Experiment No 6

Develop a Program to implement a Custom Button and Handle the displayed message on the Button Press

Aim: To Develop a Program to implement a Custom Button and Handle the displayed message on the Button Press.

Procedure:

Step1: Create the New Project->Empty Views Activity

Step 2: Design the User Interface in the **activity_main.xml** file

Activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/main"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".MainActivity">
   <EditText
      android:id="@+id/editTextMobile"
      android:layout_width="300dp"
      android:layout_height="wrap_content"
      android:layout_below="@id/textView"
      android:layout_centerHorizontal="true"
      android:layout_marginTop="20dp"
      android:hint="@string/enter_your_register_number"
      android:inputType="textShortMessage"
      android:textSize="16sp"
      android:padding="10dp"
      android:backgroundTint="@android:color/darker_gray"
      android:fontFamily="sans-serif" />
```

```
<TextView
    android:id="@+id/textView"
    android:layout_centerInParent="true"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="50dp"
    android:text="@string/welcome_to_mohan_babu_university"
    android:textSize="18sp"
    android:textColor="@color/black"
    android:fontFamily="sans-serif-medium" />

  <Button
    android:id="@+id/button"
    android:layout_width="200dp"
    android:layout_height="60dp"
    android:layout_below="@id/editTextMobile"
    android:layout_centerHorizontal="true"
    android:layout_marginTop="20dp"
    android:text="@string/register"
    android:textSize="18sp"
    android:textColor="@color/black"
    android:backgroundTint="@android:color/darker_gray"
    android:fontFamily="sans-serif-medium"
    android:elevation="8dp" />

</RelativeLayout>
```

In this experiment just we are going to see how to use and implement Notification for our application. For displaying the notification there are four important concepts that we have to know.

1. NOTIFICATION CHANNEL

2. NOTIFICATION BUILDER

3. NOTIFICATION MANAGER

4. PENDING INTENT

MainActivity.java

```
///Steps for Creating the Notifications
///STEP 1: Create a Notification Channel by method and declare the variables globally that is channel id and name
///STEP 2: Create a Notification Builder check for the API version
///Check for if(CURRENT_API>=O ) && if the android version is 13 or above you have set permissions in manifest file
///STEP 3: After the builder is created we have to create the pending intent to navigate to our activity that we have created
///Finally run the application->click the home button->You will see the notification->when clicking the notification it should be navigated to the target activity

package com.example.notification

import android.annotation.SuppressLint
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.PendingIntent
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.graphics.Color
import android.os.Build
import android.os.Bundle
import android.widget.Button
import androidx.activity.enableEdgeToEdge
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.NotificationCompat
import androidx.core.app.NotificationManagerCompat

class MainActivity : AppCompatActivity() {
    val NOTIFICATION_CHANNEL_ID = "my_channel_id"
    val NOTIFICATION_CHANNEL_NAME = "my_channel_name"
    val NOTIFICATION_ID = 1
```

```
    @SuppressLint("MissingPermission")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        enableEdgeToEdge()
        setContentView(R.layout.activity_main)
        createNotificationChannel()
        ///Creating Pending Intent
        val intent=Intent(this,MainActivity2::class.java)

        ///...This is used for the older API versions
//        val pendingIntent=TaskStackBuilder.create(this).run {
//            addNextIntentWithParentStack(intent)
//            getPendingIntent(0,PendingIntent.FLAG_UPDATE_CURRENT)
//        }

        ///...This is used for the modern API versions
        val pendingIntent = PendingIntent.getActivity(this, 0, intent,
PendingIntent.FLAG_IMMUTABLE)

        ///NOTE: Android 13 and Higher versions require Post Notifications permission
to be granted

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
            if
(checkSelfPermission(android.Manifest.permission.POST_NOTIFICATIONS)
!= PackageManager.PERMISSION_GRANTED) {

requestPermissions(arrayOf(android.Manifest.permission.POST_NOTIFICATI
ONS), 1)
            }
        }

///...After the Notification is created for showing notification we have to Build the
Notification with NotificationCompat.Builder
        ///NOTIFICATION BUILDER
        val notification = NotificationCompat.Builder(this,
NOTIFICATION_CHANNEL_ID)
            .setContentTitle("MBU Id Registered Successfully")
            .setContentText("Click here to explore")
```

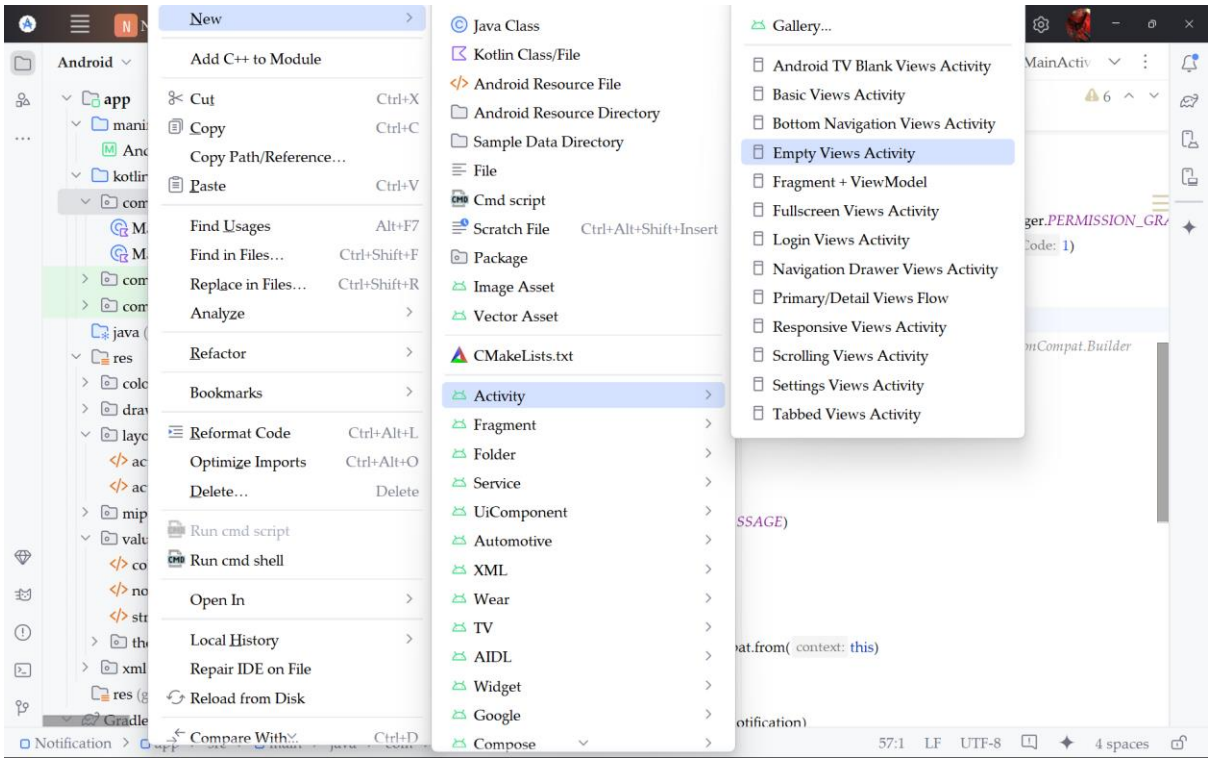
```
            .setSmallIcon(R.drawable.notification_foreground)
            .setPriority(NotificationCompat.PRIORITY_HIGH)
            .setCategory(NotificationCompat.CATEGORY_MESSAGE)
            .setContentIntent(pendingIntent)
            .setAutoCancel(true)
            .build()

        val notificationManager = NotificationManagerCompat.from(this)
        val button = findViewById<Button>(R.id.button)
        button.setOnClickListener {
            notificationManager.notify(NOTIFICATION_ID, notification)
        }
    }

    ///...This is used to create the notification channel
    @SuppressLint("ObsoleteSdkInt")
    private fun createNotificationChannel() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            val channel = NotificationChannel(
                NOTIFICATION_CHANNEL_ID,
                NOTIFICATION_CHANNEL_NAME,
                NotificationManager.IMPORTANCE_HIGH
            ).apply {
                lightColor = Color.GREEN
                enableLights(true)
            }
            val manager = getSystemService(Context.NOTIFICATION_SERVICE) as
NotificationManager
            manager.createNotificationChannel(channel)
        }
    }
}
```

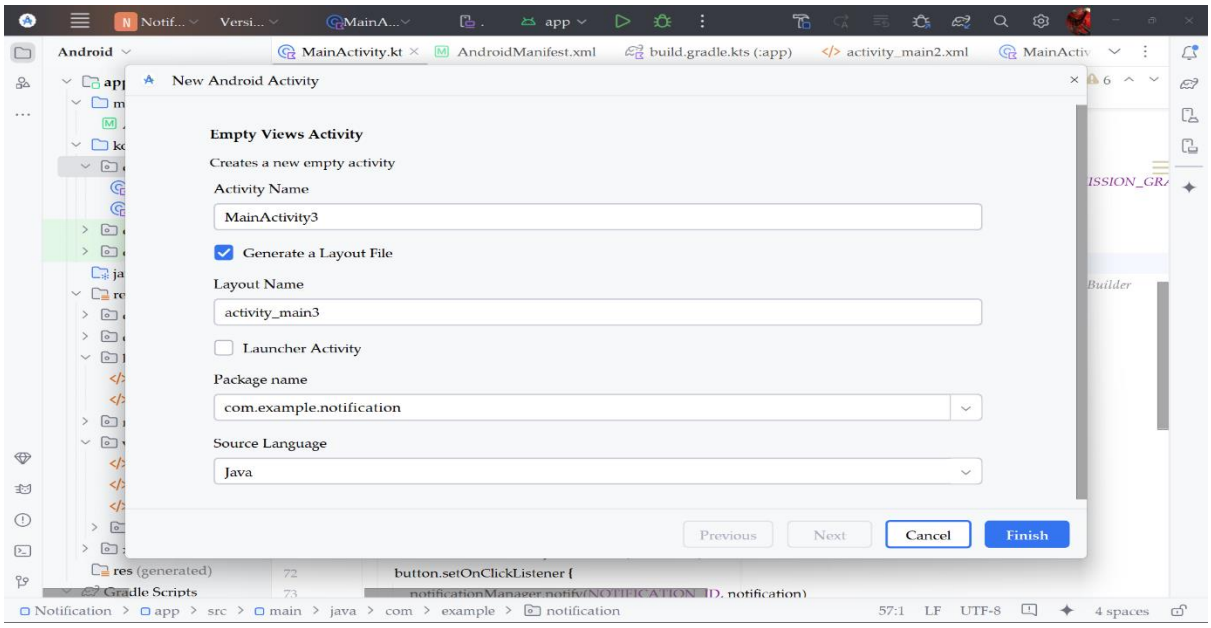
Here the first phase of the experiment is completed for creating the custom button. Simply we have designed the button and displayed a message.

NOTE: Convert the Kotlin code to Java and configure the project according to the requirements.

The second Phase is to Handling the displayed message. For this simply you can create another **Empty Views Activity by right clicking your package name**-> After that you can configure your second activity as your wish.

Note: If you are clicking the Notification it should navigate to the newly created activity.





In this experiment I'm using Web view as the Second Activity

activity_main2.xml

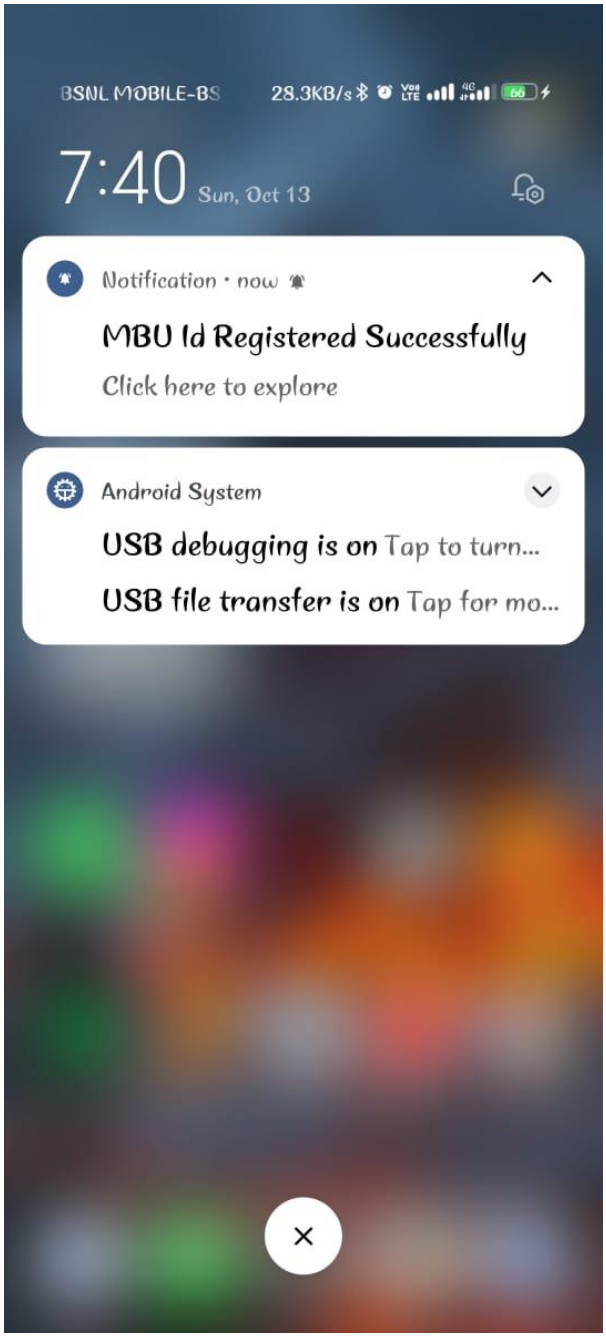
```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:id="@+id/main"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity2">

    <WebView
        android:id="@+id/oklWebView"
        android:layout_width="match_parent"
        android:layout_height="match_parent" />
</RelativeLayout>
```

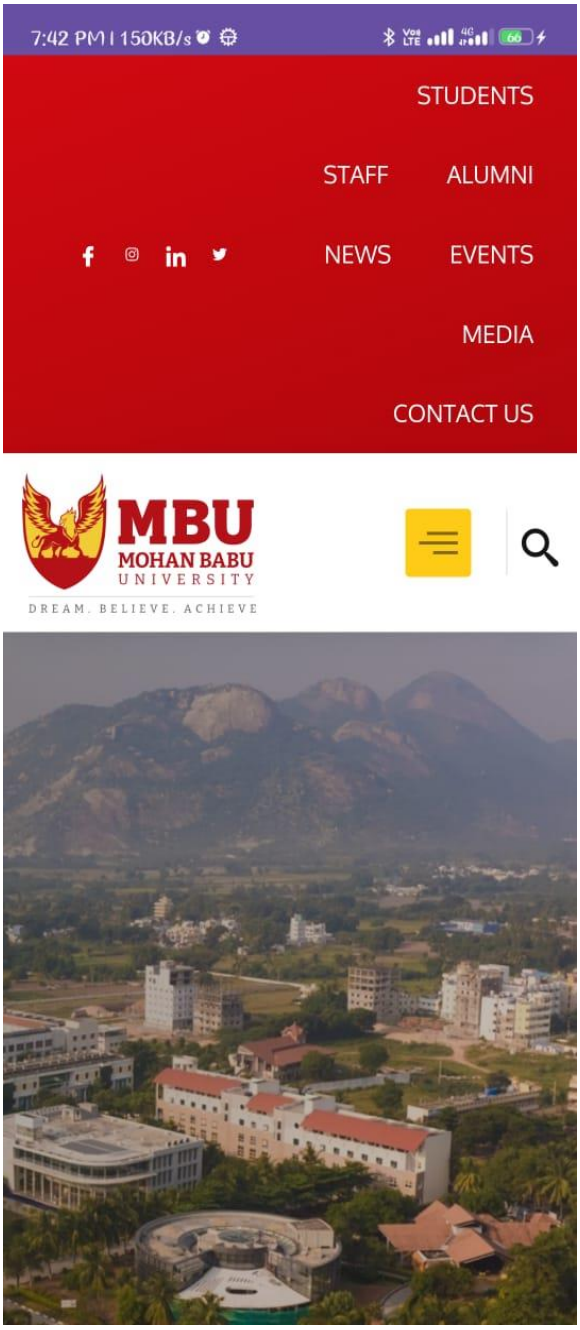
MainActivity2.kt

```
package com.example.notification

import android.annotation.SuppressLint
import android.os.Bundle
import android.webkit.WebView
import android.webkit.WebViewClient
import androidx.appcompat.app.AppCompatActivity

class MainActivity2 : AppCompatActivity() {
    @SuppressLint("SetJavaScriptEnabled")
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main2)

        val url=intent.getStringExtra("url")

        val webView=findViewById<WebView>(R.id.oklWebView)
```

```
        webView.webViewClient=WebViewClient()
        webView.settings.javaScriptEnabled=true
        webView.loadUrl("https://www.mbu.asia/")
    }
}
```

OUTPUT:





RESULT:

Finally, the program for implementing a Custom Button and Handle the displayed message on the Button Press was implemented and Executed Successfully.