Dc .;'."???///?;Trainer : Prashant Ranjan
Connect : prashant.ranjan@koenig-solutions.com
prashant.koenig@gmail.com
+91-8800342255

Course Materials:

1. Where is everything : goo.gle/3iTdWbP
2. Pathways/CodeLabs: goo.gle/3xwL2Xq

Step 1:

1. Create a Google developer profile : goo.gle/30orQvC
2. Make your profile public.
    a. Click on **gear** icon settings.
    b. Select profile visibility: Update it to Public.

Prerequisite:
System : Windows, MAC, Linux
RAM: Min 8 GB
Storage: SSD preferred
Processor: i5

Tools:

1. IntelliJ : Working with Kotlin : https://www.jetbrains.com/idea/download/download-thanks.html?platform=windows&code=IIC

2. Android Studio: Work with Android : https://developer.android.com/studio

Demo: Create Hello World with Kotlin

1. Open IntelliJ
2. Create a new project
    a. Project Name: **Hello Kotlin**
    b. Category: **Kotlin**
    c. Template: **JVM** template
    d. Select the JDK version.
    e. Finish to create the same.
3. It will take time to build the project (first time)

4. Tools Menu > Kotlin > REPL
    a. Provide below details.
        print("Hello world")

    b. To run the same, click play icon / **CTRL + ENTER.**

Variable & Constant

The keyword "**var"** is used to create variables.
Variables,data are allowed to manipulate.
It is also known as **mutable (**changeable).

The keyword **"val"** is used to create a constant.
Constant values are not allowed to be manipulated.
It is also known as **Immutable** (unchanged)

Working with Functions

1. **Select Project >src>main>kotlin :** Right Click > New > kotlin file/class
    a. Name: Hello.kt
    b. Type: **File**
2. Add below statement for main function.

```
fun main(args: Array<String>) {
    print("Hello This is Main function Kotlin")

    }
```

Day 02: Class & Objects

1. **class** keyword is used to create a class.
2. To create instances we don't use **new** keywords.

Working with Constructor:

1. **Class with Parameters : Used**
2. **Init : Used**
3. **constructor keyword**

Inheritance:

Make sure Base class/ Parent Class/ Super Class must be as **Open.** Else default type is as **final.**

Android Application Development

- **Create Hello Android**
- **Files & Folder**
- **Android Version**
- **Lifecycle Methods**

Creating Hello World

1. Open Android Studio
2. **File Menu / Create** a new project and provide below of details.
    a. **Choose Category**: Phone and Tablet
        i. **Template**: Empty Activity

    b. **Project Name**: Name of Project
    c. **Package Name** Package for project (Important part for deployment)
    d. **Location**
    e. **Language** Kotlin
    f. **Min SDK version** 4.1
    g. Finish to create project

3. **Execution of Android app**
    a. **Using Android AVD**
    b. Tools Menu > AVD (Android Virtual Device Manager) -> it will show the list of Virtual devices whichever is available.
        i. If no devices available create a new one.
            1. Select the screen size
            2. Provide Operating system **Q**
            3. Provide name
            4. Create
        ii. Once Created, From Android Device Manager, we can start the same.

Important Points:

A
B
C : Cupcake
D : Donut
E : Eclairs
F : Froyo

G	**Gingerbread**		
H	**HoneyComb**		
I	**Ice Cream Sandwich**	**Android 4.0**	
J	**JellyBean**	**Android 4.2**	**:** Android Wear
K	**Kitkat**	**Android 4.4**	
L	**Lollypop**	**Android 5**	**:** Material Design
M	**Marshmallow**	**Android 6**	**:** Run time permissions
N	**Nougat**	**Android 7**	
O	**Oreo**	**Android 8**	
P	**Pie**	**Android 9**	
Q		**Android 10**	
R		**Android 11**	

Files & Folder of Android

1. **manifest**
    a. AndroidManifest.xml : It's a configuration file that will maintain permission, number of activities, launcher activity & others.
2. **Java**
   It's also known as the Compiled Zone. Files kept inside will be compiled.
    a. **Package Name(default)**
        i. **MainActivity.kt:** It will be containing information of event handler of UI Components
    b. **Package Name UITest:** UI Testing (espresso)
    c. **Package Name Test :** Unit Testing (JUnit)

3. **Res**
    a. **Drawable:** Used to place images.
    b. **Layout :** User Interface
    c. **Mipmap :** Used to place images with different resolutions, mostly for application icons.
    d. **Values:** Useful for localization

Day 03

Android Components

1. **Activity** : Used to build User Interfaces for end users.
2. **Broadcast Receiver:** App to App communication
3. **Services** : Long running background process

Note: For above three, Intent **is a class that allows you to perform operations.**

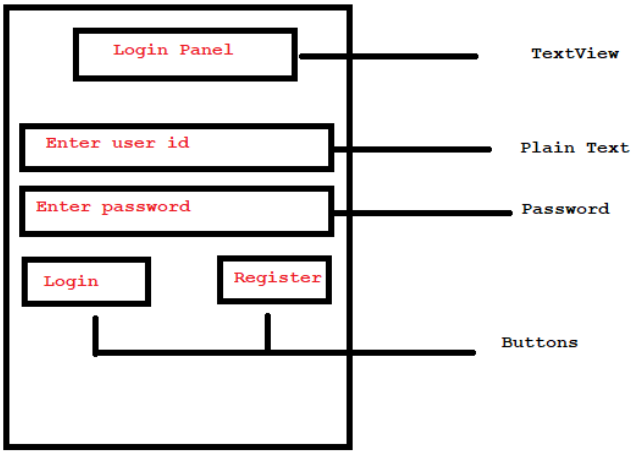
4. **Content Provider**: Share data

<u>Working with Layout Constraints:</u>

Constraint Layout, is a layout that allows to place UI components + fix the position of UI Component based on constraints.

There are four constraints that can be added.

- Minimum 2 constraints can be added to UI Element (**X , Y position)**
- Maximum 3 constraints can be added.
- Rare Cases, we can go for 4 constraints.



Demo:
Create an application named "Login App.
Update the User interface as per fig.

app>layout>activity_main.xml

1. **Drag below of UI Components.**
    a. **Text View**: Display Value as text
    b. **PlainText:** To take input as text
    c. **Password**: To take input as text, to display with dotted values.
    d. **Button** : Handling Login Event
    e. **Button** : Handle Register event.

2. **Constraint Addition:**
    a. **Login Panel**
        i. Top : 16
        ii. Left : 16
        iii. Right : 16

    b. Plain Text, Password
        i. Top: 40
        ii. Left: 16
        iii. Right: 16
        iv. layout_width= 0dp match_constraint
        v. layout_height=50dp
    c. Button Login
        i. Top: 20
        ii. Right : 16

d. Button Register
   i. Top : 20
   ii. Left: 16

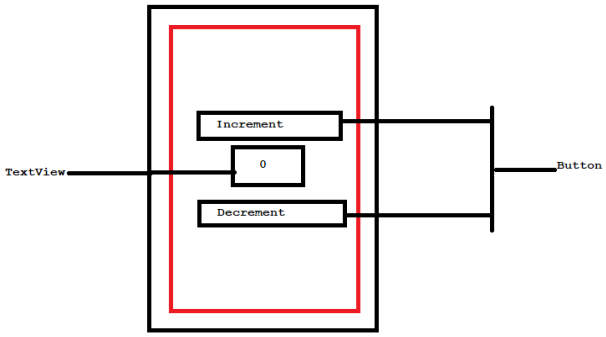
Event Handling:
Demo: Create a new project named “Counter-App”.

Steps:

1. **app>layout>activity_main.xml**
2. **Add below of UI Component**
   a. **Text View**
      i. Top: 16
      ii. Left : 16
      iii. Right: 16
      iv. Bottom: 16

      v. Text: 0
      vi. Text size: 40sp
      vii. Color: Black

      viii. Background Color : #A19A9A
      ix. Padding : 100dp



   b. Button:
      i. Bottom : Connect to Text View (8)
      ii. Right: 40
      iii. Left: 40
      iv. Layout_width: match_constraint
      v. Layout_height: 60dp

      vi. Text: Increment
   c. Button
      i. Top: Align with text View (8)
      ii. Right: 40
      iii. Left : 40
      iv. Layout_width: match_constraint
      v. Layout_height: 60dp

      vi. Text: Decrement

Connecting layout UI Component with Activity file (Source Code) for the logic.

1. Define an ID attribute to those UI Components which will be the part of logic.

2. Inside the Kotlin class file, create an object of the respective UI component and Link it with **findViewByID().**

Update ID for

1. **Text vlew** **:** resultTV
2. **Button** **:** btnInc
3. **Button** **:** btnDec

3. Update MainActivity>onCreate as below

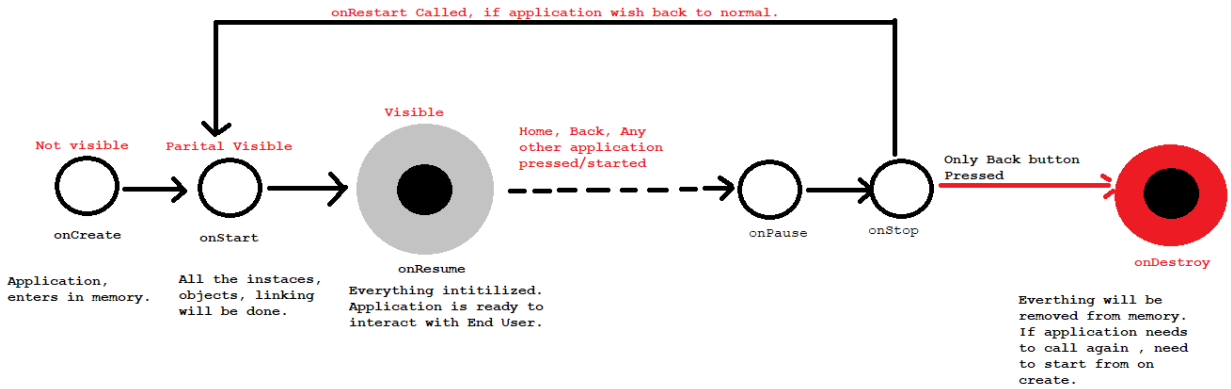
3.1 Add below of libraries
import android.widget.Button
import android.widget.TextView

Logics

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


      //1: Linking UI components
      var resultTVObject : TextView = findViewById(R.id.resultTV)
      val btnIncObject : Button = findViewById(R.id.btnInc)
      val btnDecObject : Button = findViewById(R.id.btnDec)


      //2:
      var counter = 0;

      //3: Attach Listener to button object
      btnIncObject.setOnClickListener {
         counter++

         //3.1: Update the value with text view object
         resultTVObject.text = "$counter"
      }

      //4:
      btnDecObject.setOnClickListener {
         counter--

         //4.1: Update the value with text view object
```

```
        resultTVObject.text = "$counter"
    }
}
```

Activity Life Cycle:



Working with Multi Activities:

Create two activities in the Login-App demo.

1. Right click **app** >new> Activity > Empty Activity.
    a. Provide the details
    b. Finish.

Add two activities as below

1. AccountActivity
2. RegisterActivity

Attach Event Listener for buttons Login / Register

1. Add an ID to each of the buttons.
    a. Login : btnLogin
    b. Register: btnRegister

MainActivity.kt

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

        //Linking of UI Component
        val loginButton : Button = findViewById(R.id.btnLogin)
```

```
        val regButton : Button = findViewById(R.id.btnRegister)


        loginButton.setOnClickListener {
            val accountIntent = Intent(this,AccountActivity::class.java)
            startActivity(accountIntent)
        }
        regButton.setOnClickListener {
            val registerIntent = Intent(this,RegisterActivity::class.java)
            startActivity(registerIntent)
        }
    }


}
```

Update: Make the Account Activity to be logged in , only in case where USER ID= admin and PASSWORD = admin.

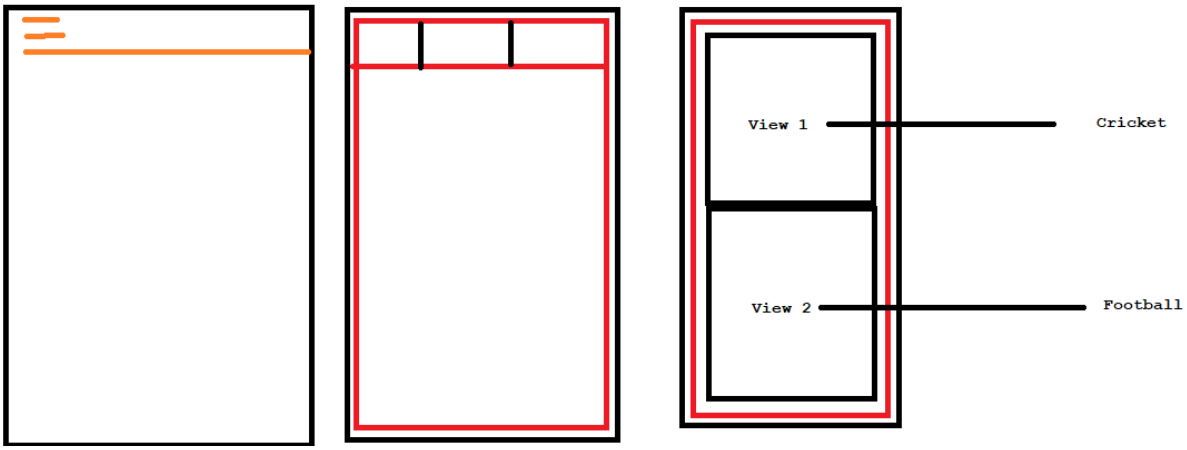
```
class MainActivity : AppCompatActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)

        //Linking of UI Component
        val loginButton : Button = findViewById(R.id.btnLogin)
        val regButton : Button = findViewById(R.id.btnRegister)

        var usrTXTObj : EditText = findViewById(R.id.userTXT)
        var passTXTObj : EditText = findViewById(R.id.passTXT)


        loginButton.setOnClickListener {

            if(usrTXTObj.text.toString().equals("admin") &&
passTXTObj.text.toString().equals("admin")) {
                val accountIntent = Intent(this, AccountActivity::class.java)
                startActivity(accountIntent)
            }else{
                Toast.makeText(this,"Unsuccessful",Toast.LENGTH_SHORT).show()
            }
        }
        regButton.setOnClickListener {
```

```
        val registerIntent = Intent(this,RegisterActivity::class.java)
        startActivity(registerIntent)
    }
  }


}
```

Day 04 : Google Android Kotlin Fundamentals

Fragments:
Fragment is a type of SubActivity that requires an activity to perform operation. It has its own lifecycle.

Type of Application
1. Navigation Drawer Based Application
2. Tab Based application
3. Multiple Data Source Application
4. & others.



Demo: Working with Fragment

1. Create a project named “Hello-Fragment”.
2. Activity_main.xml
    a. Drag a frame layout.
        i. Top: 8
        ii. Left: 8
        iii. Right: 8

iv. Bottom:8
v. Layout_width : match_constraint
vi. Layout_height: match_cosntraint
vii. Id: myFrame

3. Add a blank fragment
    a. File/app-> Right Click /Click>New > Fragment> Fragment Blank
        i. Name: Login Fragment.

4. Update the layout of fragments as **Constraint** instead **Frame Layout.**
    a. layout>fragment_login.xml
        i. Right click on **Frame Layout> Convert it to Constraint.**
        ii. Click the agreement
        iii. Remove the id of Frame Layout
        iv. Delete any Existing UI Component.

    b. Add below of UI Components
        i. Text View
            1. Top: 20
            2. Left : 16
            3. Right : 16
            4. Text : Login Panel

        ii. PlainText
            1. Top : 40
            2. Left: 16
            3. Right: 16
            4. layout_width=”match_constraint”
            5. Hint: Please enter user id
            6. Text: Remove it.
        iii. Password:
            1. Top :40
            2. Left: 16
            3. Right: 16
            4. layout_width=”match_constraint:
            5. Hint: Please enter password
            6. Text: remove it.
        iv. Button
            1. Top: 32
            2. Left: 16
            3. Text: Register
        v. Button:
        vi. 1. Top: 32
        vii. 2. Right: 16

viii. Left: 0

fragmentLogin.kt

```
class LoginFragment: Fragment(){

    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        var fragmentView : View = inflater.inflate(R.layout.fragment_login,container,false)
        return fragmentView
    }
}
```

Linking of Fragment with Activity

java>pacakge>MainActivity

1. **Override the on resume lifecycle.**

```
    override fun onResume() {
        super.onResume()

        //1: Create object of Fragment Transaction
        val fragmentTransaction : FragmentTransaction =
supportFragmentManager.beginTransaction()

        //2: Replace the view from Frame layout with new fragment
        fragmentTransaction.replace(R.id.myFrame,LoginFragment())

        //3: Commit the trasnaction
        fragmentTransaction.commit()
    }
```

Update with Demo:

Add two more fragment

1. **Account Fragment**

```
class AccountFragment : Fragment() {


    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        val accountView: View = inflater.inflate(R.layout.fragment_account,container,false);
        return accountView;
    }


}
```

2. **Register Fragment**

```
class RegisterFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {


        val registerView: View = inflater.inflate(R.layout.fragment_register,container,false);
        return registerView;
    }

}
```

Linking of Account + Register fragment with Login Fragment

1. **Define an id to each button**
2. **Link with LoginFragment**

Update with Login Fragment

```
class LoginFragment: Fragment(){
```

```
    override fun onCreateView(
        inflater: LayoutInflater,
        container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {

        var fragmentView : View = inflater.inflate(R.layout.fragment_login,container,false)

        //Linking UI Components

        val LoginBT : Button = fragmentView.findViewById(R.id.btnLOGIN)
        val registeBT: Button = fragmentView.findViewById(R.id.btnRegister)

        val fragmentTransaction: FragmentTransaction =
parentFragmentManager.beginTransaction()

        //Event Handler
        LoginBT.setOnClickListener {

            fragmentTransaction.replace(R.id.myFrame,AccountFragment())
            fragmentTransaction.commit()
        }
        registeBT.setOnClickListener {
            fragmentTransaction.replace(R.id.myFrame,RegisterFragment())
            fragmentTransaction.commit()
        }

        return fragmentView
    }
}
```

Demo: Create an Application as Tab Based application using Fragments.

1. Create a project named “Tabbed-Demo”.
2. Add three fragments
    a. RedFragment : Red Screen
    b. GreenFragment : Green Screen
    c. BlueFragment : Blue Screen
3. Add ViewPager inside **activity_main_xml**
    a. Top:0
    b. Left:0
    c. Right:0
    d. Bottom:0

e. **Layout_width: match_constraint**
f. **Layout_height: match_constraint**

<u>Activity_main.xml : In Code View</u>

```
<?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">

    <androidx.viewpager.widget.ViewPager
        android:id="@+id/myPager"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

RedFragment

Layout:

```
<?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"
    android:background="#E91E63"
    tools:context=".RedFragment">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:text="Red Screen"
        android:textColor="@color/white"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Kotlin

```
class RedFragment : Fragment() {


    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_red, container, false)
    }

}
```

Green Fragment
Layout:

```
<?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"
    android:background="#CDDC39"
    tools:context=".GreenFragment">

    <TextView
```

```
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:text="Green Screen"
        android:textColor="@color/white"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Kotlin

```
class GreenFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_green, container, false)
    }


}
```

Blue Fragment

Layout:

```
<?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"
```

```
    android:background="#3F51B5"
    tools:context=".BlueFragment">

    <TextView
        android:id="@+id/textView4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="24dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:text="Blue Screen"
        android:textColor="@color/white"
        android:textSize="36sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Kotlin

```
class BlueFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.fragment_blue, container, false)
    }

}
```

Create an Adapter for view pager with all three fragment.

1. Create a class named “MyAdapter”.
    a. java> Package>Right Click > new > kotlin file/class
        i. MyAdapter

```
class MyAdapter(fragmentManager: FragmentManager) :
FragmentPagerAdapter(fragmentManager) {
```

```
    override fun getCount(): Int {
        return 3
    }

    override fun getItem(position: Int): Fragment {
        when(position){
            0->return  RedFragment()
            1->return GreenFragment()
            2->return BlueFragment()
        }
        return  RedFragment()
    }

}
```

Update with MainActivity.kt

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

        //Linking
        val myPagerObj:ViewPager = findViewById(R.id.myPager);
        myPagerObj.adapter = MyAdapter(supportFragmentManager)
    }
}
```

Remove Deprecation
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">

    <androidx.viewpager2.widget.ViewPager2
        android:id="@+id/myPager"
        android:layout_width="0dp"
        android:layout_height="0dp"
```

```
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.kt

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

        //Linking
        val myPagerObj:ViewPager2 = findViewById(R.id.myPager);
       myPagerObj.adapter = MyAdapter(this)
    }
}
```

MyAdapter.kt

```
class MyAdapter(fragmentActivity: FragmentActivity) : FragmentStateAdapter(fragmentActivity){
    override fun getItemCount(): Int {
        return 3
    }

    override fun createFragment(position: Int): Fragment {
       when(position){
           0-> return RedFragment()
           1->return GreenFragment()
           2->return BlueFragment()
       }
        return RedFragment()
    }

}
```

RecyclerView:
Introduced in **Android 5.0** with the feature of dynamic loading with less impact.

Demo: Create a RecyclerView.

1. Create a project named “Recycler-View-Demo”.
2. **Inside activity_main.xml**
    a. Add a recycler_view
        i. Top: 8
        ii. Right: 8
        iii. Left : 8
        iv. Bottom: 8

        v. layout_height : 0dp match_constraint
        vi. layout_width: 0dp match_constraint

        vii. Id: myRecycler

3. **Add a layout.**
    a. Right click layout > new > layout resource file
        i. Name: recycler_row_layout

```
<?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="wrap_content">

   <ImageView
      android:id="@+id/productImageView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="8dp"
      android:layout_marginLeft="8dp"
      android:layout_marginTop="8dp"
      android:layout_marginBottom="8dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      tools:srcCompat="@tools:sample/avatars" />
```

```
<TextView
    android:id="@+id/productNameTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="10dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Product Name"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/productImageView"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/productPriceTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Product Price"
    android:textColor="@color/black"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/productImageView"
    app:layout_constraintTop_toBottomOf="@+id/productNameTV" />

<TextView
    android:id="@+id/productCountryTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:layout_marginBottom="8dp"
    android:text="Product Made"
```

```
        android:textColor="@color/black"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/productImageView"
        app:layout_constraintTop_toBottomOf="@+id/productPriceTV" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Creating Recycler Adapter

1. **Java>package default> Right Click > new>Kotlin/class**

   MyAdapter

```
class MyAdapter : RecyclerView.Adapter<MyAdapter.ViewHolder>() {

    //Step1: Prepare Data
    val productNameList = listOf<String>("Cartoon1","Cartoon2","Cartoon3","Cartoon4")
    val productPriceList = listOf<Int>(100,310,21,41)
    val productCountryList = listOf<String>("India","USA","UK","Germany")


    //It will link the UI Component(RecyclerView Row) with Adapter using findViewBYId
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        var productIV: ImageView
        var productName: TextView
        var productPrice: TextView
        var productCountry: TextView


        init {
            productIV = itemView.findViewById(R.id.productImageView)
            productName = itemView.findViewById(R.id.productNameTV)
            productPrice = itemView.findViewById(R.id.productPriceTV)
            productCountry = itemView.findViewById(R.id.productCountryTV)
        }
    }

    //It will Link with Layout ( RecyclerView Row)
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyAdapter.ViewHolder
{
        val recycler_row_view =
LayoutInflater.from(parent.context).inflate(R.layout.recycler_row_layout,parent,false)
        return ViewHolder(recycler_row_view)
```

```
    }

    //IT will link data with UI Component ( RecyclerView Row)
    override fun onBindViewHolder(holder: MyAdapter.ViewHolder, position: Int) {
        holder.productName.text = "Name: ${productNameList[position]}"
        holder.productPrice.text ="Price: ${productPriceList[position]} INR"
        holder.productCountry.text = "Made: ${productCountryList[position]}"
    }

    //Size of Data
    override fun getItemCount(): Int {
        return productCountryList.size
    }
}
```

MainActivity

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

        //Linking of Recycler View
        var myRecyclerView :RecyclerView = findViewById(R.id.myRecyclerView)

        myRecyclerView.adapter = MyAdapter()
        myRecyclerView.layoutManager = LinearLayoutManager(this)

myRecyclerView.addItemDecoration(DividerItemDecoration(this,LinearLayoutManager.VERTIC
AL))
    }
}
```

Day 05:

Android Navigation

Android Navigation is a part of JetPack library. It's used to graphically represent the navigation mechanism with Fragments.

Demo: Create an application with named “**Navigation-App-Demo”**

1. Add two fragments.
    a. FirstFragment
    b. Second Fragment

2. **layout>fragment_first.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"
   android:background="#3e3e3e"
   tools:context=".FirstFragment" >

   <TextView
      android:id="@+id/textView2"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:layout_marginStart="16dp"
      android:layout_marginLeft="16dp"
      android:layout_marginTop="32dp"
      android:layout_marginEnd="16dp"
      android:layout_marginRight="16dp"
      android:text="First Fragment"
      android:textColor="@color/white"
      android:textSize="30sp"
      android:textStyle="bold"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

   <Button
      android:id="@+id/button"
      android:layout_width="0dp"
      android:layout_height="80dp"
      android:layout_marginStart="32dp"
      android:layout_marginLeft="32dp"
      android:layout_marginTop="32dp"
      android:layout_marginEnd="32dp"
      android:layout_marginRight="32dp"
      android:text="Go To Next"
```

```
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView2" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. **layout>fragment_second.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"
    android:background="#7DB140"
    tools:context=".SecondFragment">

    <TextView
        android:id="@+id/textView3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:text="Second Screen"
        android:textColor="@color/white"
        android:textSize="30sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

4. C**reate a Navigation Resource file**
    a. **Right click > res folder> new > Android Resource File**
        i. Name of file: main_nav
        ii. Type: Navigation

    b. It will ask for downloading libraries, click yes to proceed.
    c. Once successfully done, there will be a folder named **navigation**, within it **main_nav.xml** file can be found.

5. Define Fragments inside the Navigation Component.
6. **res>navigation>main_nam**
    a. **Add +**
        i. Select both fragments and arrange it.
        ii. By highlighting the circle for the **First fragment,** connect with another fragment.
            1. There will be **an id generated** for the connection.
7. Connect main_nav.xml for activity_main.
    a. Delete all other existing UI Components from activity_main.xml
    b. Drag **NavHostFragment** select the **main_nav** and drop it.
        i. Top :0
        ii. Right: 0
        iii. Left: 0
        iv. Bottom: 0

8. Update with first fragment layout.
    a. Add an **ID** to button Next
    b. Link ID with Class file.

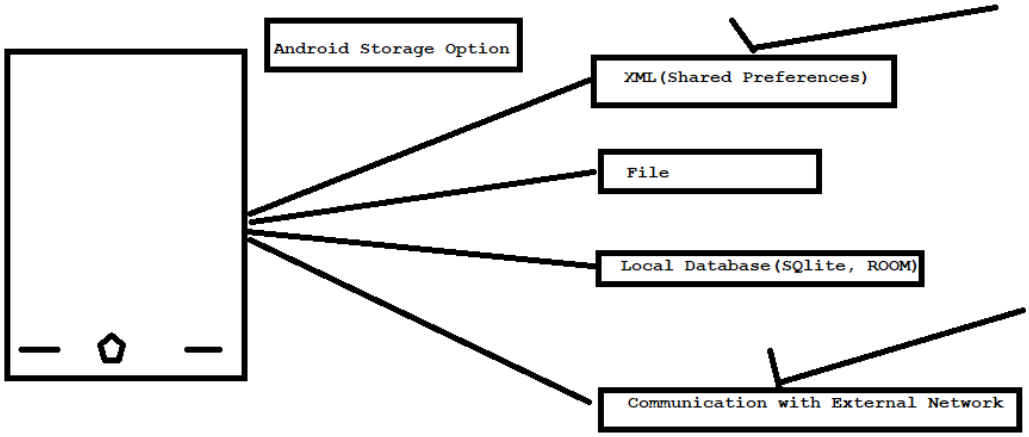
FirstFragment

```
class FirstFragment : Fragment() {

    override fun onCreateView(
        inflater: LayoutInflater, container: ViewGroup?,
        savedInstanceState: Bundle?
    ): View? {
        // Inflate the layout for this fragment

        val firstView: View = inflater.inflate(R.layout.fragment_first, container, false);

        val btnNEXTBT: Button = firstView.findViewById(R.id.btnNEXT);

        btnNEXTBT.setOnClickListener {

NavHostFragment.findNavController(this).navigate(R.id.action_firstFragment2_to_secondFragment2)
        }

        return firstView
    }
```

}

9. NavController ID is the id which is defined inside **main_nav**
10. Run application and test the same.

Android Storage Option



SharedPreferences:

Shared preferences allows to store data in form of XML Key Pair value. Where the value will be stored basis of key. Similar key will be used to access the value.

It will always store latest value.

Demo: Create a project with below of UI Components

layout>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">
```

```
<TextView
    android:id="@+id/textView"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:text="Registeration"
    android:textColor="@color/black"
    android:textSize="30sp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<EditText
    android:id="@+id/nameINPUT"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:ems="10"
    android:hint="Please enter name"
    android:inputType="textPersonName"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView" />

<EditText
    android:id="@+id/ageINPUT"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="16dp"
    android:layout_marginLeft="16dp"
    android:layout_marginTop="32dp"
    android:layout_marginEnd="16dp"
    android:layout_marginRight="16dp"
    android:ems="10"
```

```
        android:hint="Please enter age"
        android:inputType="number"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/nameINPUT" />

    <EditText
        android:id="@+id/addressINPUT"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:ems="10"
        android:hint="Please enter address"
        android:inputType="textPersonName"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ageINPUT" />

    <Button
        android:id="@+id/button"
        android:layout_width="0dp"
        android:layout_height="60dp"
        android:layout_marginStart="32dp"
        android:layout_marginLeft="32dp"
        android:layout_marginTop="32dp"
        android:layout_marginEnd="32dp"
        android:layout_marginRight="32dp"
        android:text="Submit"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/addressINPUT" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

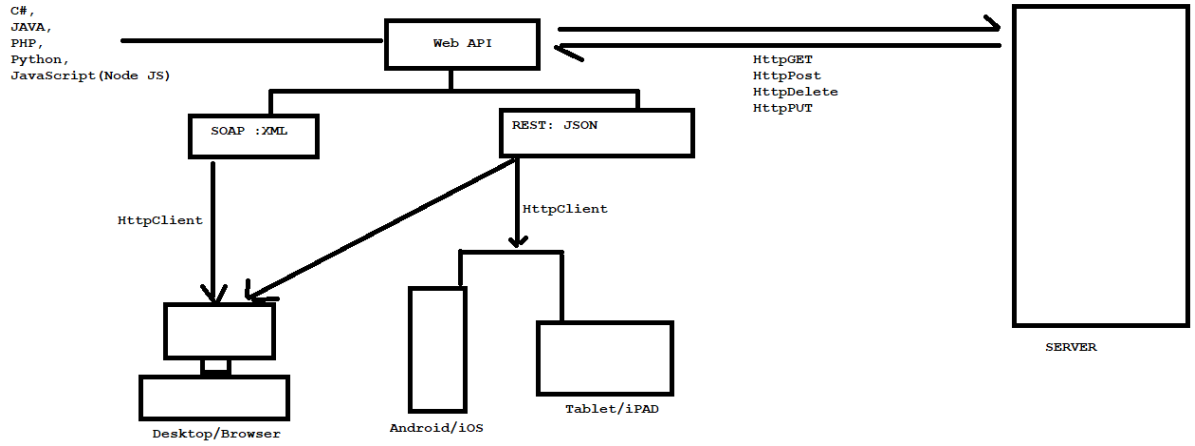
MainActivity.kt

```
class MainActivity : AppCompatActivity() {

    var nameED : EditText? = null
```

```
    var ageED: EditText? = null
    var addressED: EditText? = null
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContentView(R.layout.activity_main)
    }

    override fun onStart() {
        super.onStart()

        nameED = findViewById(R.id.nameINPUT);
        ageED = findViewById(R.id.ageINPUT)
        addressED = findViewById(R.id.addressINPUT)
    }

    //Procedure to save data withing Preferences
    override fun onPause() {
        super.onPause()

        //1: Create a Shared Preferences Object
        val sharedPreferences :SharedPreferences = this.getSharedPreferences("myPrefs",
Context.MODE_PRIVATE)

        //2: Create and SharedPreferences Editor to edit the file.
        val editor : SharedPreferences.Editor = sharedPreferences.edit()

        //3.Add the data
        editor.putString("nameKey",nameED!!.text.toString())
        editor.putString("ageKey",ageED!!.text.toString())
        editor.putString("addressKey",addressED!!.text.toString())

        //4: Save Operation
        editor.apply()
    }

    override fun onResume() {
        super.onResume()

        //1: Create a Shared Preferences Object
        val sharedPreferences :SharedPreferences = this.getSharedPreferences("myPrefs",
Context.MODE_PRIVATE)

        //2: Reterive the value
        val nameValue = sharedPreferences.getString("nameKey",null)
```

```
        val ageValue = sharedPreferences.getString("ageKey",null)
        val addressValue = sharedPreferences.getString("addressKey",null)

        //3:Check the value
        if(nameValue != null && ageValue!= null && addressValue!= null){
            //3.1 Update the value with Edit Text US component

            nameED!!.setText(nameValue.toString())
            ageED!!.setText(ageValue.toString())
            addressED!!.setText(addressValue.toString())
        }

    }
}
```

Web API



Web API creates an Interface through which any platform based device can communicate. Using HTTPClient API can be consumed for the operation for different devices.

Types of API

1. **SOAP :** SOAP API does the communication using XML style. Which is heavy in architecture.
2. **REST:** It's one of the popular API and perfect for smaller devices. Communication made using JSON.

Web API Creation

There are many languages which supports to write WEB API.

1. C#
2. JAVA
3. PHP
4. PYTHON
5. JavaScript(Node JS)

Type of OPeration

1. **HTTPGet** : Get Records single/multi
2. **HTTPPost** : Add Record
3. **HTTPPut** : Update Record based on ID
4. **HTTPDelete** : Delete Record based on ID

Android Volley

Volley library introduced to work network based operation.

Demo: Create a project to work with Web API

1. Create a project named “Web-API-Demo”
2. Add the volley library.
    a. Check for **GradleScripts**
        i. Select **build.gradle(Module: ProjectName.app)**
        ii. Check for **dependencies** tag.
        iii. Add below of line
        Implementation "com.android.volley:volley:1.2.0"

        iv. Click on **Sync Now** to make the changes.

3. **MainActivity.kt**

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


   override fun onResume() {
      super.onResume()

      //1: Create volley object
```

```
        val queue = Volley.newRequestQueue(this)
        val fetchURL = "https://jsonplaceholder.typicode.com/posts"

        //2: Create a String Request to connect with API and fetch data and provide response as string
        val fetchRequest = StringRequest(Request.Method.GET,fetchURL,
                Response.Listener<String> {response ->
                    Log.d("Response",response.toString())
                },
                Response.ErrorListener {error ->
                    Log.d("Response Error",error.toString())
                })

        //3: Add the request inside volley object
        queue.add(fetchRequest)
    }
}
```

4. Add **Internet** Permission for the application
    a. manifest> Android Manifest.xml
    b. Add the below line just before the **<application>** tag.

```
<uses-permission android:name="android.permission.INTERNET"/>
```

Display the Data Inside Recycler View.

1. **Update the activity_main.xml file**

```
<?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">

    <TextView
        android:id="@+id/textView"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginStart="16dp"
        android:layout_marginLeft="16dp"
        android:layout_marginTop="16dp"
```

```
        android:layout_marginEnd="16dp"
        android:layout_marginRight="16dp"
        android:text="REST API DATA"
        android:textColor="@color/black"
        android:textSize="24sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/myRecycler"
        android:layout_width="0dp"
        android:layout_height="0dp"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:layout_marginEnd="8dp"
        android:layout_marginRight="8dp"
        android:layout_marginBottom="8dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/textView" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

2. **Create a layout for recycler View row.**
    a. **Right Click > layout> Android layout > “recycler_row”**

```
<?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="wrap_content">

    <TextView
        android:id="@+id/textView2"
        android:layout_width="80dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="8dp"
        android:layout_marginLeft="8dp"
        android:layout_marginTop="8dp"
        android:text="ID:"
```

```
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/textView3"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="USER ID:"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView2" />

<TextView
    android:id="@+id/textView4"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:text="Title:"
    android:textColor="@color/black"
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView3" />

<TextView
    android:id="@+id/textView5"
    android:layout_width="80dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="8dp"
    android:layout_marginLeft="8dp"
    android:layout_marginTop="16dp"
    android:layout_marginBottom="16dp"
    android:text="Body:"
    android:textColor="@color/black"
```

```
    android:textSize="18sp"
    android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/textView4" />

<TextView
    android:id="@+id/idTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="8dp"
    android:layout_marginEnd="24dp"
    android:layout_marginRight="24dp"
    android:text="TextView"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/textView2"
    app:layout_constraintTop_toTopOf="parent" />

<TextView
    android:id="@+id/userTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="24dp"
    android:layout_marginRight="24dp"
    android:text="TextView"
    android:textSize="18sp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/textView3"
    app:layout_constraintTop_toBottomOf="@+id/idTV" />

<TextView
    android:id="@+id/titleTV"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="24dp"
    android:layout_marginLeft="24dp"
    android:layout_marginTop="16dp"
    android:layout_marginEnd="24dp"
```

```
        android:layout_marginRight="24dp"
        android:text="TextView"
        android:textSize="18sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView4"
        app:layout_constraintTop_toBottomOf="@+id/userTV" />

    <TextView
        android:id="@+id/bodyTV"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="24dp"
        android:layout_marginLeft="24dp"
        android:layout_marginTop="16dp"
        android:layout_marginEnd="24dp"
        android:layout_marginRight="24dp"
        android:layout_marginBottom="16dp"
        android:text="TextView"
        android:textSize="18sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toEndOf="@+id/textView5"
        app:layout_constraintTop_toBottomOf="@+id/titleTV" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

3. Update the **MainActivity.kt.**

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

        recyclerView = findViewById(R.id.myRecycler)
    }



    override fun onResume() {
        super.onResume()

        //1: Create volley object
```

```
        val queue = Volley.newRequestQueue(this)
        val fetchURL = "https://jsonplaceholder.typicode.com/posts"

        //2: Create a String Request to connect with API and fetch data and provide response as
string
        val fetchRequest = StringRequest(Request.Method.GET,fetchURL,
                Response.Listener<String> {response ->

                    //Update Recycler View Adapter to display data as List
                    try{
                        val jsonArray = JSONArray(response)
                        val adapter = MyAdapter(jsonArray)

                        recyclerView!!.adapter = adapter
                        recyclerView!!.layoutManager = LinearLayoutManager(this)

recyclerView!!.addItemDecoration(DividerItemDecoration(this,DividerItemDecoration.VERTICAL
))

                    }catch (e:Exception){
                        e.printStackTrace()
                    }



                },
                Response.ErrorListener {error ->
                    Log.d("Response Error",error.toString())
                })

        //3: Add the request inside volley object
        queue.add(fetchRequest)
    }
}
```

4. Create **Recycler View Adapter**
    a. **Right Click on java: Package> new > kotlin class : MyAdapter**

```
class MyAdapter(jsonArray: JSONArray) : RecyclerView.Adapter<MyAdapter.ViewHolder>() {

    var recivedArray : JSONArray?= null
    init {
```

```
        recivedArray = jsonArray
    }



    //1: Link UI components for Recycler Row
    class ViewHolder(itemView: View) : RecyclerView.ViewHolder(itemView) {
        var userTXTV : TextView
        var idTXTV: TextView
        var titleTXTV:TextView
        var bodyTXTV:TextView


        init {
            userTXTV = itemView.findViewById(R.id.userTV)
            idTXTV = itemView.findViewById(R.id.idTV)
            titleTXTV = itemView.findViewById(R.id.titleTV)
            bodyTXTV = itemView.findViewById(R.id.bodyTV)

        }

    }

    //2:Link Recycler Row layout
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyAdapter.ViewHolder
{
        val recycler_row_view =
LayoutInflater.from(parent.context).inflate(R.layout.recycler_row,parent,false)
        return ViewHolder(recycler_row_view)
    }

    //3: Link Data for Recycler Row UI Component
    override fun onBindViewHolder(holder: MyAdapter.ViewHolder, position: Int) {

        var jsonObject : JSONObject = recivedArray!!.getJSONObject(position)

        holder.idTXTV.text = jsonObject.getString("id")
        holder.userTXTV.text = jsonObject.getString("userId")
        holder.titleTXTV.text= jsonObject.getString("title")
        holder.bodyTXTV.text = jsonObject.getString("body")

    }

    //4: Size of Data
```

```
    override fun getItemCount(): Int {
        return recivedArray!!.length()
    }
}
```

5. **Run Applicatio**

Connect Real Device with Android Studio

1. **Make sure developer option is active on Real Device**
    a. Settings of Phone
    b. About Phone > Build Version/Number
    c. Tap 6-7 times, developer option will be active.
2. Connect with USB with your system.
    a. Check for drivers if required
    b. Else while execution application your device will be listed.

Day -06

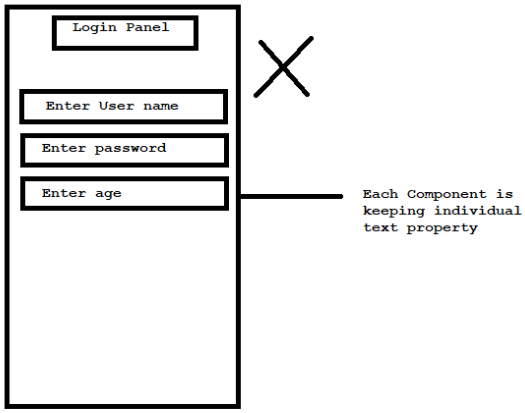
Localization / Internationalization

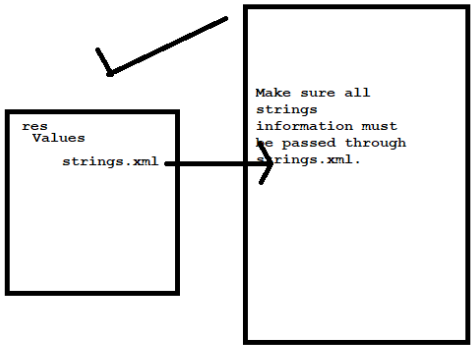
Publish App to Google play

Localization/ Internationalization

It is used to translate the application textual information as per Locale.

Notes:
While implementing Localization make sure all information of string must be called from **strings.xml** file.





Demo: Create a project.

1. **Project Name : Localization Demo**

Update the Text value from strings.xml

1. res>values>strings.xml

<string name="activity_title">Localization Demo</string>

<string name="greetings">Good Morning</string>

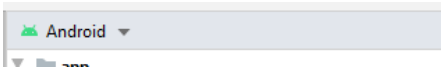
<string name="intro">Hi, I am Prashant Ranjan. We are currently in the session of Google Training.</string>

2. Connect with UI Components.
    a. Select Text View for text property and update it as below.

        @string/intro

<u>Apply Localization</u>

1. **Create a new directory.**
    a. res> right click > new directory
        i. Name:
            1. values-ar
            2. values-hi
            3. Values-fr
2. **Change Project view from Android to Packages.**



    a. **All the folders will be listed there.**

        i. Copy **strings.xml** and paste inside all newly created values folder.
    b. Change the project view to Android.
3. **Update the string for each variable as per their language.**
4. **To Test**
    a. Check phone/emulator settings for language
    b. Change the language and run application.

Deploy/Publish Application:

1. **Visit https://play.google.com/app/publish**
2. Require an account as a developer.
    a. It will charge USD25 / lifetime
3. Need to sign apk.
    a. Signed APK, apk signed with Certificate.
    b. Certificate will keep information about yours details + system information
    c. Same certificate will be required to release the update.

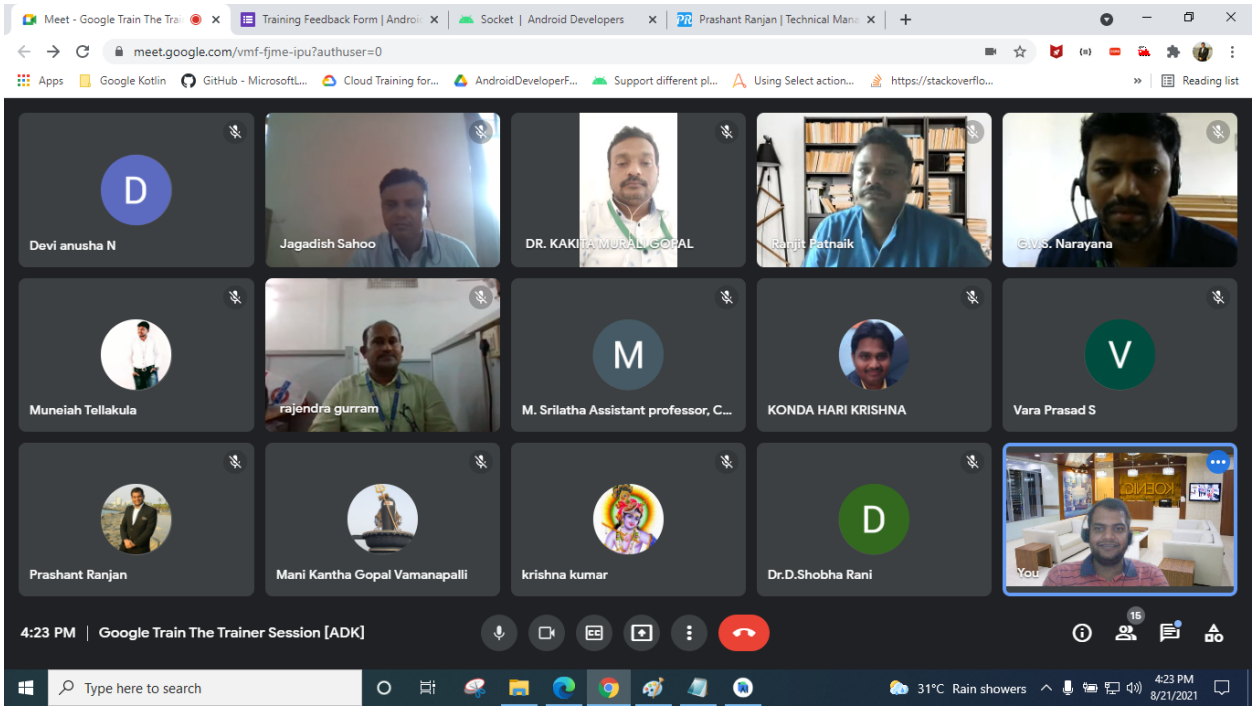
Very Important: Things need to keep safe

1. **Certificate**
2. **Certificate Password**
3. **Alias**
4. **Alias Password**

Process to Deploy : https://youtu.be/rk1AZvwRMw4

To Release update , above information is required.

Thank you


meet.google.com/vmf-fjme-ipu?authuser=0
Devi anusha N
Jagadish Sahoo
DR. KAKITA MURALI GOPAL
Ranjit Patnaik
G.V.S. Narayana
Muneiah Tellakula
rajendra gurram
M. Srilatha Assistant professor, C...
KONDA HARI KRISHNA
Vara Prasad S
Prashant Ranjan
Mani Kantha Gopal Vamanapalli
krishna kumar
Dr.D.Shobha Rani
You
4:23 PM | Google Train The Trainer Session [ADK]
Type here to search
31°C Rain showers
4:23 PM
8/21/2021