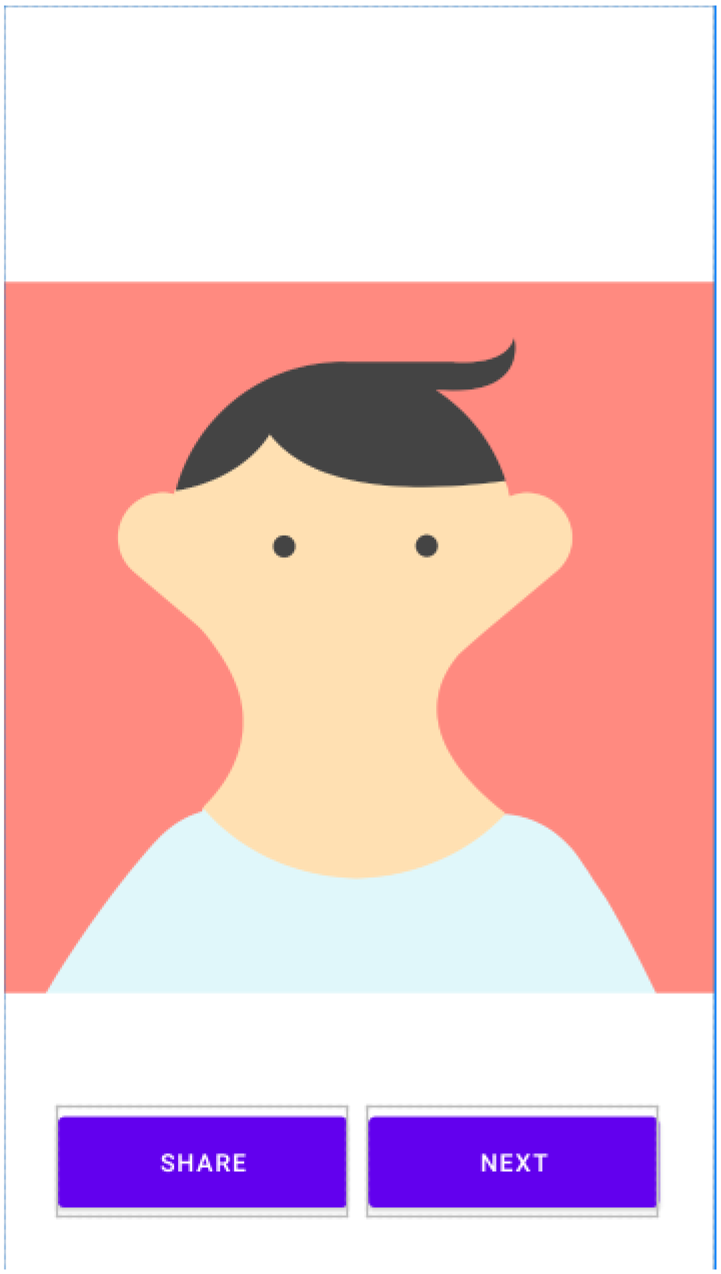
Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        android:id="@+id/imageView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf=" "@id/shareButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:srcCompat="@tools:sample/avatars"
        android:contentDescription="@string/memebanadia"/>

    <Button
        android:id="@+id/shareButton"
        android:layout_width="168dp"
        android:layout_height="64dp"
        android:layout_marginStart="32dp"
        android:layout_marginBottom="30dp"
        android:text="@string/share"
        android:onClick="shareMeme"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

    <Button
        android:id="@+id/nextButton"
        android:layout_width="168dp"
        android:layout_height="64dp"
        android:layout_marginEnd="32dp"
        android:layout_marginBottom="30dp"
        android:text="@string/next"
        android:onClick="nextMeme"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



https://developer.android.com/training/volley

Gradle Scripts>> build.gradle(Module:MemeShareApp.app)

```
plugins {
    id 'com.android.application'
    id 'kotlin-android'
    id 'kotlin-android-extensions'
}
```

```
dependencies {

    implementation "org.jetbrains.kotlin:kotlin-stdlib:$kotlin_version"
    implementation 'androidx.core:core-ktx:1.3.1'
    implementation 'androidx.appcompat:appcompat:1.2.0'
    implementation 'com.google.android.material:material:1.2.1'
    implementation 'androidx.constraintlayout:constraintlayout:2.0.1'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.2'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.3.0'

    implementation("com.android.volley:volley:1.1.1")

}
```

Android View

Manifests>> AndroidManifest.xml

```
manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.memeshareapp">

<uses-permission android:name="android.permission.INTERNET"/>
<application
    android:allowBackup="true"
    android:icon="@mipmap/ic_launcher"
    android:label="@string/app_name"
```

MainActivity.kt

Taking help of this

https://developer.android.com/training/volley/simple

```
package com.example.memeshareapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import com.android.volley.Request
import com.android.volley.Response
import com.android.volley.toolbox.StringRequest
import com.android.volley.toolbox.Volley

class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    private fun loadMeme(){
        // Instantiate the RequestQueue.
        val queue = Volley.newRequestQueue(this)
        val url = "https://www.google.com"


// Request a string response from the provided URL.
        val stringRequest = StringRequest(
            Request.Method.GET, url,
            Response.Listener<String> { response ->

            },
            Response.ErrorListener { })

// Add the request to the RequestQueue.
        queue.add(stringRequest)
    }

    fun shareMeme(view: View) {

    }
    fun nextMeme(view: View) {

    }
```

Now do the following changes

```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        loadMeme()
    }

    private fun loadMeme(){
        // Instantiate the RequestQueue.
        val queue = Volley.newRequestQueue(this)
        val url = "https://www.google.com"

// Request a string response from the provided URL.
        val stringRequest = StringRequest(
            Request.Method.GET, url,
            Response.Listener<String> { response ->
                Log.d("success Request",response.substring(0, 500))

            },
            Response.ErrorListener {
                Log.d("error", it.localizedMessage)
                })
```

As this is A string Object we have to change it to JSON object

Use this https://developer.android.com/training/volley/request

Make a Standard request

```
private fun loadMeme(){
        // Instantiate the RequestQueue.
        val queue = Volley.newRequestQueue(this)
        val url = "https://meme-api.herokuapp.com/gimme"

// Request a string response from the provided URL.
        val jsonObjectRequest = JsonObjectRequest(
            Request.Method.GET, url,null,
            val url=response.getString("url")

},
Response.ErrorListener {


})
```

Now needed Glide Android Library for Android images.
https://github.com/bumptech/glide

Gradle Scripts>>build.gradle(project)

```
llprojects {
    repositories {
        google()
        mavenCentral()
        jcenter() // Warning: this repository is going to shut down soon
    }
}
```

Gradle Scripts>>build.gradle(App)

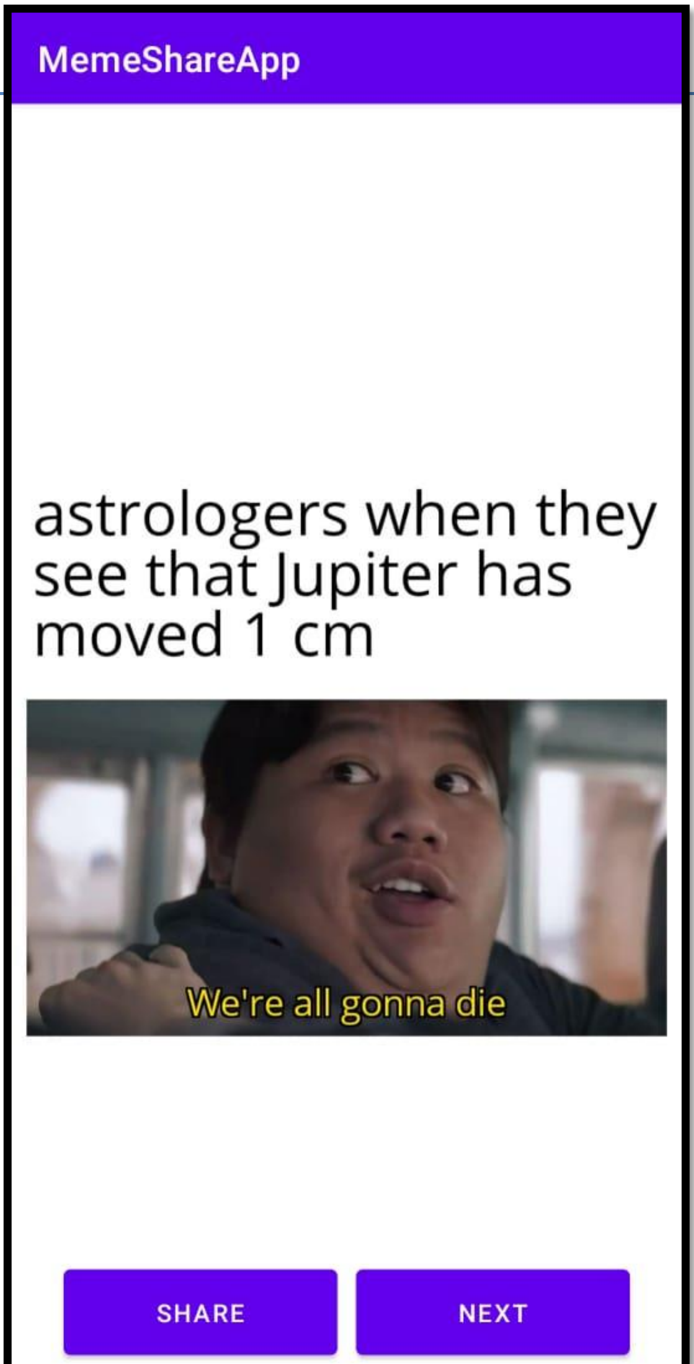
```
dependencies {
  implementation 'com.github.bumptech.glide:glide:4.12.0'
  annotationProcessor 'com.github.bumptech.glide:compiler:4.12.0'
}
```

Sync now

```
private fun loadMeme(){
        // Instantiate the RequestQueue.
        val queue = Volley.newRequestQueue(this)
        val url = "https://meme-api.herokuapp.com/gimme"
// Request a string response from the provided URL.
        val jsonObjectRequest = JsonObjectRequest(
            Request.Method.GET, url,null,
            Response.Listener { response ->
                val url=response.getString("url")
                Glide.with(this).load(url).into(memeimageView)

            },
            Response.ErrorListener {


            })

// Add the request to the RequestQueue.
        queue.add(jsonObjectRequest)
    }
```

Run the Application

Screenshot:-



Next phase of Project:- Decoration and Using buttons

Loading next meme >>mainActivity.kt

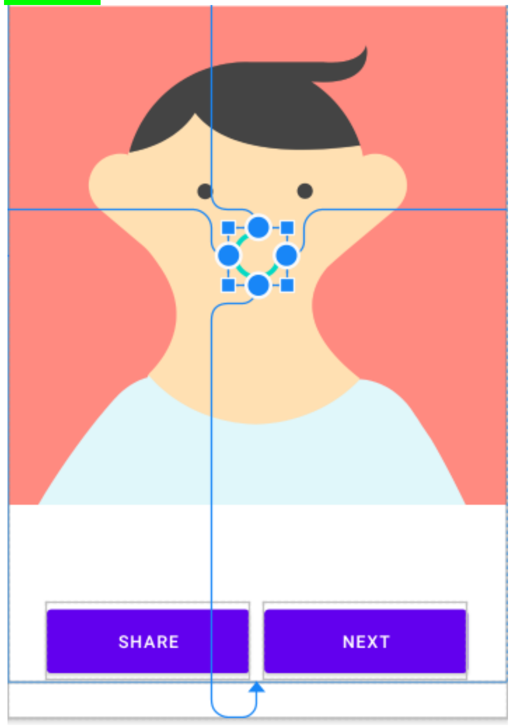
```
fun nextMeme(view: View) {
    loadMeme()
}
```

Again run the APP and see the Next button is working

Now the problem is if I am loading the App it may take seconds to load the images
so we will add Loader like something that **indicates wait its working** and showing you the result.

Put it in center go to activity_main.xml (below image view)

```
<ProgressBar
    android:layout_width="wrap_content"
    android:id="@+id/progressBar"
    android:layout_height="wrap_content"
    app:layout_constraintTop_toTopOf="@id/memeimageView"
    app:layout_constraintBottom_toBottomOf="@id/memeimageView"
    app:layout_constraintLeft_toLeftOf="@id/memeimageView"
    app:layout_constraintRight_toRightOf="@id/memeimageView"
    />
```



But the thing is when image loaded remove the progress Bar so

Actually loadimgae responds fast
else
gradle takes time to load download and show the image

So put in gradle with if loaded
and error cases

```
Request.Method.GET, url,null,
Response.Listener { response ->
    val url=response.getString("url")
    Glide.with(this).load(url).listener(object : RequestListener<Drawable>{
        override fun onLoadFailed(
            e: GlideException?,
            model: Any?,
            target: Target<Drawable>?,
            isFirstResource: Boolean
        ): Boolean {
            TODO("Not yet implemented")
        }

        override fun onResourceReady(
            resource: Drawable?,
            model: Any?,
            target: Target<Drawable>?,
            dataSource: DataSource?,
            isFirstResource: Boolean
        ): Boolean {
            TODO("Not yet implemented")
        }
    }).into(memeimageView)

},
Response.ErrorListener {


})
```

Then

```
override fun onLoadFailed(
    e: GlideException?,
    model: Any?,
    target: Target<Drawable>?,
    isFirstResource: Boolean
): Boolean {
    progressBar.visibility=View.GONE
    return false
}

override fun onResourceReady(
    resource: Drawable?,
    model: Any?,
    target: Target<Drawable>?,
    dataSource: DataSource?,
    isFirstResource: Boolean
): Boolean {
    progressBar.visibility=View.GONE
    return false
}
```

Progress bar is working fine

Now lets go with

Share button

For this we are just going to share the URL so taking url as common variable and put all url places as variable

```
class MainActivity : AppCompatActivity() {

    var currentImageURL: String?=null
    override fun onCreate(savedInstanceState: Bundle?) {
        super
```

..

```
val jsonObjectRequest = JsonObjectRequest(
    Request.Method.GET, url,null,
    Response.Listener { response ->
        currentImageURL=response.getString("url")
        Glide.with(this).load(currentImageURL).listener(object :
RequestListener<Drawable>{
            override fun onLoadFailed(
```

Now for share button we will use **INTENT**

It is basically a intercommunication between next page or other pages

Communication between one proccess to another

So done the changes like

In MainActivity.kt

```
fun shareMeme(view: View) {
        val intent = Intent(Intent.ACTION_SEND)
//now this will be used to send
        intent.type= "text/plain" //according to these apps will be added
        intent.putExtra(Intent.EXTRA_TEXT,"Hey, checkout this cool meme,
click on the link below $currentImageURL")
        //add a app chooser
        val chooser=Intent.createChooser(intent, "Share this meme using...")
        startActivity(chooser)
    }
```

Now it’s time to update the theme

One can also update button colors and backgrounds but I am not doing that

Set some colors in **colors.xml**

App>>res>>values>>colors.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <color name="purple_200">#FFBB86FC</color>
    <color name="purple_500">#FF6200EE</color>
    <color name="purple_700">#FF3700B3</color>
    <color name="teal_200">#FF03DAC5</color>
    <color name="teal_700">#FF018786</color>
    <color name="black">#FF000000</color>
    <color name="white">#FFFFFFFF</color>

</resources>
```

Go to acticity_main.xml

```
<ImageView
        android:id="@+id/memeimageView"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:contentDescription="@string/memebanadia"
        app:layout_constraintBottom_toBottomOf="@id/shareButton"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0"
android:background="@color/black"
```

```
tools:srcCompat="@tools:sample/avatars" />
```

But I not done this as I want the white theme

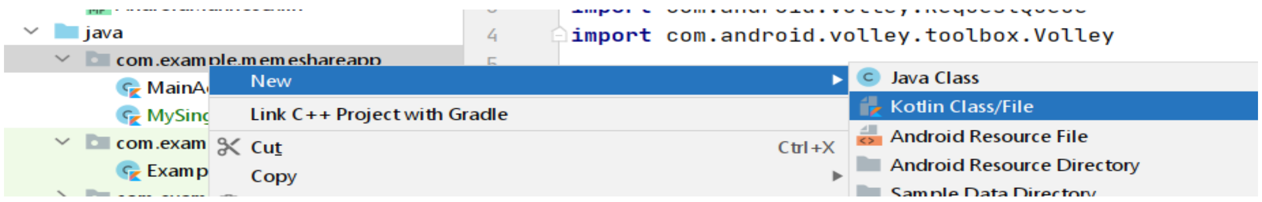
One problem or a Suggestion from google is that use singleton pattern for volley api as this app handles only one url so going through the singleton pattern

https://developer.android.com/training/volley/requestqueue#kotlin

Copy the content of second code part

```
class MySingleton constructor(context: Context) {
    companion object {
        @Volatile
        private var INSTANCE: MySingleton? = null
        fun getInstance(context: Context) =
            INSTANCE ?: synchronized(this) {
                INSTANCE ?: MySingleton(context).also {
                    INSTANCE = it
                }
            }
    }
    val imageLoader: ImageLoader by lazy {
        ImageLoader(requestQueue,
                object : ImageLoader.ImageCache {
                    private val cache = LruCache<String, Bitmap>(20)
                    override fun getBitmap(url: String): Bitmap {
                        return cache.get(url)
                    }
                    override fun putBitmap(url: String, bitmap: Bitmap) {
                        cache.put(url, bitmap)
                    }
                })
    }
    val requestQueue: RequestQueue by lazy {
        // applicationContext is key, it keeps you from leaking the
        // Activity or BroadcastReceiver if someone passes one in.
        Volley.newRequestQueue(context.applicationContext)
    }
    fun <T> addToRequestQueue(req: Request<T>) {
        requestQueue.add(req)
    }
}
```

Put it in a separate class of kotlin



```
import com.android.volley.toolbox.Volley
```

Name it as MySIngleton

And after importing All it will look like

MySingleTon.kt

```
import android.content.Context
import com.android.volley.Request
import com.android.volley.RequestQueue
import com.android.volley.toolbox.Volley

class MySingleton constructor(context: Context) {
    companion object {
        @Volatile
        private var INSTANCE: MySingleton? = null
        fun getInstance(context: Context) =
            INSTANCE ?: synchronized(this) {
                INSTANCE ?: MySingleton(context).also {
                    INSTANCE = it
                }
            }
    }

    private val requestQueue: RequestQueue by lazy {
        // applicationContext is key, it keeps you from leaking the
        // Activity or BroadcastReceiver if someone passes one in.
        Volley.newRequestQueue(context.applicationContext)
    }
    fun <T> addToRequestQueue(req: Request<T>) {
        requestQueue.add(req)
    }
}
```

Now go to ActivityMain

Update the following

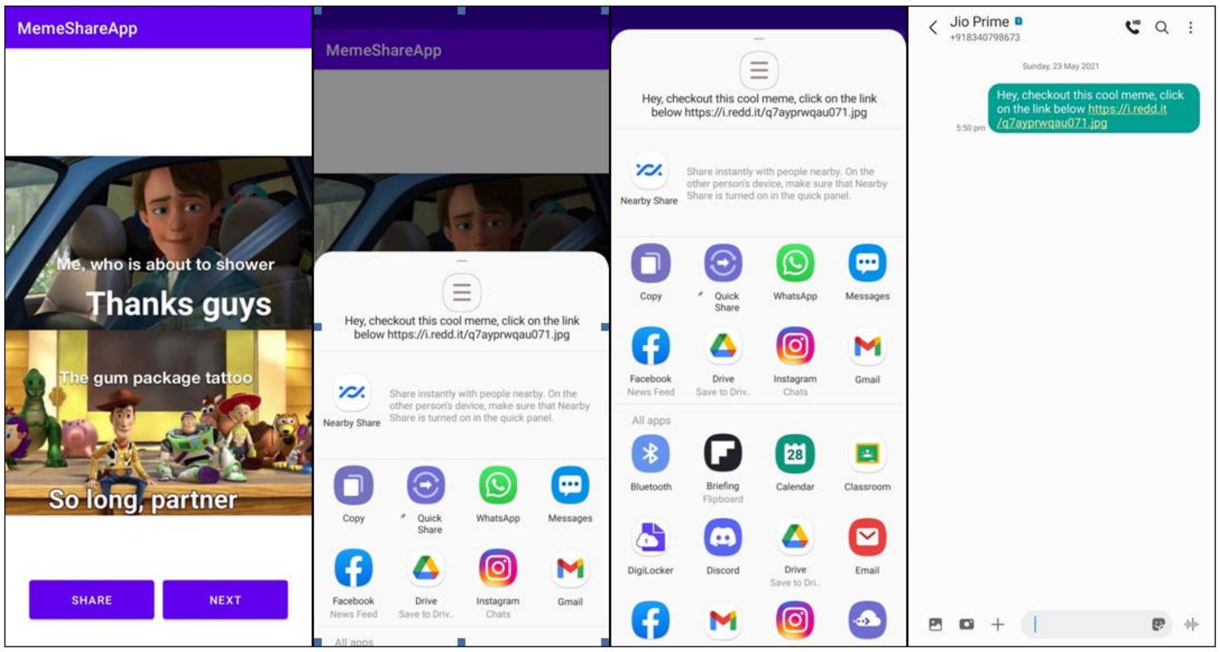
```
override fun onResourceReady(
                        resource: Drawable?,
                        model: Any?,
                        target: Target<Drawable>?,
                        dataSource: DataSource?,
                        isFirstResource: Boolean
                    ): Boolean {
                        progressBar.visibility=View.GONE
                        return false
                    }
                }).into(memeimageView)

            },
            {
                Toast.makeText(this, "Something went
wrong",Toast.LENGTH_LONG).show()
```

```
            })
// Add the request to the RequestQueue.
        MySingleton.getInstance(this).addToRequestQueue(jsonObjectRequest)
    }

    fun shareMeme(view: View) {
        val intent = Intent(Intent.ACTION_SEND)
```

Lastly added app icon from manifests s

And the App is Ready.

Screenshots:-


Hey, checkout this cool meme, click on the link below https://i.redd.it/q7ayprwqau071.jpg

Run build and Download the app
You can explore more API’s in rapidAPI website

Some projects are

- Weather app
- Recepi app