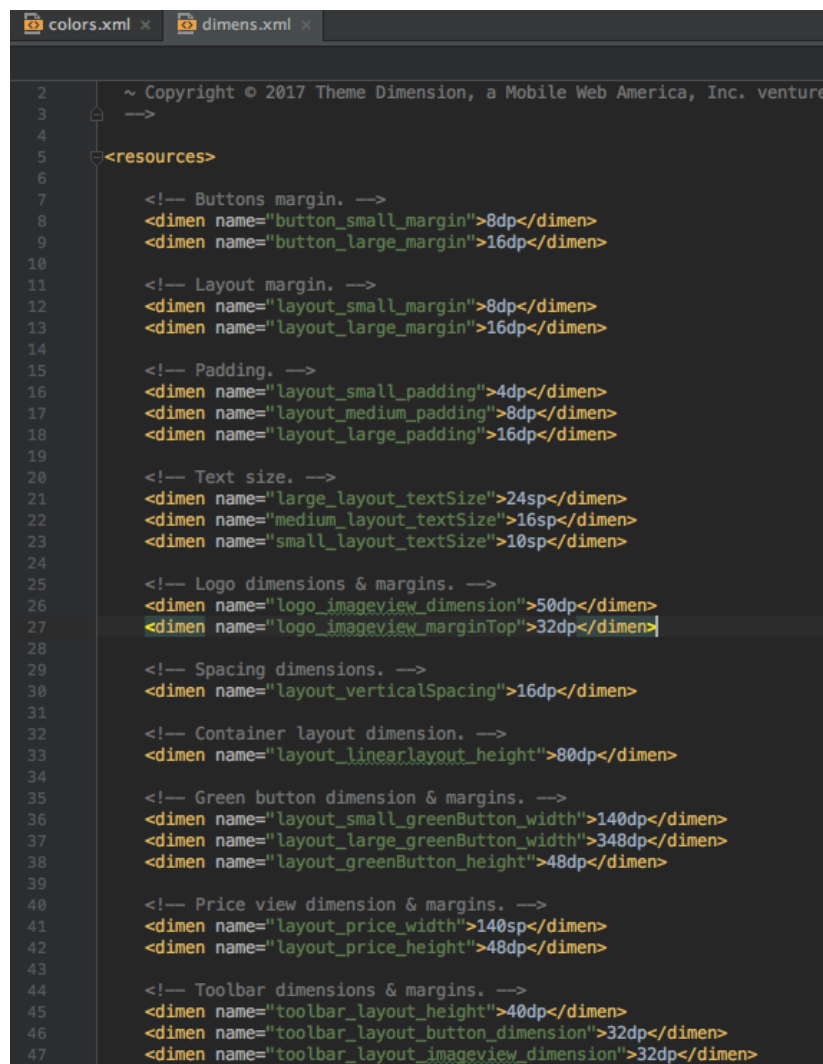



How to change the dimensions used across the app:

In order to change different margins / padding for the elements present in the app, you just have to navigate to the resources directory following the path:

... -> main -> res -> *dimens.xml*

There you can find all the values representing different size constants such as margins, padding, heights and widths (where existent). As before, to change one value you just have to find the *dimen* representing the dimension you want to change, and change its value to the desired one.



```
2 ~ Copyright © 2017 Theme Dimension, a Mobile Web America, Inc. venture
3 -->
4
5 <resources>
6
7     <!-- Buttons margin. -->
8     <dimen name="button_small_margin">8dp</dimen>
9     <dimen name="button_large_margin">16dp</dimen>
10
11     <!-- Layout margin. -->
12     <dimen name="layout_small_margin">8dp</dimen>
13     <dimen name="layout_large_margin">16dp</dimen>
14
15     <!-- Padding. -->
16     <dimen name="layout_small_padding">4dp</dimen>
17     <dimen name="layout_medium_padding">8dp</dimen>
18     <dimen name="layout_large_padding">16dp</dimen>
19
20     <!-- Text size. -->
21     <dimen name="large_layout_textSize">24sp</dimen>
22     <dimen name="medium_layout_textSize">16sp</dimen>
23     <dimen name="small_layout_textSize">10sp</dimen>
24
25     <!-- Logo dimensions & margins. -->
26     <dimen name="logo_imageview_dimension">50dp</dimen>
27     <dimen name="logo_imageview_marginTop">32dp</dimen>
28
29     <!-- Spacing dimensions. -->
30     <dimen name="layout_verticalSpacing">16dp</dimen>
31
32     <!-- Container layout dimension. -->
33     <dimen name="layout_linearlayout_height">80dp</dimen>
34
35     <!-- Green button dimension & margins. -->
36     <dimen name="layout_small_greenButton_width">140dp</dimen>
37     <dimen name="layout_large_greenButton_width">348dp</dimen>
38     <dimen name="layout_greenButton_height">48dp</dimen>
39
40     <!-- Price view dimension & margins. -->
41     <dimen name="layout_price_width">140sp</dimen>
42     <dimen name="layout_price_height">48dp</dimen>
43
44     <!-- Toolbar dimensions & margins. -->
45     <dimen name="toolbar_layout_height">40dp</dimen>
46     <dimen name="toolbar_layout_button_dimension">32dp</dimen>
47     <dimen name="toolbar_layout_imageview_dimension">32dp</dimen>
```

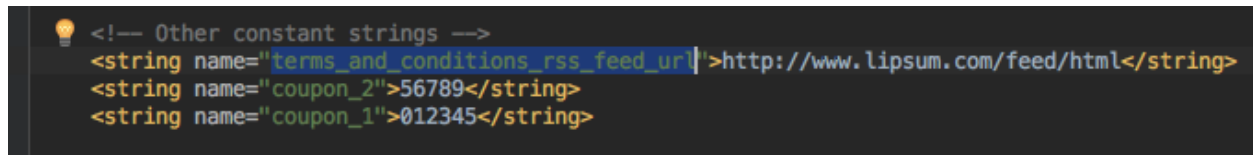


HOW TO CHANGE THE PRIVACY LINK

The “Privacy” section is meant to send the user to the privacy policy of the app.

To place your own privacy link navigate to

... -> **main** -> **res** -> **values** -> **strings**, and then search and replace the value stored at string **"terms_and_conditions_rss_feed_url"** with your own link.

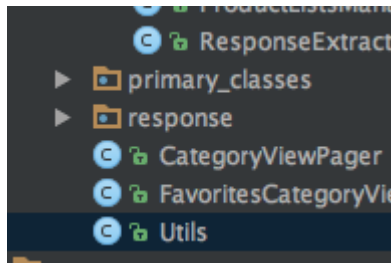


```

<!-- Other constant strings -->
<string name="terms_and_conditions_rss_feed_url">http://www.lipsum.com/feed/html</string>
<string name="coupon_2">56789</string>
<string name="coupon_1">012345</string>

```

Linking the app to a server:




In order for the app to fulfill its objectives, it is recommended that a server connection should be created in order for the user to experience the full Ivory experience and real-time communication to be enabled. To help you achieve this, the app comes with helping Java classes designed to represent the link between app, server and any necessary databases.

You can link your server using the constant strings stored in the **Utils** class, located at

... -> **main** -> **java** ->

com.themedimension.ivoryshop.android -> **Utils.java**.

Note that the NetworkManager is just the pattern for server-side communication, and any additional implementation needed must be developed in order for the app to work accordingly.



```

BASE_URL = "http://192.168.1.168:8000/";
String BASE_URL = "http://192.168.1.203:8000/";
BASE_IMAGE_RES_URL = BASE_URL + "images/products/";
URL_BASE_JSON = BASE_URL + "productlist";
URL_USER_LOGIN = BASE_URL + "loginUser?email=%s&password=%s";
URL_USER_SIGNUP = BASE_URL + "registerUser";
RESET_PASS_URL = BASE_URL + "resetPass?id=%d";
UPDATE_URL = BASE_URL + "update";
GET_CART_PRODUCTS_URL = BASE_URL + "cartProducts?id=%d";
GET_WISHLIST_URL = BASE_URL + "favoriteProducts?id=%d";
UPDATE_CART_URL = BASE_URL + "insertCartProducts";
UPDATE_WISHLIST_URL = BASE_URL + "insertFavoriteProducts";
PAY_PAL_PAYMENT_URL = BASE_URL + "paypal";
DELIVERY_PAYMENT_URL = BASE_URL + "delivery";
STRIPE_PAYMENT_URL = BASE_URL + "stripe";
CUSTOM_PAYMENT_URL = BASE_URL + "custom";

```



How to change the URL to an existing server:

In order to link-up the app to an existing server, you need to navigate to

... -> **src** -> **main** -> **java** -> **Utils.java**

and replace the **BASE_URL** string to the URL used by the server.

Ex: `public static final String BASE_URL = "https://secure-server-domain.web-extension/";`

Now, if your server does need to suffer any additional editing, and neither does the app itself, your users should be able to correctly send requests and get responses from the server.

Note that the app might need extra configuring in order to properly sustain server-side communication.

```
// Change this with the server URL in order for the app to correctly co
public static final String BASE_URL = "http://192.168.1.168:8000/";
```

Note:

Since update v2.1.3, the app supports different custom APIs in order to correctly communicate with the server.

That being said, it means that, for different API calls (such as registration, login, etc.) the app will need to be provided with the correct **BASE_URL** and **url variations**.

BASE_URL: The domain of your server (followed by a "/")

BASE_IMAGE_RES_URL: The extension needed in order to reach the image resource storage location of your server

URL_BASE_JSON: The extension needed in order to reach the data JSON to use as source data. (check [HOW TO CHANGE THE SOURCE DATA DISPLAYED BY THE APP](#) for further details)

URL_USER_LOGIN & URL_USER_SIGNUP: Links needed for the Login/Register APIs communication.

Note: You will need to provide the URL to the endpoint that you link to the app.

```
BASE_IMAGE_RES_URL = BASE_URL + "image"
URL_BASE_JSON = BASE_URL + "productli
URL_USER_LOGIN = BASE_URL + "loginUse
URL_USER_SIGNUP = BASE_URL + "registe
RESET_PASS_URL = BASE_URL + "resetPas
UPDATE_URL = BASE_URL + "update";
GET_CART_PRODUCTS_URL = BASE_URL + "c
GET_WISHLIST_URL = BASE_URL + "favori
UPDATE_CART_URL = BASE_URL + "insertC
UPDATE_WISHLIST_URL = BASE_URL + "ins
PAY_PAL_PAYMENT_URL = BASE_URL + "pay
DELIVERY_PAYMENT_URL = BASE_URL + "de
STRIPE_PAYMENT_URL = BASE_URL + "stri
CUSTOM_PAYMENT_URL = BASE_URL + "cust
```



How to change the currency that the app uses to charge users:

In order to change the currency that the app uses to charge its users, you will need to change the string value of the field **TRANSACTION_CURRENCY** from *USD* to any other currency that you would like to use.

To locate the string mentioned above, you will need to navigate to

... -> **src** -> **main** -> **java** -> **Utils.java**

where you will find the **TRANSACTION_CURRENCY** string. Now you just have to change the value from **USD** to any other currency you want.

Ex: USD, EUR, RON, etc...

```
public static String TRANSACTION_CURRENCY = "USD";
```

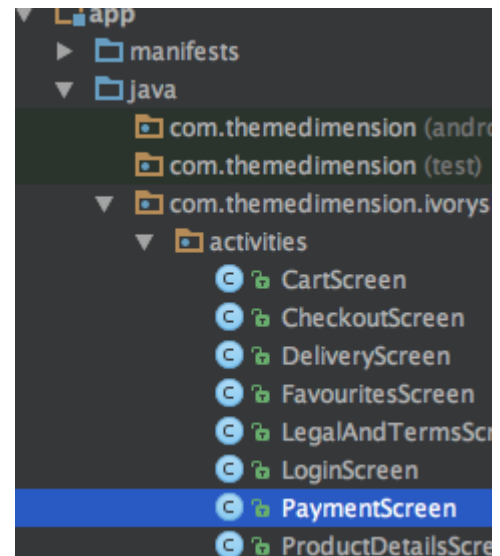
How to enable Stripe:

In order to enable Stripe as the main card payment method, you have to navigate to

... -> **src** -> **main** -> **java** -> **activities** -> **PaymentScreen.java**

and set the **STRIPE_FLAG** value to true. Once you do that, the app will automatically change its card input method accordingly, and the layout will change at the same time.

```
// Change this in order to use a custom card payment  
// Supported methods:  
// - Stripe  
// - PayPal  
// In order to activate one of the methods, just change the flag value  
private static final boolean STRIPE_FLAG = false;  
private static final boolean PAYPAL_FLAG = true;
```



What is more, in order for the app to correctly finish payment requests, an online server must be added, else the app will only get a failure response.



Note that you must change the `STRIPE_PUBLISHABLE_KEY` & `STRIPE_SECRET_KEY` to your own keys provided by Stripe.

```
// Change these with your Stripe API publishable/sec  
public static final String STRIPE_PUBLISHABLE_KEY = "  
public static final String STRIPE_SECRET_KEY = "sk_te
```

Basic workflow of the payment method:

- The app prompts the user with the Stripe CardInputWidget.
- The user enters the card credentials into the widget.
- The app sends a request to the Stripe servers to convert the sensitive data into a token.
- The Stripe servers return the token to the app.
- The app sends the token further to its server for any other processing.

For more details about the Stripe API and how it can be used, please follow the reference below:

<https://stripe.com/docs/mobile/android>

Note that you can also find useful information about how the app manages payments in the source code.

How to enable PayPal Payments:

In order to activate the PayPal Payments inside the app, you need to navigate to
... -> **src** -> **main** -> **java** -> **activities** -> **PaymentScreen.java**

```
// In order to activate one of the methods, just ch  
private static final boolean STRIPE_FLAG = false;  
private static final boolean PAYPAL_FLAG = false;
```

and change the value for the **PAYPAL_FLAG** from *false* to *true* as in the example above:

This will make the app load the button that enables the user to create an order using his / her PayPal account.

As PayPal takes into account different taxes (ex. *Shipping costs*, *Tax*, etc.) you might want to change those values. In order to do that, **in the same file**, just change the values of **PAYPAL_TAX_VALUE** and **PAYPAL_SHIPPING_COST** to the values you need.



```
private static final int RESULT_SUCCEEDED = 1;
private static final double PAYPAL_SHIPPING_COST = 5.00;
private static final double PAYPAL_TAX_VALUE = 0.05;
private static String CARD HOLDER RULE MSG;
```

Note: The app features the actual PayPalActivity in order to enable the user to pay using a PayPal account.

Note: You must change the **CONFIG_ENVIRONMENT** & **CONFIG_CLIENT_ID** with your own keys provided by PayPal.

```
// Change this with the correct PayPal environment
public static final String CONFIG_ENVIRONMENT = "sandbox";
// Change this with your PayPal API key.
public static final String CONFIG_CLIENT_ID = "A";
```

Basic workflow of the payment method:

- The app prompts the user with the PayPal UI.
- The user enters PayPal account credentials.
- The API processes the PayPal account and returns a PaymentConfirmation object to the app.
- The app sends the confirmation to the server for further processing.

Note that in order for the app to correctly finish payment requests, an online server must be added, else the app will only get a failure response.

For further details about the PayPal API integration and PayPal documentation, please refer to the link below:

[PayPal Android SDK Documentation](#)

More about enabling Server-side communication.

Since v2.1.3, the App supports custom backend integration which enables you to easily switch the app-server connectivity from on to off. In order to switch between using or not a server, you need to change the **DEMO_DATA** field from **true** to **false**.

The field mentioned above is located at:

... -> **src** -> **main** -> **java** -> **manager** -> **NetworkManager.java**

```
private static final boolean DEMO_DATA = false;
```

Note that the app comes with the flag set to TRUE by default.



Logging in and Register:

(This section only applies to the Demo version -> no backend)

Since v2.1.2, the app supports users to login with an existing account, create a new account, and choose to skip the registration screen. In order to work accordingly to expectations, the app offers a demo version (with no backend communication) which simulates the server-side communication.

Note that once a backend is added, the following guideline will not be taken into consideration.

Login: The user tries to login with an already existing account.

As the app does not feature a backend to respond to the request, the existing accounts are stored in an external file named ***users.json*** which contains all the accounts created by the app on the running device.

Once a Login action happens, the email and password are tested to see if they appear in the external file. In case of a positive result, all the data used by the user to create that account will be retrieved and used across the app.



Forgot Password: The user wants to reset the password for an account.

In case the email exists inside the ***users.json*** file, the account will have the password overwritten to " **pass1234** " in order to simulate the server-side communication.

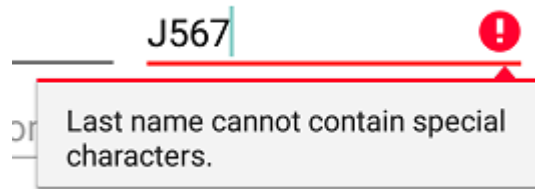
**Register:** The user creates a new account.

In order to create a new account, the data that the user wants to use in order to register inside the app will also be stored in the ***users_json*** external file.

The image shows a mobile app interface for a registration screen. At the top, there is a black header bar with a white back arrow and the title "Register". Below the header, the form consists of several input fields: "First name", "Last name", "Email address", "Phone number", "Street", "Number", "City", "ZIP", "District", "Country", "Password", and "Confirm password". Each field has a dashed line indicating a character limit of 8. Below the password fields, there is a checkbox labeled "Accept the Terms & Conditions". At the bottom of the form is a large purple button labeled "Register". The entire form is enclosed in a light gray border.

The workflow is a little different than in case of a Login action, as the email is searched inside the file and in case it is found, the app will let the new user know that the email has already been used. Moreover, in case the app can create the new user, it will write the user's data inside the file and will login him (taking him to the All Products Screen).

Note: If the user completes a field with the wrong input, an alert will inform him about the error.

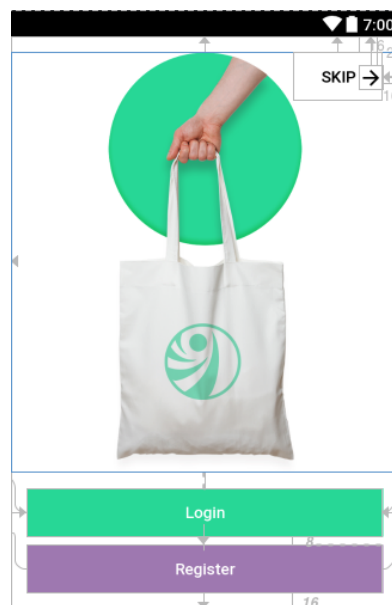


Skip registration: The user skips any type of authentication method using the **Skip** button.

In case of a Skip action, the app will use default data to instantiate an anonymous user with the ID **-1**. This means that, in case a backend is used, the server will be correctly provided with the necessary data in order to make orders (if it is the case); all other fields are left empty.

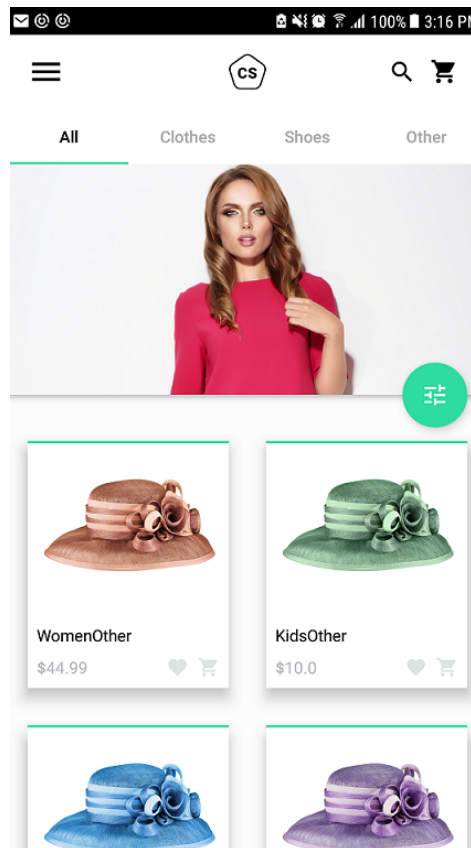
For the user to make orders, he will need to provide all the data needed (Address, email, phone, etc.) later in the app. (if it is the case)

Note: All the data that must remain persistent across multiple uses of the app (products added in cart / favorites lists) will be stored in [SharedPreferences](#).



USAGE

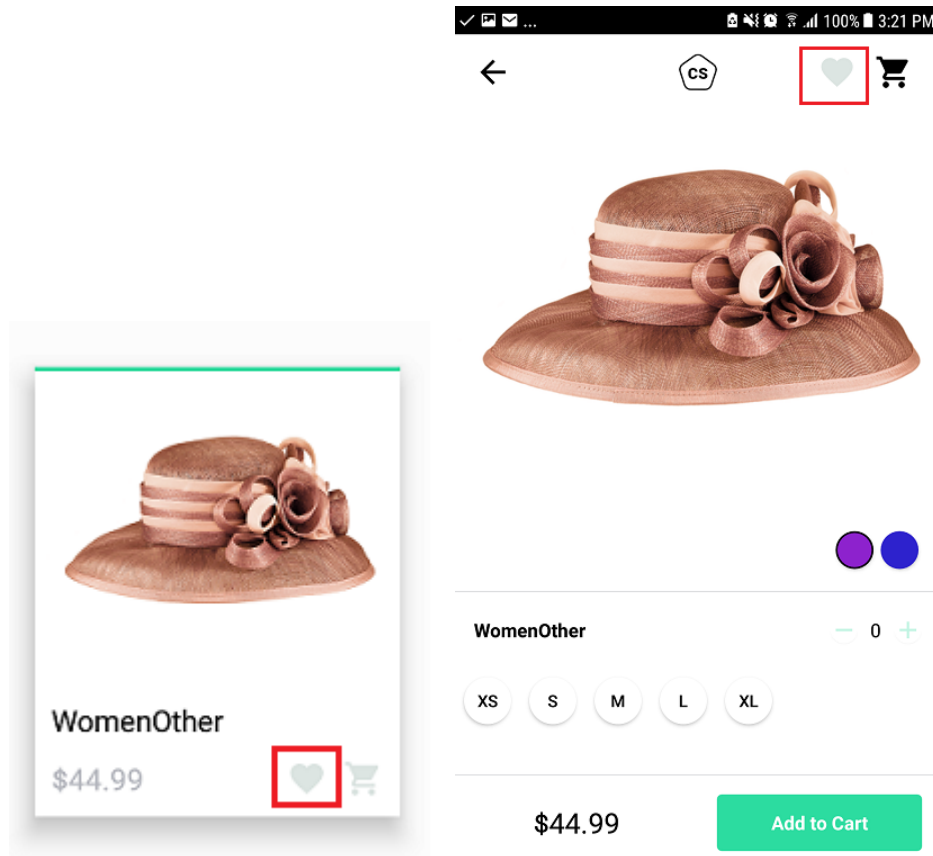
The main page shows all the product items from the feed.



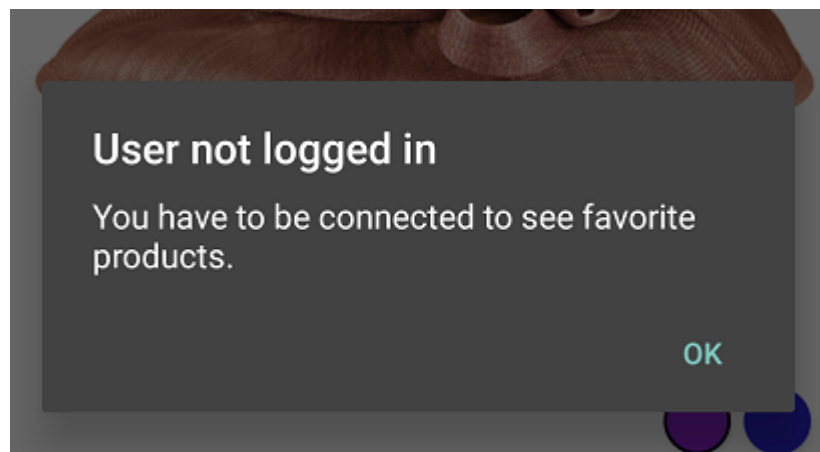
HOW TO ADD PRODUCTS TO FAVORITES

You can add your products to favorites by pressing the heart shaped icon located next to the product's price, or by entering the product's details screen, where you will find the same icon on the top of the screen.

You can add products to cart by accessing the product's details screen. To add items to cart, you need to first select a size. You can modify the number of products to add to cart by pressing the "+" and "-" symbols located in the screen.



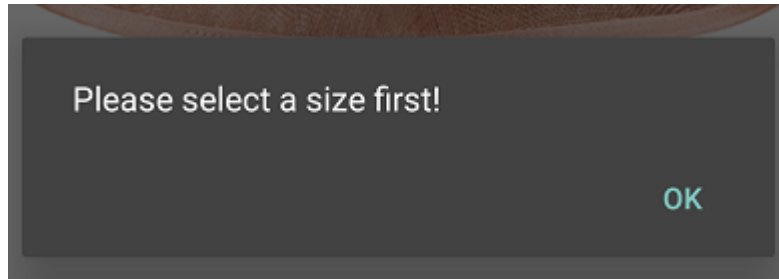
Note: This option is available just for users that are logged in. For the other users a popup will inform them that they have to login in order to be able to add products to favorites.



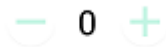


HOW TO ADD PRODUCTS TO CART

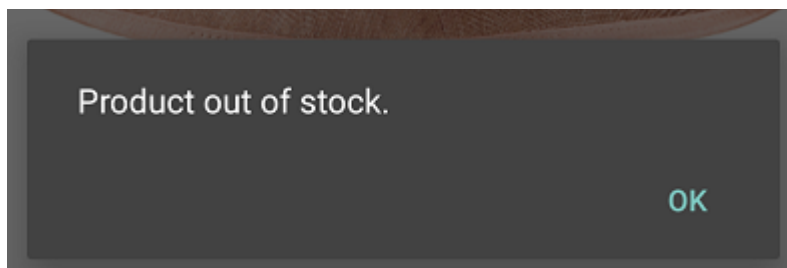
To add products to cart access the product's details screen. To add items to cart, you need to first select a size, otherwise a popup will inform the user that he has to select a size.



User can modify the number of products to add to cart by pressing the "+" and "-" symbols located in the screen. **Note:** these buttons are available if the user has pre-selected a size.



Note: Products can be unavailable. If so, a popup will inform the user that the selected product, the selected size is out of stock.



HOW TO SEARCH PRODUCTS

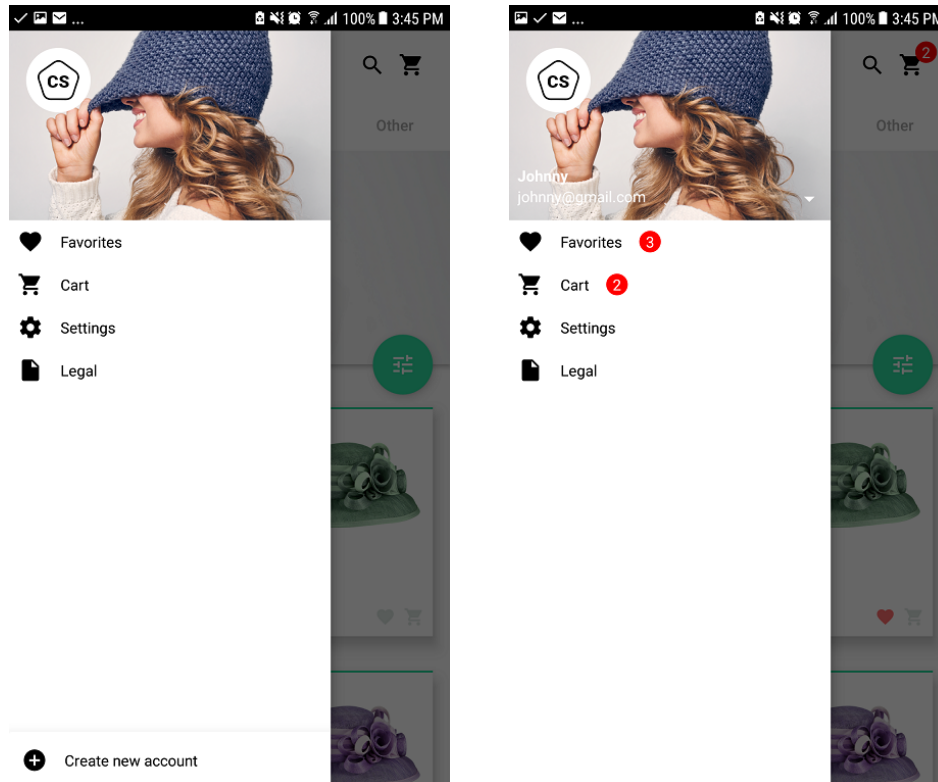
You can use the search function to find articles that match your search. Access the search function by clicking the search icon on top of the page.



HOW TO ACCESS THE MENU

You can access the menu by Clicking the menu button in the left or by dragging from the left margin. You can dismiss the menu by clicking outside of the menu or by dragging the right margin in the left.

In the top of the menu you will see a list from where you can access the Favorite products page, the Cart Page, the Settings page or the Privacy Policy page.



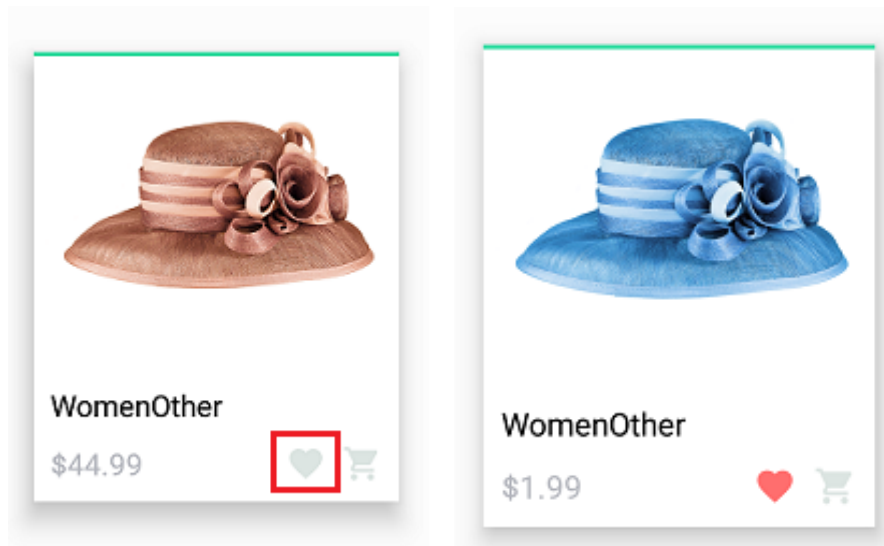
Note: Favorite Products and Settings are available just for users that are logged in.

Note: Create new account option is available just for users that are not logged in.



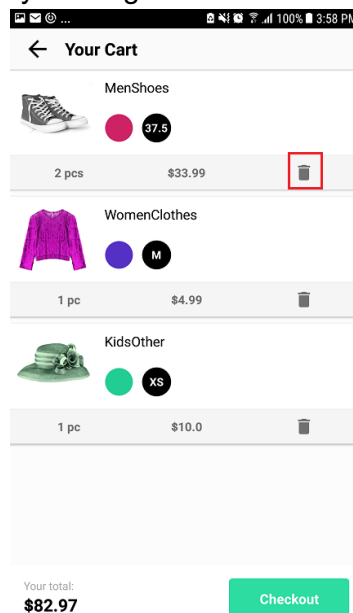
HOW TO REMOVE ARTICLES FROM “FAVORITE”

You can modify the list of favorite products by clicking heart icon from the respective product. The icon's color will change accordingly.



HOW TO REMOVE ARTICLES FROM “CART”

You can modify the cart's content by clicking the remove icon in the cart screen.

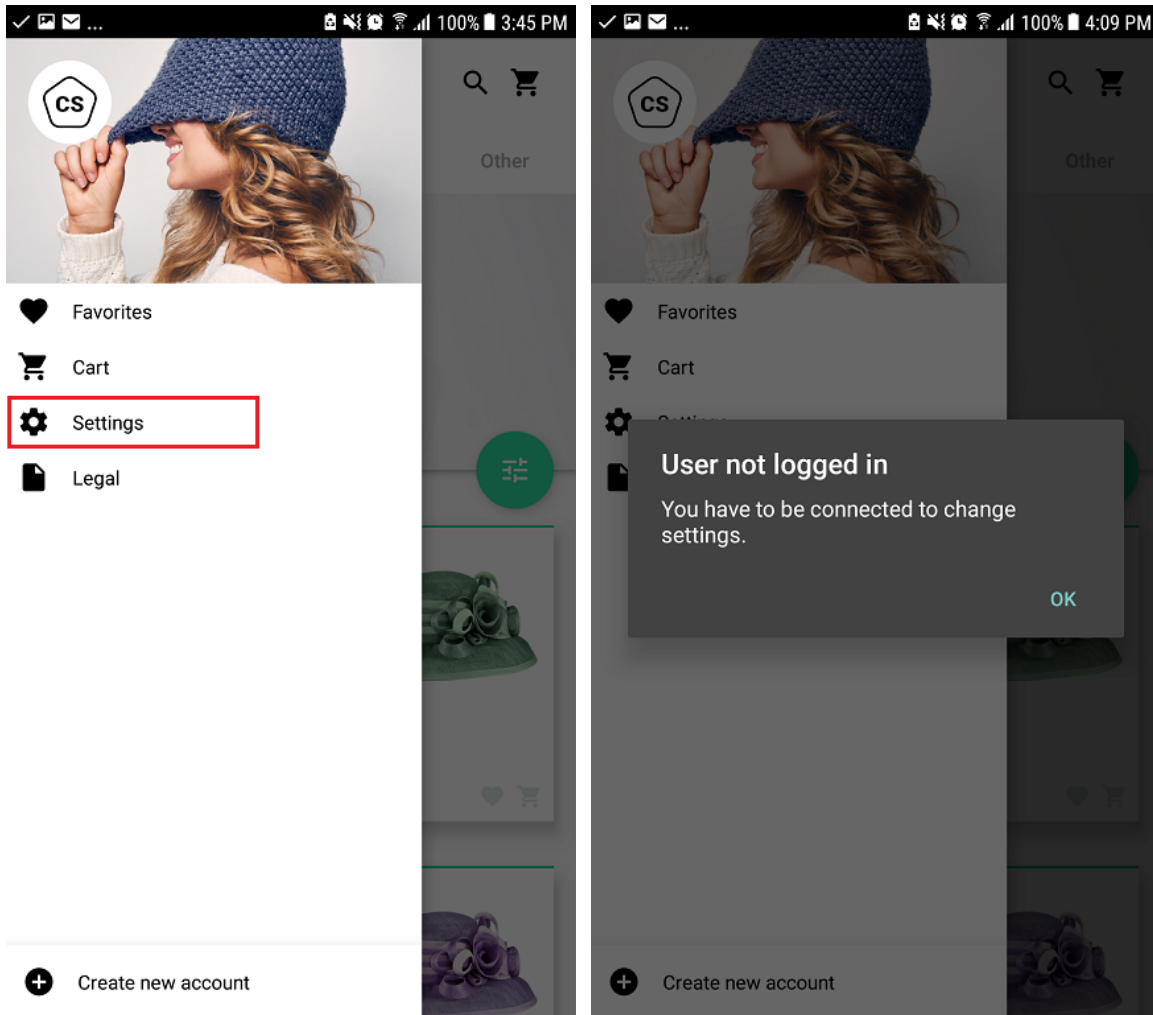




HOW TO ACCESS THE SETTINGS

You can access the settings by clicking on the “Settings” section from the menu.

Note: **Settings** are available just for users that are logged in, otherwise a popup will inform the user that he has to login to be able to change settings.





HOW TO MODIFY THE SETTINGS

You can modify the settings by changing the data saved in the respective fields and then clicking the button “SAVE”.

✓ 📧 ... 100% 4:11 PM

← Settings

Delivery details

Johnny J

johnny@gmail.com

1234567890

A Street 12

A City 123

This Ro

Card details

1234567890 CVS Exp. Date: MM/YY 📅

Card Holder Name

Save

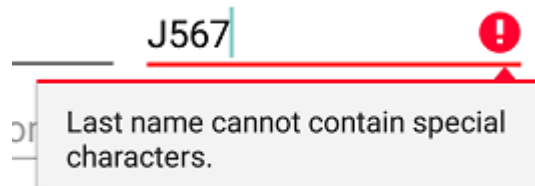
Note: The email cannot be edited.



Note: In order to be able to complete the card expire date, the user has to click on the calendar icon.

Exp. Date: MM/YY 

Note: Whenever the user inserts a field incorrectly, a popup will inform him about the error.





GIVE FEEDBACK

You can give us feedback in order to help improve and create higher quality template and apps at:

support@themedimension.com