**Report**

**Introduction:**

The application that we have made is called the PhotoSnap. We have used the Kotlin Programming Language in the application. The application main purpose is to use the idea of Reverse Image Search. So according to this idea the PhotoSnap application is used to find out the related images on the top 3 Search Engines API’s as given in the document. And the after searching we can view the related images that came from the server and can also download and save it into database for offline usage in the app. We can choose the image either from the gallery or the camera for searching purposes. There is also an option for deleting the images from the database. The application has modern and attractive design of which any user can use the application without any difficulties.

**Coding Structure:**

First, the application main architecture is based on MVVM (Model View ViewModel), the recommended architecture provided by the android community. And then after that we have arranged the code in packages or component wise so that every related functionality should be in the same directory. We have made the 5 packages and below are their names

1. Data
2. Home
3. Reverse Image Result
4. Saved result
5. Util

* **Packages:**

The data package hold all the database, network and model related code. The Home package has the main screen which is visible to the user first time after the splash screen. The Reverse Image Result package is used to search the result from the 3 provided search engines by uploading the image URL and then it will show the result images related to the uploaded image as the main purpose of Reverse Image Result. The Saved Result package is for displaying the all the image that are saved into the database and use can view the image as a pop up screen for bigger view. The last Util package is used for all the utilities used in the app like permission, toast, etc.

* **Architecture Components**

We have used the latest libraries like navigation, Extension function, Databinding in the application because they provide us a lot of simplicity and it is very few lines to code and easy to understand. For getting data from the network/API’s I have used the Retrofit Library as it is very popular and easy to use and its annotation are very interesting that make our task very simple by using its annotations.

* **3rd Party Libraries:**

We have used 3 – 4 third party libraries in the application . We have used the Glide library for image viewing purposes . We have also use the Croppy Library for cropping the Image into different sizes purposes. We have use the Image Popup library for viewing the image in a pop up dialogue as bigger complete images.

* **Background/Long Running Tasks**

For heavy or long running operations we have used the ViewModel Scope and Global Scope of Coroutine scope to make the Mani UI thread safe from hanging the screen. The coroutine has make our long running operations easy to use by using the suspend keyword so that they run in separate thread and then continue to the next code in the coroutine block. For Updating the UI we have use the Dispatchers.Main to update the UI from the coroutine scope.

* **Designing Component:**

For designing the User interface, we have used the Constraint Layout as it is rich layout with a lot of features and functionalities. The drag and drop-in android studio help us to design the User Interface very easily. we have also made custom Dialogue box for notification purposes and for choosing between the camera and gallery to redirect the user accordingly.

Here are the screens design attached below

A person holding a phone

Description automatically generated with low confidence Graphical user interface, application

Description automatically generated

Graphical user interface, application

Description automatically generated Graphical user interface, website

Description automatically generated

**Sub-Requirements:**

1. **Elvis Operator:**

Basically, the Elvis operator is used to check whether the value is null then it returns that value which is given to it (left side value) otherwise the default value (right side value). So in the application we can use this operator but we have mostly used the if condition to check the value null or many different scenarios for different purposes.

Elvis Operator = ‘ **?:** ‘

1. **Lambda Functions:**

So we have used in like onclickListener of some buttons text views etc. Here we have used on image clear button to clear the image.

A screenshot of a computer

Description automatically generated with medium confidence

As we have used MVVM so all Observers are using lambda functions to observe the variable of the View Model. Here is the sample

Graphical user interface, text, application, email

Description automatically generated

1. **Classes and Companions :**

we have made model classes to follow the principle of OOP so that easy to work with. We have made a class **Server Response** to get the response from server . It is used with retrofit to read then response and return the list of Object to us where we used to make operations on it.

A picture containing graphical user interface

Description automatically generated

we also have made a Database Model class to for database operations purposes so that make it easy to work on .

Graphical user interface, text, application

Description automatically generated

Companions objects are those which can be access by the class name , we also have used for database columns names , table name etc. purposes.

Graphical user interface, text, application

Description automatically generated

Also in the utility class we have made all the functions in Companion so that to access them directly without making the object of the class and use the functions of Utility class.

1. **Recycler View Adapter and its View:**

we have made 2 Recycler view Adapters in the application. The one is used to show all the fetched results from the API and the other is used to show all the images saved in the database.

1: ReverseImagesRecyclerViewAdapter

Graphical user interface, text, application, email

Description automatically generated

The view which Is used here is called item\_products in the view holder.

Graphical user interface, text, application, email

Description automatically generatedGraphical user interface, application

Description automatically generated

Firstly we pass the empty list to initialize the Recycler View Adapter and then we have made a swap function which refresh the Adapter and show us the latest data after calling the notifyDataSetChanged() function.

The same is for the other recycler view adapter where we use the same logic and implementation but the design is different and some parts.

1. **Activities/Fragments:**

We have used only 2 activities .The one is splash activity for splash screen which appear on the start of the application and the second is Main Activity where we have configured the Navigation Controller and Navigation view and menus. So we have used the Navigation which replaces the Main Navigation with the other that we want to show.

All the other 3 screen are designed as Fragments as they are light weight and faster performance and easy to use and hold the same context to an activity to share details between fragments.

Text

Description automatically generated

Graphical user interface

Description automatically generated

1. **Pass Data b/w Fragments/Activities:**

As we have used the Navigation component so that’s why it is easy and safe to use the NavigationsArgs . But according to my application design we did not have used it because we used the Companion object to get the data between activates/fragments by calling the name of the class and the required data variable.

1. **Coroutines / Background Tasks:**

As we know that long running operations should not be done on main UI thread so as we have used the MVVM due to which we can easily use the view model scope to RUN the coroutine scope. we have use view model Scope for fetching the search related image by uploading the image URL. And then after getting the result we post the value of a Live Data variable which is observing in the Fragment and then all data of related images passes to the recycler view adapter from the fragment.

A picture containing table

Description automatically generated

1. **Sql lite Database :**

As mentioned in the document that we have to store the images as BLOB so we store all the images as BLOB after downloading from the server which send us the the URL of the image. But in kotlin it is equivalent to Byte Array. So here you can check the insert code

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application, email

Description automatically generated

So you can see the we have made 2 functions to download and convert the image to Byte Array and then we saved it into database. Its functionality is used in the adapter class on every item there is a button to download the image and save into database.

1. **3rd Party view:**

we have the 3rd party library for viewing the image of on a bigger screen in the dialogue box. Here you can see that Image Pop Up library is used to display the image in pop up when click on it in the Adapter Recycler view.

Graphical user interface, text

Description automatically generated with medium confidence

we also use the Croppy Image Library to crop the images of different size in the main screen after selecting the image Graphical user interface, text, application, email

Description automatically generated

we also have made a custom notification dialogue which shows when we have to select the image either from Gallery or Camera. On it we have made 2 buttons to get input from user of their choice to select the image.

Graphical user interface, text, application

Description automatically generated

1. **Callbacks:**

we have used an interface to get the result of Croppy Image from activity to fragment because the croppy library returns the result to the activity so we have to get that result in the fragment by making my own interface and callback when it receive the URI of the image.

Graphical user interface, text, application

Description automatically generated

Graphical user interface, text, application

Description automatically generated

And in the Fragment we just make the object of that interface to get the callback and receive the URI of the cropped image by croppy Image library

Graphical user interface, text, application, email

Description automatically generated

1. **Followed Instructions:**

Yes we have followed all the main core rules as mentioned in the document like coroutine scope , database operations etc. All of the conditions must be satisfied according to the guidelines.

Some of things may differ like I used MVVM architecture and some 3 party libraries to make the applications easier to use .

1. **Additional challenges:**
   * + we have used the 3rd party libraries
       - implementation 'com.github.lyrebirdstudio:Croppy:0.2'  
           
         implementation 'com.github.chathuralakmal:AndroidImagePopup:1.2.2'
     + As we have convert the image for uploading because the server accepts on JEPG format so we convert the format of the image

Graphical user interface, text, application

Description automatically generated

* The design is comprehensive and easy to navigate as any user can use the application without any training
* Yes we have used the Android core service like network permissions and storage permission to read and write on users device otherwise user will not be able to select the image if user does not allow the permission

**Text

Description automatically generated**

**Conclusion:**

There are lot of application usage of Reverse Image Search in the daily life. As We have use Reverse Image search idea in our PhotoSnap Application whose main purpose is to find out the related images from the 3 main search engines. We can also extend the app functionality in such a way that user can download the images into their gallery for using them as a wallpaper or later purposes but we had only limited used in the application only.