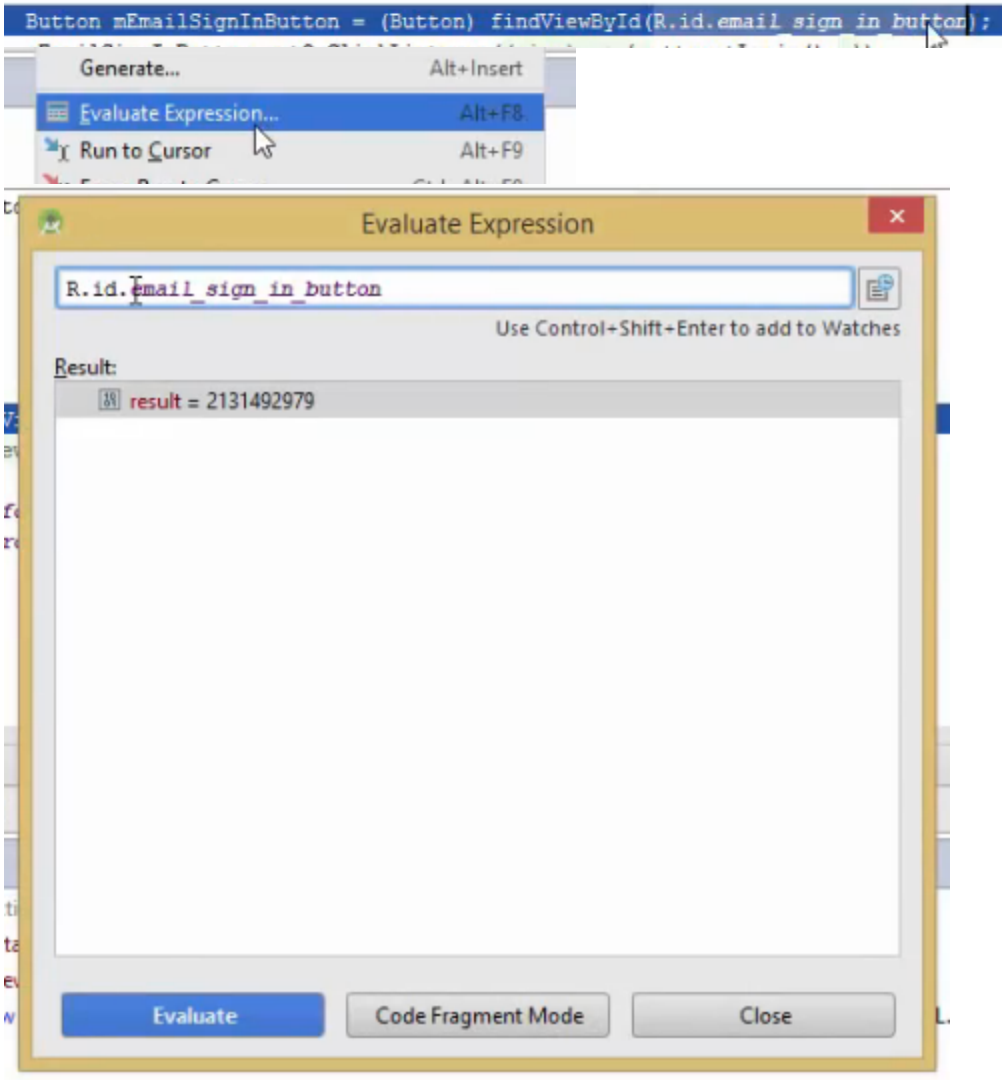
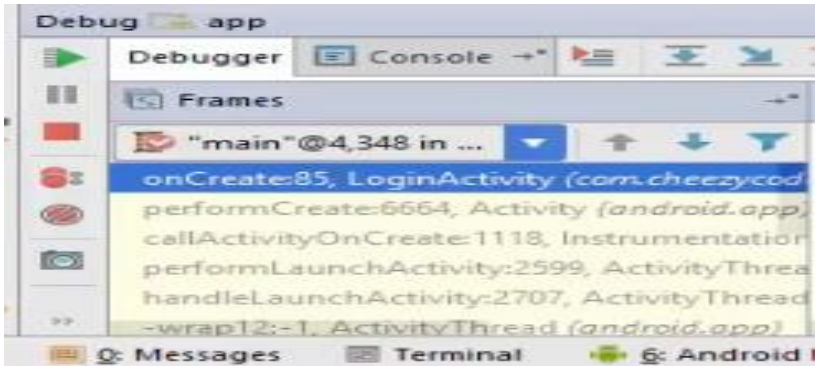
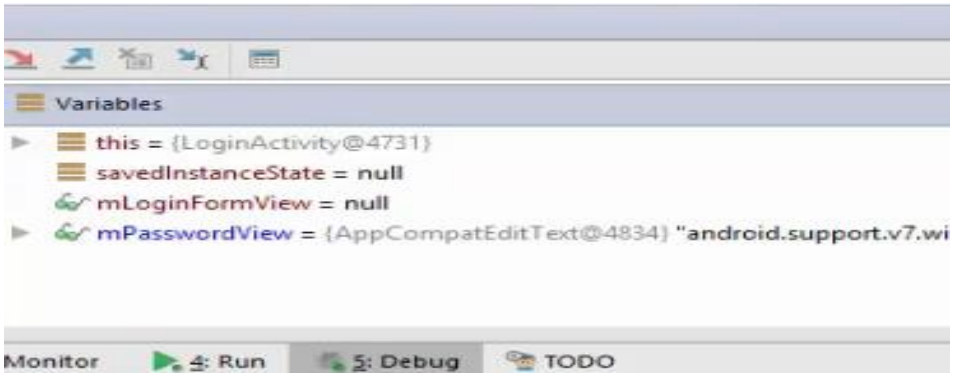
Debugging

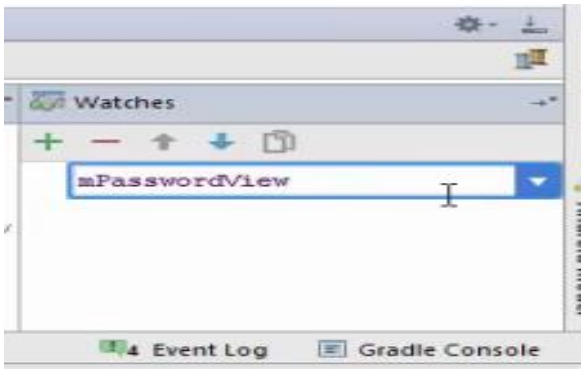

mEmailView = (AutoCompleteTextView) findViewById(R.id.email);
Line 70 in LoginActivity.java
Enabled
Suspend
All
Thread
Condition:
variable1 == "test"
More (Ctrl+Shift+F8)
Done


Button mEmailSignInButton = (Button) findViewById(R.id.email_sign_in_button);
Generate... Alt+Insert
Evaluate Expression... Alt+F8
Run to Cursor Alt+F9
Evaluate Expression
R.id.email_sign_in_button
Use Control+Shift+Enter to add to Watches
Result:
result = 2131492979
Evaluate
Code Fragment Mode
Close


Debug app
Debugger
Console
Frames
"main"@4,348 in ...
onCreate:85, LoginActivity (com.cheezycod
performCreate:6664, Activity (android.app)
callActivityOnCreate:1118, Instrumentation
performLaunchActivity:2599, ActivityThrea
handleLaunchActivity:2707, ActivityThread
-wrap12:-1, ActivityThread (android.app)
Variables
this = {LoginActivity@4731}
savedInstanceState = null
mLoginFormView = null
mPasswordView = {AppCompatEditText@4834} "android.support.v7.wi
Watches
mPasswordView
0: Messages
Terminal
6: Android Monitor
4: Run
5: Debug
TODO
4 Event Log
Gradle Console


Debug app
Debugger
Console
Frames
"main"@4,348 in ...
onCreate:85, LoginActivity (com.cheezycod
performCreate:6664, Activity (android.app)
callActivityOnCreate:1118, Instrumentation
performLaunchActivity:2599, ActivityThrea
handleLaunchActivity:2707, ActivityThread
-wrap12:-1, ActivityThread (android.app)
0: Messages
Terminal
6: Android


Variables
this = {LoginActivity@4731}
savedInstanceState = null
mLoginFormView = null
mPasswordView = {AppCompatEditText@4834} "android.support.v7.wi
Monitor
4: Run
5: Debug
TODO


Watches
mPasswordView
4 Event Log
Gradle Console

Location Services

Show Current Location

AndroidManifest.xml

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

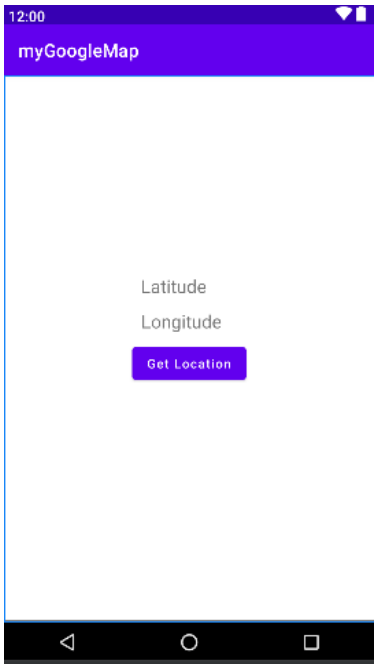
build.gradle(:app)

```
implementation 'com.google.android.gms:play-services-
location:21.0.1'
implementation 'com.google.android.gms:play-services-
maps:18.1.0'
```

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/latitude"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Latitude"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.449"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.381" />
    <TextView
        android:id="@+id/longitude"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Longitude"
        android:textSize="20dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.473"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.449" />
```

```
    <Button
        android:id="@+id/button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginBottom="260dp"
        android:text="Get Location"
        android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.5"
        app:layout_constraintStart_toStartOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



MainActivity.kt

```
package com.ghani.mygooglemap
import android.annotation.SuppressLint
import android.content.Context
import android.content.Intent
import android.content.pm.PackageManager
import android.location.LocationManager
import android.os.Build
import android.os.Bundle
import android.os.Looper
import android.provider.Settings
import android.widget.Button
import android.widget.TextView
import androidx.appcompat.app.AppCompatActivity
import androidx.core.app.ActivityCompat
import com.google.android.gms.location.*
```

```
class MainActivity : AppCompatActivity() {
    lateinit var fusedLocationProviderClient:
    FusedLocationProviderClient

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

        fusedLocationProviderClient =
        LocationServices.getFusedLocationProviderClient(this)

        onGPS()
        findViewById<Button>(R.id.button).setOnClickListener {
            fetchLocation()
        }
    }
    fun onGPS() {
        if(!isLocationEnabled()) {
        startActivity(Intent(Settings.
        ACTION_LOCATION_SOURCE_SETTINGS))
        }else{
            fetchLocation()
        }
    }
    fun isLocationEnabled():Boolean {
        val locationManager = applicationContext.
        getSystemService(Context.LOCATION_SERVICE)
        as LocationManager

        return locationManager. isProviderEnabled
        (LocationManager.GPS_PROVIDER) || locationManager.
        isProviderEnabled(LocationManager.NETWORK_PROVIDER)
    }
    private fun fetchLocation() {
        if(Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            if(ActivityCompat.checkSelfPermission
            (this,android.Manifest.permission.
            ACCESS_FINE_LOCATION) != PackageManager.
            PERMISSION_GRANTED) { ActivityCompat
            .requestPermissions(this,arrayOf(android.Manifest.
            permission.ACCESS_FINE_LOCATION,), 200)
                return
            }else {
                requestLocation()
            }
        }
    }
```

```
    @SuppressLint("MissingPermission")
    private fun requestLocation() {
        val requestLocation = LocationRequest()
        requestLocation.priority =LocationRequest.
        PRIORITY_HIGH_ACCURACY
        requestLocation.interval = 0
        requestLocation.fastestInterval = 0
        requestLocation.numUpdates = 1
        fusedLocationProviderClient.requestLocationUpdates
       (requestLocation,callback, Looper.myLooper())
    }
    private val callback = object:LocationCallback(){
        override fun onLocationResult(result: LocationResult) {

            val lastLocation = result?.lastLocation

            findViewById<TextView>(R.id.longitude).text =
            "Longitude: " + lastLocation?.longitude.toString()

            findViewById<TextView>(R.id.latitude).text =
            "Latitude: " + lastLocation?.longitude.toString()

            super.onLocationResult(result)
        }
    }
 }
```

Tracking Background Location

AndroidManifest.xml

```
<uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
<uses-permission android:name="android.permission.ACCESS_BACKGROUND_LOCATION" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />

<service
    android:name=".LocationService"
    android:enabled="true"
    android:exported="false">
</service>
```

build.gradle(:app)

```
dataBinding{
    enabled = true
}

implementation ("org.greenrobot:eventbus:3.3.1")

implementation 'com.google.android.gms:play-services-
location:21.0.0'
```

activity_main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<layout
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">
  <androidx.constraintlayout.widget.ConstraintLayout
    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:text="Location Updates"
        android:textAllCaps="false"
        android:textColor="@color/black"
        android:textSize="35sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.566"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.098" />
    <TextView
        android:id="@+id/tvLatitude"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Latitude:"
        android:textAllCaps="false"
        android:textColor="@color/black"
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.222"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.279" />
    <TextView
        android:id="@+id/tvLongitude"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Longitude:"
        android:textAllCaps="false"
        android:textColor="@color/black"
```

```
        android:textSize="18sp"
        android:textStyle="bold"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.232"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.354" />
    <Button
        android:id="@+id/btnStartLocationTracking"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="140dp"
        android:layout_marginTop="272dp"
        android:text="Start Location Tracking"
        android:textAllCaps="false"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
    <Button
        android:id="@+id/btnRemoveLocationTracking"
        android:layout_width="160dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="140dp"
        android:layout_marginTop="356dp"
        android:text="Remove Location Tracking"
        android:textAllCaps="false"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

MainActivity.kt

```
package com.ghani.mylocationapp3
import android.content.Intent
import android.content.pm.PackageManager
import android.os.Build
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.activity.result.contract.ActivityResultContracts
import androidx.core.app.ActivityCompat
import com.ghani.mylocationapp3.databinding.ActivityMainBinding
import org.greenrobot.eventbus.EventBus
import org.greenrobot.eventbus.Subscribe

class MainActivity : AppCompatActivity() {
    //_binding in only valid between onCreate and onDestroy
    private var _binding: ActivityMainBinding? = null
    private val binding: ActivityMainBinding
         get() = _binding!!
    private var service: Intent? = null
    private val locationPermissions = registerForActivityResult
(ActivityResultContracts.RequestMultiplePermissions()){
        when {
it.getOrDefault(android.Manifest.permission.ACCESS_COARSE_LOCATI
ON,false) -> {
                if (Build.VERSION.SDK_INT >=
Build.VERSION_CODES.Q ){
                        if (ActivityCompat.checkSelfPermission
(this,android.Manifest.permission.ACCESS_BACKGROUND_LOCATION)!=
PackageManager.PERMISSION_GRANTED){
backgrounLocation.launch(android.Manifest.permission.ACCESS_BACK
GROUND_LOCATION)
                        }
                    }
            }
it.getOrDefault(android.Manifest.permission.ACCESS_FINE_LOCATION
,false) -> {

            }
        }
    }

    private val backgrounLocation = registerForActivityResult
(ActivityResultContracts.RequestPermission()) {
        if (it) {

        }
    }
```

```
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        _binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)

        service = Intent(this, LocationService::class.java)

        binding.apply {
            btnStartLocationTracking.setOnClickListener {
                checkPermissions()
            }
            btnRemoveLocationTracking.setOnClickListener {
                stopService(service)
            }
        }
    }


   fun checkPermissions() {
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O ){
            if (ActivityCompat.checkSelfPermission
(this,android.Manifest.permission.ACCESS_FINE_LOCATION)!=
PackageManager.PERMISSION_GRANTED||ActivityCompat.checkSelfPermi
ssion(this,android.Manifest.permission.ACCESS_COARSE_LOCATION)!=
PackageManager.PERMISSION_GRANTED){
                locationPermissions.launch(
                    arrayOf(
android.Manifest.permission.ACCESS_FINE_LOCATION,
android.Manifest.permission.ACCESS_COARSE_LOCATION
                    )
                )
            }else{
                startService(service)
            }
        }
    }



    override fun onStart() {
        super.onStart()
        if (!EventBus.getDefault().isRegistered(this)){
            EventBus.getDefault().register(this)
        }
    }
```

```
    @Subscribe
    fun receiveLocationEvent(locationEvent: LocationEvent){
        binding.tvLatitude.text = "Latitude ->
${locationEvent.latitude}"
        binding.tvLongitude.text = "Latitude ->
${locationEvent.longitude}"
    }

    override fun onDestroy() {
        super.onDestroy()
        stopService(service)
        if (EventBus.getDefault().isRegistered(this)){
            EventBus.getDefault().unregister(this)
        }
    }

}
```

LocationService.kt

```
package com.ghani.mylocationapp3
import android.app.Notification
import android.app.NotificationChannel
import android.app.NotificationManager
import android.app.Service
import android.content.Intent
import android.location.Location
import android.os.Build
import android.os.IBinder
import androidx.core.app.NotificationCompat
import com.google.android.gms.location.*
import org.greenrobot.eventbus.EventBus

class LocationService : Service() {

    //const initialized at the runtime and val is immutable
    companion object{
        const val CHANNEL_ID = "12345"
        const val NOTIFICATION_ID = 12345
    }

    private var fusedLocationProviderClient:
    FusedLocationProviderClient? = null
    private var locationRequest: LocationRequest? = null
    private var locationCallback:LocationCallback? = null
    private var location: Location? = null
    private var notificationManager: NotificationManager? = null
```

```
override fun onCreate() {
    super.onCreate()
    fusedLocationProviderClient =
    LocationServices.getFusedLocationProviderClient(this)
    locationRequest =
    LocationRequest.Builder
   (Priority.PRIORITY_HIGH_ACCURACY,1000)
   .setIntervalMillis(500).build()
    locationCallback = object : LocationCallback() {
        override fun onLocationAvailability(p0:
        LocationAvailability) {
            super.onLocationAvailability(p0)
        }
        override fun onLocationResult(locationResult:
        LocationResult) {
            super.onLocationResult(locationResult)
            onNewLocation(locationResult)
        }
    }
    notificationManager =
    this.getSystemService(NOTIFICATION_SERVICE) as
    NotificationManager
    if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O ){
        val notificationChannel =
        NotificationChannel(CHANNEL_ID,"locations",
        NotificationManager.IMPORTANCE_HIGH)
        notificationManager?.createNotificationChannel
       (notificationChannel)
    }
}
private fun onNewLocation(locationResult: LocationResult) {
    location = locationResult.lastLocation
    EventBus.getDefault().post(LocationEvent (
        latitude = location?.latitude,
        longitude = location?.longitude
    ))
    startForeground(NOTIFICATION_ID,getNotification())
}
private fun getNotification(): Notification {
    val notification = NotificationCompat.Builder(this,
         CHANNEL_ID)
        .setContentTitle("Location Updates")
        .setContentText("Latitude --> ${location?.latitude}
        \nLongitude --> ${location?.longitude}")
        .setSmallIcon(R.mipmap.ic_launcher)
        .setPriority(NotificationCompat.PRIORITY_HIGH)
        .setOngoing(true)
```

```
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.O) {
            notification.setChannelId(CHANNEL_ID)
        }
        return notification.build()
    }
    override fun onStartCommand(intent: Intent?, flags: Int,
    startId: Int): Int {
        super.onStartCommand(intent, flags, startId)
        createLocationRequest()
        return START_STICKY
    }
    @Suppress("MissingPermission")
    private fun createLocationRequest() {
        try {
            fusedLocationProviderClient?. requestLocationUpdates
           (locationRequest!!,locationCallback!!,null)
        }
        catch (e:Exception){
            e.printStackTrace()
        }
    }
    override fun onBind(intent: Intent): IBinder? = null
    override fun onDestroy() {
        super.onDestroy()
        removeLocationUpdates()
    }
    private fun removeLocationUpdates() {
    fusedLocationProviderClient?.
    removeLocationUpdates(locationCallback!!)
        stopForeground(true)
        stopSelf()
    }
}
```

LocationEvent.kt

```
package com.ghani.mylocationapp3

data class LocationEvent(
    val latitude:Double?,
    val longitude:Double?
 )
```

```
package com.ghani.myjetpackcomposeapp

import android.os.Bundle
import android.util.Log
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.*
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.LazyRow
import androidx.compose.foundation.shape.CircleShape
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.foundation.text.KeyboardOptions
import androidx.compose.foundation.text.selection.DisableSelection
import androidx.compose.foundation.text.selection.SelectionContainer
import androidx.compose.material.*
import androidx.compose.material.icons.Icons
import androidx.compose.material.icons.filled.Add
import androidx.compose.material.icons.filled.Menu
import androidx.compose.material.icons.filled.Person
import androidx.compose.material.icons.filled.Search
import androidx.compose.runtime.Composable
import androidx.compose.runtime.setValue
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.remember
import androidx.compose.ui.Alignment
import androidx.compose.ui.Modifier
import androidx.compose.ui.draw.clip
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.layout.ContentScale
import androidx.compose.ui.res.painterResource
import androidx.compose.ui.text.TextStyle
import androidx.compose.ui.text.font.FontStyle
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.input.KeyboardType
import androidx.compose.ui.text.input.PasswordVisualTransformation
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import com.ghani.myjetpackcomposeapp.ui.theme.MyFontFamily
import coil.compose.rememberImagePainter
```

```
class MainActivity : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            //Text(text="Hello Jetpack Compose",fontSize = 30.sp)
            //Disp1()
            //Disp2("Rony")
            //Disp3("Rony")
            //Disp4()
            //SimpleText()
            //LongText()
            //SelectableText()
            //PartiallySelectableText()
            //RowExample()
            //ColumnExample()
            //LazyRowExample()
            //LazyColumnExample()
            //BoxExample()
            //MaterialUIApp()
            //ImageAssetExample()
            //ImageNetworkExample()
            //IconExample()
            //CardExample()
            //StatefulExample()
            //HelloSreen()
            //ButtonsExample()
            //TextFieldExample()
            Registration()
        }
    }
}
@Composable
fun Disp1(){
    Text(text="Hello! Jetpack Compose",fontSize = 30.sp)
}
@Composable
fun Disp2(name:String){
    Text(text="Hello! $name",fontSize = 30.sp)
}
@Composable
fun Disp3(name:String){
    Column{
        Text(text="Hello! $name Kapor",fontSize = 30.sp)
        Text(text="Hello! $name Singh",fontSize = 30.sp)
        Text(text="Hello! $name Khan",fontSize = 30.sp)
    }
}
```

```
@Composable
fun Disp4(){
    val names = listOf<String>("Rony","Jony","Tony")
    Column{
        for (name in names){
            Text(text="Hello! $name Kapor",fontSize = 30.sp)
        }
    }
}
@Composable
fun SimpleText(){
    Text(
        text="Hello! kotlin",
        fontSize = 30.sp,
        color = Color.Magenta,
        fontStyle = FontStyle.Italic,
        fontWeight = FontWeight.ExtraBold,
        textAlign = TextAlign.Center,
        fontFamily = MyFontFamily,
        modifier = Modifier
            .width(410.dp)
            .background(color = Color.Yellow),
    )
}
@Composable
fun LongText(){
    Text(text="Jetpack  ".repeat(10),fontSize = 30.sp,maxLines = 2)
}
@Composable
fun SelectableText(){
    SelectionContainer{
        Text(text="Start Tutorial Jetpack",fontSize = 30.sp)
    }
}
@Composable
fun PartiallySelectableText(){
    Column() {
        SelectionContainer{
            Column{
                Text(text="This is selectable text",fontSize = 30.sp)
                DisableSelection{
                  Text(text="This is non selectable text",fontSize = 30.sp)
                }
            }
        }
        Text(text="This is non selectable text",fontSize = 30.sp)
    }
}
```

```
@Composable
fun RowExample(){
    Row(
        modifier = Modifier
            .fillMaxHeight()
            .fillMaxWidth()
            .background(color = Color.Gray)
            .horizontalScroll(rememberScrollState()),
        horizontalArrangement = Arrangement.Center,
        verticalAlignment = Alignment.CenterVertically
    ) {
        Text(text="First  ",fontSize = 30.sp)
        for (i in 1..50){
            Text(text="Item$i  ",fontSize = 30.sp)
        }
        Text(text="Last",fontSize = 30.sp)
    }
}

@Composable
fun ColumnExample(){
    Column(
        modifier = Modifier
          .fillMaxSize()//.fillMaxHeight()+.fillMaxWidth()= .fillMaxSize()
          .background(color = Color.Gray)
          .verticalScroll(rememberScrollState()),
        verticalArrangement = Arrangement.Center,
        horizontalAlignment = Alignment.CenterHorizontally
    ) {
        Text(text="First  ",fontSize = 30.sp)
        for (i in 1..50){
            Text(text="Item$i  ",fontSize = 30.sp)
        }
        Text(text="Last",fontSize = 30.sp)
    }
}

@Composable
fun LazyRowExample(){
    LazyRow(
        modifier = Modifier
            .fillMaxSize()
            .background(color = Color.Gray),
        verticalAlignment =  Alignment.CenterVertically,
        horizontalArrangement =Arrangement.spacedBy(50.dp),
        contentPadding=PaddingValues(horizontal=99.dp,vertical= 50.dp)
    ) {
```

```
        item{Text(text="First  ",fontSize=30.sp)}//Adding Single item
        items(51){i->Text(text="Item$i  ",fontSize=30.sp)}//Adding Multiple items
        item{Text(text="Last",fontSize=30.sp)}//Adding Single item
    }
}
@Composable
fun LazyColumnExample(){
    LazyColumn(
        modifier = Modifier
            .fillMaxWidth()
            .background(color = Color.Magenta),
        horizontalAlignment =  Alignment.CenterHorizontally,
        verticalArrangement =Arrangement.spacedBy(50.dp),
        contentPadding=PaddingValues(horizontal=50.dp,vertical= 100.dp)
    ) {
        item{Text(text="First  ",fontSize=30.sp)}
        items(51){ i -> Text(text="Item$i  ",fontSize = 30.sp) }
        item { Text(text="Last",fontSize = 30.sp) }
    }
}
@Composable
fun BoxExample(){
    Box(
        modifier = Modifier
            .fillMaxHeight(0.5f)
            .fillMaxWidth(0.5f)
            .background(color = Color.Magenta)
    ) {
        Text(text="This is outer box",
            fontSize = 20.sp,
            color = Color.Green)
        Box(
            modifier = Modifier
                .fillMaxHeight(0.5f)
                .fillMaxWidth(0.5f)
                .background(color = Color.Blue)
        ) {
            Text(text="This is inner box",
                fontSize = 20.sp,
                color = Color.Red,
                modifier = Modifier.align(Alignment.Center))
        }
        Text(text="This is outer box",
            fontSize = 20.sp,
            color = Color.Green,
            modifier = Modifier.align(Alignment.Center))
    }
}
```

```
@Composable
fun MaterialUIApp(){
    Scaffold(topBar ={TopAppBar(title ={Text(text ="Material App")},
                                actions ={IconButton(onClick =
                                {Log.d("Button Clicked","Search Button
                                Clicked")})
                                          {
                                           Icon(Icons.
                                           Filled.Search,
                                           contentDescription = null)
                                                                  }})}
    ) {
         LazyColumnExample()
      }
}
@Composable
fun ImageAssetExample(){
    Image(
        painter = painterResource(id = R.drawable.river),
        contentDescription = "Profile Image",
        contentScale = ContentScale.Crop,
        modifier = Modifier
            .border(5.5.dp, Color.Red, CircleShape)
            .size(300.dp)
            .clip(shape = CircleShape)
    )
}
@Composable
fun ImageNetworkExample(){
    Image(
        painter = rememberImagePainter(data
        ="https://cdn.pixabay.com/photo/2018/01/29/17/01/woman-
        3116587_960_720.jpg"),
        contentDescription = "Profile Image",
        contentScale = ContentScale.Crop,
        modifier = Modifier
            .border(5.5.dp, Color.Red, CircleShape)
            .size(300.dp)
            .clip(shape = CircleShape)
    )
}
@Composable
fun IconExample(){
    Icon(
        Icons.Filled.Menu,
        contentDescription = "Menu",
        modifier = Modifier.size(50.dp),
        tint = Color.Green
    )
}
```

```
@Composable
fun CardExample() {
    Card(
        shape = RoundedCornerShape(10.dp),
        backgroundColor = Color.Magenta,
        border = BorderStroke(5.dp, Color.Red),
        contentColor = Color.Green
    ) {
        Column(modifier = Modifier.padding(40.dp)) {
            Text(
                text = "This is a card",
                fontSize = 30.sp,
                textAlign = TextAlign.Center)

            Spacer(modifier = Modifier.height(50.dp))

            Text(
                text = "Looking nice!!",
                fontSize = 30.sp,
                textAlign = TextAlign.Center)
        }
    }
}
@Composable
fun StatefulExample(){
    var name:String by remember {mutableStateOf("")}
    Column{
        OutlinedTextField(value = name,onValueChange={name = it}
)
        Text(text = name, fontSize = 30.sp)
    }
}

//State Hoisting
@Composable
fun HelloSreen(){
    var name:String by remember {mutableStateOf("")}
    HelloContent(name = name, onNameChange = {name = it})
}
@Composable
fun HelloContent(name:String, onNameChange:(String) -> Unit){
    Column{
       OutlinedTextField(value=name,onValueChang =onNameChange )
       Text(text = name, fontSize = 30.sp)
    }
}
```

```
@Composable
fun ButtonsExample(){
    Column() {
        Button(
            onClick = {Log.d ("Button","Button is clicked")},
            modifier = Modifier.padding(30.dp),
            contentPadding = PaddingValues(start = 40.dp,top =
            20.dp,end = 40.dp,bottom = 20.dp),
            colors = ButtonDefaults.buttonColors(contentColor =
            Color.Green,backgroundColor = Color.Yellow),
            shape = CircleShape,
            border = BorderStroke(1.dp,Color.Blue)
        ){
            Text(text = "Button", fontSize = 24.sp)
        }

        TextButton(
            onClick = {Log.d ("Text Button","Text Button is
            clicked")},
            modifier = Modifier.padding(30.dp),
            colors = ButtonDefaults.outlinedButtonColors
           (contentColor = Color.Green,backgroundColor =
            Color.Magenta)
        ){
            Text(text = "Text Button", fontSize = 24.sp)
        }

        OutlinedButton(
            onClick={Log.d("Outline Button","Outline
            Button is clicked")},
            modifier = Modifier.padding(30.dp),
            contentPadding = PaddingValues(start = 40.dp,top =
            20.dp,end = 40.dp,bottom = 20.dp),
            colors = ButtonDefaults.outlinedButtonColors
            (contentColor = Color.Green, backgroundColor =
            Color.Magenta),
            shape = CircleShape,
            border = BorderStroke(2.dp,Color.Magenta)
        ){
            Text(text = " Outlined Button", fontSize = 24.sp)
        }

        IconButton(
            onClick = {Log.d ("Icon Button","Icon Button is
            clicked")},
            modifier = Modifier
                .padding(40.dp)
```

```
                .then(Modifier.size(50.dp))
                .border(1.dp, Color.Red, shape = CircleShape)
        ){
            Icon(Icons.Default.Add, contentDescription = "Icon
            Button",tint = Color.Red)
        }

        FloatingActionButton(
            onClick = {Log.d ("Floating Action Button","Floating
            Action Button is clicked")},
            modifier = Modifier.padding(40.dp),
            backgroundColor = Color.Green
        ){
            Icon(Icons.Default.Add, contentDescription =
           "Floating Action Button")
        }

    }
}

@Composable
fun TextFieldExample(){
    LazyColumn(
        modifier = Modifier.fillMaxSize(),
        horizontalAlignment =  Alignment.CenterHorizontally,
        verticalArrangement =Arrangement.spacedBy(40.dp),
        contentPadding = PaddingValues(horizontal = 50.dp,
        vertical = 70.dp)
    ) {
        item {
            TextField(
                value = "",
                onValueChange = {},
                label = { Text(text = "Name")},
                placeholder = { Text(text = "Write your name")},
                textStyle = TextStyle(color = Color.Blue),
                leadingIcon = { Icon(imageVector =
                Icons.Filled.Person, contentDescription ="")},
                shape = RoundedCornerShape(10.dp),
                colors =
                TextFieldDefaults.textFieldColors
               (focusedIndicatorColor = Color.Green,
                unfocusedIndicatorColor = Color.Magenta),
              //readOnly = true,
              //singleLine = true
            )
        }
```

```
        item {
            OutlinedTextField(
                value = "",
                onValueChange = {},
                label = { Text(text = "Email")},
                placeholder ={ Text(text = "Write your email")},
                textStyle = TextStyle(color = Color.Blue),
                leadingIcon = { Icon(imageVector =
                Icons.Filled.Person, contentDescription ="")},
                shape = RoundedCornerShape(10.dp),
                colors =
                TextFieldDefaults.textFieldColors
               (focusedIndicatorColor = Color.Green,
                unfocusedIndicatorColor = Color.Magenta),
              //readOnly = true,
              //singleLine = true
            )
        }
        item {
            OutlinedTextField(
                value = "Password",
                onValueChange = {},
                label = { Text(text = "Password")},
                leadingIcon = { Icon(imageVector =
                Icons.Filled.Person, contentDescription ="")},
                shape = RoundedCornerShape(10.dp),
                colors =
                TextFieldDefaults.textFieldColors
               (focusedIndicatorColor = Color.Green,
                unfocusedIndicatorColor = Color.Magenta),
                visualTransformation =
                PasswordVisualTransformation(),
                keyboardOptions = KeyboardOptions(keyboardType =
                KeyboardType.Password)
            )
        }
    }
}

@Composable
fun Registration(){
    var name:String by remember {mutableStateOf("")}
    var email:String by remember {mutableStateOf("")}
    var password:String by remember {mutableStateOf("")}
    LazyColumn(
        Modifier.fillMaxSize(),
        horizontalAlignment =  Alignment.CenterHorizontally,
```

```
        verticalArrangement =Arrangement.spacedBy(10.dp),
        contentPadding = PaddingValues(horizontal = 10.dp,
        vertical = 20.dp)
    ){
        item {
            OutlinedTextField(
                value = name,
                onValueChange = {name = it},
                label = { Text(text = "Name") }
            )
        }
        item {
            OutlinedTextField(
                value = email,
                onValueChange = {email = it},
                label = { Text(text = "Email") },
                keyboardOptions = KeyboardOptions(keyboardType =
                KeyboardType.Email)
            )
        }
        item {
            OutlinedTextField(
                value = password,
                onValueChange = {password = it},
                label = { Text(text = "Password") },
                visualTransformation =
                PasswordVisualTransformation(),
                keyboardOptions = KeyboardOptions(keyboardType =
                KeyboardType.Password)
            )
        }
        item {
            Button(
                onClick = {Log.d ("SubmitButton","Name:$name
                Email:$email   Password:$password")},
                contentPadding = PaddingValues(start = 20.dp,top
                = 10.dp,end = 20.dp,bottom = 10.dp),
                colors=ButtonDefaults.buttonColors(contentColor
                = Color.White,backgroundColor = Color.DarkGray),
                shape = CircleShape,
            ){
                Text(text = "Submit")
            }
        }
    }
}
```

Android Architecture Component

Pattern:

- MVC
- MVP
- MVVM

Need:

- Separation Of Concerns.
- Data Driven Applications.

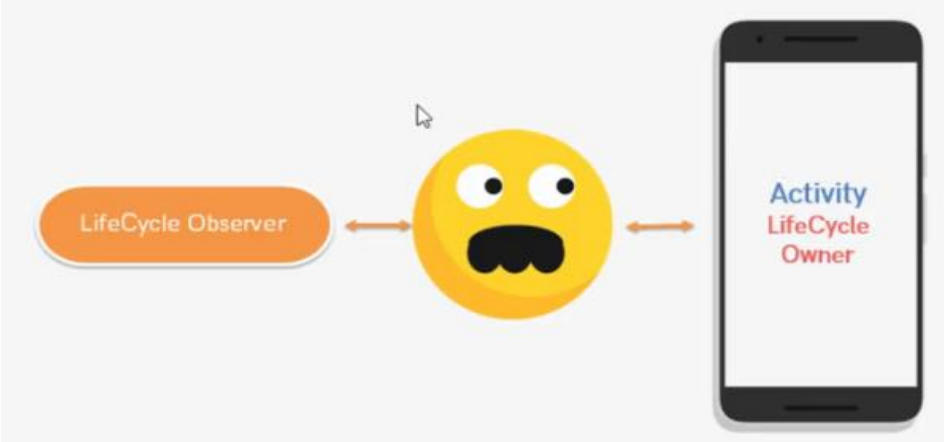
Features:

- Data Binding
- ViewModel
- LiveData
- Room Database
- Navigation Component
- Work Manager
- Paging

Life Cycle Aware Components:

- Most of the code is written inside Activity LifeCycle Methods - onCreate, onResume, onPause etc. Due to this, Activity has multiple responsibilities.
- But there are scenarios where we want to take actions based on the activity lifecyle.
- For e.g.
  - Access the User's Location.
  - Playing Video.
  - Downloading Images.

- LifeCycle Owner
- LifeCycle Observer



Observer.kt:

```
package com.ghani.lifecycleaware

import android.util.Log
import androidx.lifecycle.Lifecycle
import androidx.lifecycle.LifecycleObserver
import androidx.lifecycle.OnLifecycleEvent

class Observer : LifecycleObserver {

   @OnLifecycleEvent(Lifecycle.Event.ON_CREATE)
   fun onCreate(){
      Log.d("Main","Observer - On Create")
   }
   @OnLifecycleEvent(Lifecycle.Event.ON_RESUME)
   fun onResume(){
      Log.d("Main","Observer - On Resume")
   }
   @OnLifecycleEvent(Lifecycle.Event.ON_PAUSE)
   fun onPause(){
      Log.d("Main","Observer - On Pause")
   }
   @OnLifecycleEvent(Lifecycle.Event.ON_STOP)
   fun onStop(){
      Log.d("Main","Observer - On Stop")
   }
   @OnLifecycleEvent(Lifecycle.Event.ON_DESTROY)
   fun onDestroy(){
      Log.d("Main","Observer - On Destroy")
   }
}
```

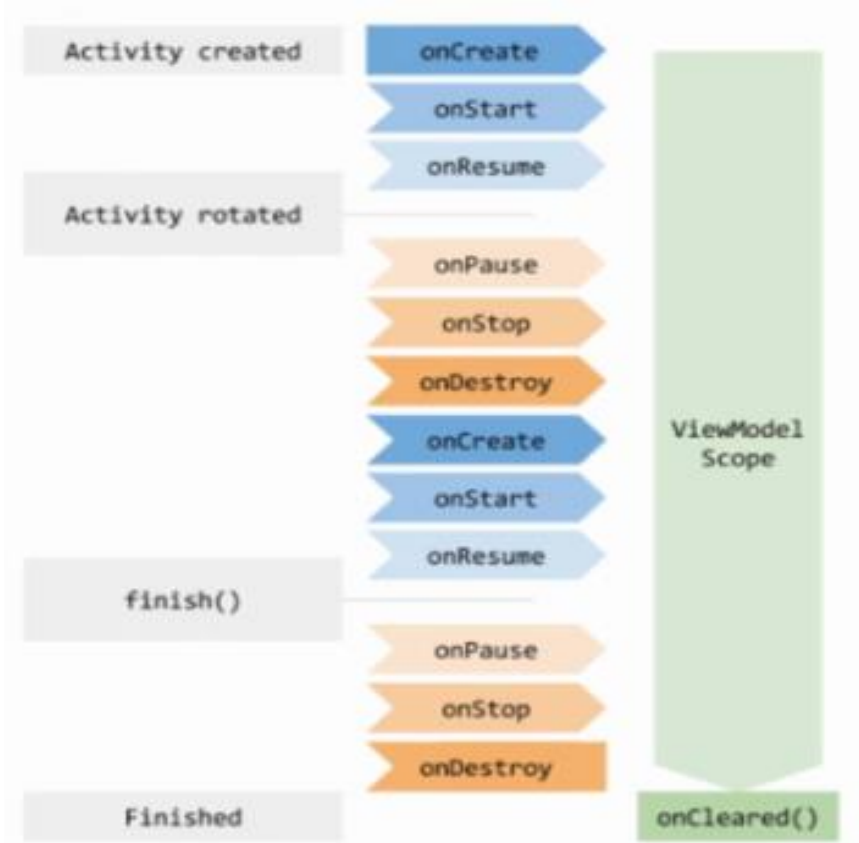
MainActivity.kt:

```
package com.ghani.lifecycleaware

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log

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

        lifecycle.addObserver(Observer())
        Log.d("Main","Activity - On Create")
    }
    override fun onResume(){
        super.onResume()
        Log.d("Main","Activity - On Resume")
    }
    override fun onPause(){
        super.onPause()
        Log.d("Main","Activity - On Pause")
    }
    override fun onStop(){
        super.onStop()
        Log.d("Main","Activity - On Stop")
    }
    override fun onDestroy(){
        super.onDestroy()
        Log.d("Main","Activity - On Destroy")
    }
}
```

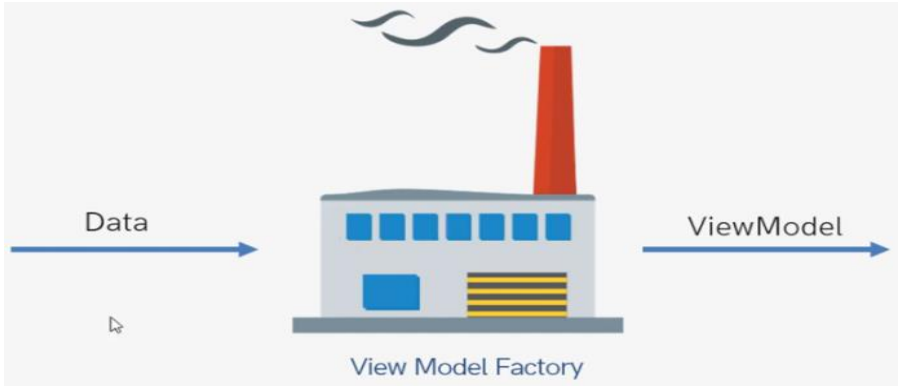
Output in Logcat:

```
Activity - On Create
Observer - On Create
Activity - On Resume
Observer - On Resume
Observer - On Pause
Activity - On Pause
Observer - On Stop
Activity - On Stop
```

View Model:



View Model Factory:



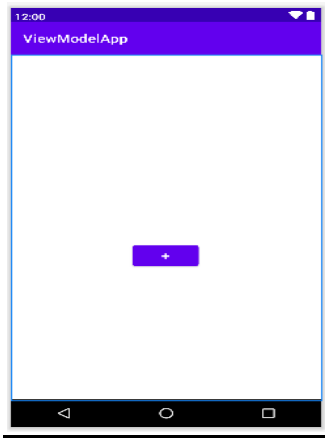
---

build.gradle:

```
dependencies {
    //Adding by me
   def lifecycle_version = "2.5.1"
   implementation("androidx.lifecycle:lifecycle-viewmodel-
   ktx:$lifecycle_version")
}
```

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/tvCounter"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:textSize="36sp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
   <Button
      android:onClick="increment"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="+"
      android:textSize="20sp"
      android:textAllCaps="false"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintHorizontal_bias="0.498"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent"
      app:layout_constraintVertical_bias="0.591" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



MainActivity.kt:

```
package com.ghani.viewmodelapp

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.lifecycle.ViewModelProvider

class MainActivity : AppCompatActivity() {

   lateinit var tvCounter: TextView
   lateinit var mainViewModel:MainViewModel

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

     mainViewModel = ViewModelProvider(this,MainViewModelFactory(10)).get(MainViewModel::class.java)
      tvCounter = findViewById(R.id.tvCounter)
      setText()
   }

   private fun setText() {
      tvCounter.text = mainViewModel.count.toString()
   }

   fun increment (v:View){      // Note: View -> show data, ViewModel -> hold data
      mainViewModel.increment ()
      setText()
   }
}
```

MainViewModelFactory.kt:

```
package com.ghani.viewmodelapp

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class MainViewModelFactory(val counter:Int):ViewModelProvider.Factory {
   override fun <T : ViewModel> create(modelClass: Class<T>): T {
      return MainViewModel(counter) as T
   }
}
```

MainViewModel.kt:

```
package com.ghani.viewmodelapp

import androidx.lifecycle.ViewModel

class MainViewModel(val initialValue:Int): ViewModel() {
   var count:Int = initialValue

   fun increment (){
      count++
   }
}
```

Quote App:

build.gradle:

```
dependencies {
    //Adding by me
   def lifecycle_version = "2.5.1"
   implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
   implementation "com.google.code.gson:gson:2.8.6"
}
```

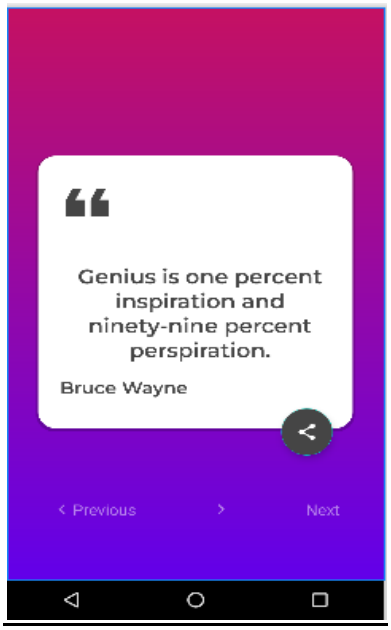
activity_main.xml:

```
<?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"
   android:background="@drawable/bg_gradient"
   android:padding="32dp"
   android:paddingLeft="0dp"
   android:paddingTop="32dp"
   tools:context=".MainActivity">
   <TextView
      android:id="@+id/textView"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="32dp"
      android:fontFamily="@font/montserrat_semibold"
      android:text="Quotify"
      android:textAlignment="center"
```

```
    android:textColor="#FFF"
    android:textSize="28sp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent" />
  <LinearLayout
    android:id="@+id/linearLayout"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:background="@drawable/bg_card"
    android:orientation="vertical"
    android:paddingStart="16dp"
    android:paddingTop="16dp"
    android:paddingEnd="16dp"
    android:paddingBottom="40dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toTopOf="parent">
    <ImageView
      android:layout_width="80dp"
      android:layout_height="80dp"
      android:scaleX="-1"
      android:scaleY="-1"
      app:tint="#454545"
      android:src="@drawable/ic_quote"/>
    <TextView
      android:id="@+id/quoteText"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="32dp"
      android:fontFamily="@font/montserrat_semibold"
      android:paddingStart="10dp"
      android:text="Genius is one percent inspiration and ninety-nine percent perspiration."
      android:textAlignment="center"
      android:textColor="#454545"
      android:textSize="24sp" />
    <TextView
      android:id="@+id/quoteAuthor"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_marginTop="16dp"
      android:fontFamily="@font/montserrat_semibold"
```

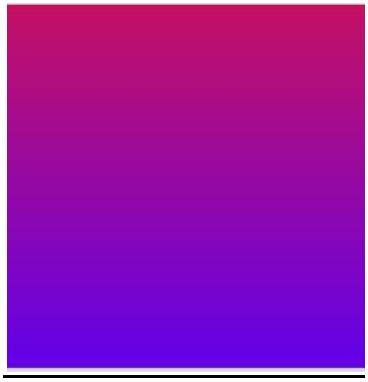
```
        android:paddingStart="10dp"
        android:text="Bruce Wayne"
        android:textColor="#454545"
        android:textSize="20sp" />
</LinearLayout>
<LinearLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:paddingStart="16dp"
    android:paddingTop="16dp"
    android:paddingEnd="16dp"
    android:paddingBottom="40dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent">
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:drawableStart="@drawable/ic_left_arrow"
        android:drawableTint="#80FFFFFF"
        android:text="Previous"
        android:onClick="onPrevious"
        android:textColor="#80FFFFFF"
        android:textSize="18sp" >
    </TextView>
    <TextView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:drawableStart="@drawable/ic_arrow"
        android:drawableTint="#80FFFFFF"
        android:text="Next"
        android:onClick="onNext"
        android:textAlignment="textEnd"
        android:textColor="#80FFFFFF"
        android:textSize="18sp" >
    </TextView>
</LinearLayout>
<com.google.android.material.floatingactionbutton.FloatingActionButton
    android:id="@+id/floatingActionButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
```

```
        android:clickable="true"
        android:layout_marginRight="24dp"
        android:backgroundTint="#454545"
        android:onClick="oneShare"
        android:foregroundTint="#FFF"
        app:layout_constraintBottom_toBottomOf="@+id/linearLayout"
        app:layout_constraintEnd_toEndOf="@+id/linearLayout"
        app:layout_constraintTop_toBottomOf="@+id/linearLayout"
        app:srcCompat="@drawable/ic_share" />
</androidx.constraintlayout.widget.ConstraintLayout>
```



bg_gradient.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <gradient android:startColor="#C51162" android:endColor="#6200EA"
       android:type="linear"
       android:angle="270"/>
</shape>
```



bg_card.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layer-list xmlns:android="http://schemas.android.com/apk/res/android">
   <item>
      <shape>
         <solid android:color="#33000000"/>
         <corners android:radius="20dp"/>
      </shape>
   </item>
   <item android:right="2dp" android:left="2dp" android:bottom="4dp">
      <shape>
         <solid android:color="#FFF"/>
         <corners android:radius="20dp"/>
      </shape>
   </item>
</layer-list>
```

MainActivity.kt:

```
package com.ghani.quotifyappusingviewmodel

import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.view.View
import android.widget.TextView
import androidx.lifecycle.ViewModelProvider

class MainActivity : AppCompatActivity() {
   lateinit var mainViewModel: MainViewModel

   private val quoteText:TextView
   get() = findViewById(R.id.quoteText)

   private val quoteAuthor:TextView
   get() = findViewById(R.id.quoteAuthor)
```

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

        mainViewModel =
        ViewModelProvider(this,MainViewModelFactory(application)).get(MainViewModel::class.java)
        setQuoue(mainViewModel.getQuote())
    }

    fun setQuoue(quote:Quote){
        quoteText.text = quote.text
        quoteAuthor.text = quote.author
    }

    fun onNext(view: View) {
        setQuoue(mainViewModel.nextQuote())
    }

    fun onPrevious(view: View) {
        setQuoue(mainViewModel.previousQuote())
    }

    fun oneShare(view: View) {
        val intent = Intent(Intent.ACTION_SEND)
        intent.setType("text/plain")
        intent.putExtra(Intent.EXTRA_TEXT,mainViewModel.getQuote().text)
        startActivity(intent)
    }
}
```

MainViewModelFactory.kt:

```
package com.ghani.quotifyappusingviewmodel

import android.content.Context
import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider

class MainViewModelFactory(val context: Context):ViewModelProvider.Factory {

    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return MainViewModel(context) as T
    }
}
```

MainViewModel.kt:

```
package com.ghani.quotifyappusingviewmodel

import android.content.Context
import androidx.lifecycle.ViewModel
import com.google.gson.Gson

class MainViewModel(val context: Context):ViewModel() {
    private var quoteList:Array<Quote> = emptyArray()
    private var index = 0

    init {
        quoteList = loadQuoteFromAssets()
    }

    private fun loadQuoteFromAssets(): Array<Quote> {
        val inputStream = context.assets.open("quotes.json")
        val size:Int = inputStream.available()
        val buffer = ByteArray(size)
        inputStream.read(buffer)
        inputStream.close()
        val json = String(buffer,Charsets.UTF_16)
        val gson = Gson()
        return gson.fromJson(json,Array<Quote>::class.java)
    }

    fun getQuote() = quoteList[index]

    fun nextQuote() = quoteList[(++index + quoteList.size) % quoteList.size]

    fun previousQuote() = quoteList[(--index + quoteList.size) % quoteList.size]
}
```

Quote.kt:

```
package com.ghani.quotifyappusingviewmodel

data class Quote (val text:String,val author:String)
```

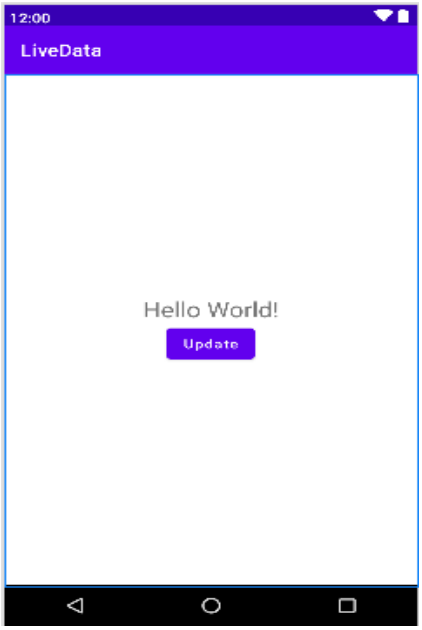
Live Data

build.gradle:

```
dependencies {
   //Adding by me
   def lifecycle_version = "2.5.1"
   implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version")
   implementation("androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version")
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   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:gravity="center"
   android:orientation="vertical"
   tools:context=".MainActivity">
   <TextView
      android:id="@+id/factsTextView"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Hello World!"
      android:textSize="25sp"/>
   <Button
      android:id="@+id/btnUpdate"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
      android:text="Update"
      android:textAllCaps="false"
      android:textSize="15sp"/>
</LinearLayout>
```



MainActivity.kt:

```
package com.ghani.livedata

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Button
import android.widget.TextView
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider

class MainActivity : AppCompatActivity() {

   lateinit var mainViewModel: MainViewModel

   private val factsTextView: TextView
      get()= findViewById(R.id.factsTextView)

   private val btnUpdate:Button
   get()= findViewById(R.id.btnUpdate)

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

      mainViewModel = ViewModelProvider(this).get(MainViewModel::class.java)
```

```
        mainViewModel.factsLiveData.observe(this, Observer {
            //This code will be executed automatically when factsLiveData changed.
            factsTextView.text = it
        })

        btnUpdate.setOnClickListener {
            mainViewModel.updateLiveData()
        }
    }
}
```

MainViewModel.kt:

```
package com.ghani.livedata

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class MainViewModel: ViewModel() {
    var factsLiveDataObject = MutableLiveData<String>("This is a fact")
    val factsLiveData
    get()= factsLiveDataObject

    fun updateLiveData(){
        factsLiveDataObject.value = "This is another fact"
    }
}
```

Data Binding

build.gradle:

```
android {
    //Adding by me
    buildFeatures{
        dataBinding true
    }
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layout 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">
```

```
    <data>
        <variable
            name="quote"
            type="com.ghani.databinding.Quote" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        tools:context=".MainActivity">
        <TextView
            android:id="@+id/quoteText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{quote.text}"
            android:textSize="28sp"
            android:textStyle="bold" />
        <TextView
            android:id="@+id/quoteAuthor"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{quote.author}"
            android:layout_marginTop="16dp"
            android:textSize="20sp"
            android:textStyle="bold" />
    </LinearLayout>
</layout>
```

MainActivity.kt:

```
package com.ghani.databinding

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import androidx.databinding.DataBindingUtil
import com.ghani.databinding.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
   lateinit var binding:ActivityMainBinding

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      binding = DataBindingUtil.setContentView(this,R.layout.activity_main)

      //binding.quoteText.text = "Do or do not. There is no try."
      //binding.quoteAuthor.text = "Yoda"

      val quoteObj = Quote("Do or do not. There is no try.","Yoda")
      binding.quote = quoteObj
   }
}
```

Quote.kt:

```
package com.ghani.databinding

data class Quote(val text:String,val author:String)
```

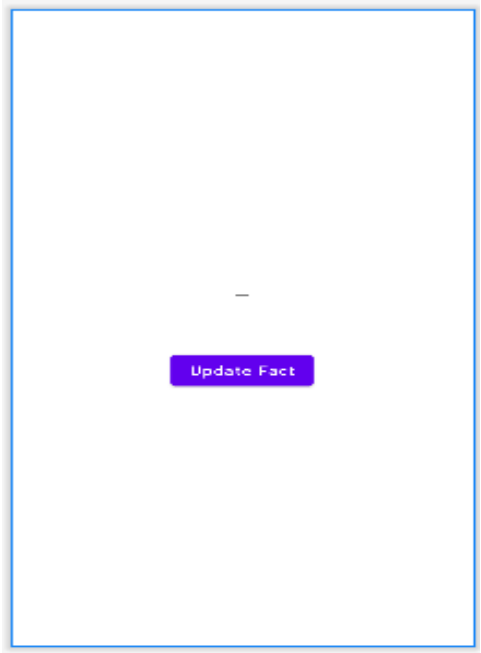
Data Binding With Live Data

build.gradle:

```
android {
   //Adding by me
   buildFeatures{
      dataBinding true
   }
}
dependencies {
   //Adding by me
   def lifecycle_version = "2.5.1"
   implementation("androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version")
   implementation("androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version")
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layout 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">
    <data>
        <variable
            name="mainViewModel"
            type="com.ghani.databindingwithlivedata.MainViewModel" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        tools:context=".MainActivity">
        <EditText
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@={mainViewModel.quoteLiveData}"/>
        <TextView
            android:id="@+id/quoteText"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@{mainViewModel.quoteLiveData}"
            android:textAlignment="center"
            android:textSize="28sp"
            android:textStyle="bold" />
        <Button
            android:id="@+id/btnUpdate"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:onClick="@{()->mainViewModel.updateQuote()}"
            android:text="Update Fact"
            android:textAllCaps="false"
            android:textSize="15sp"/>
    </LinearLayout>
</layout>
```



MainActivity.kt:

```
package com.ghani.databindingwithlivedata

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.databinding.DataBindingUtil
import androidx.lifecycle.ViewModelProvider
import com.ghani.databindingwithlivedata.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {

    lateinit var binding: ActivityMainBinding
    lateinit var mainViewModel: MainViewModel

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

        binding = DataBindingUtil.setContentView(this,R.layout.activity_main)
        mainViewModel = ViewModelProvider(this).get(MainViewModel::class.java)

        binding.mainViewModel = mainViewModel
        binding.lifecycleOwner = this
    }
}
```

MainViewModel.kt:

```
package com.ghani.databindingwithlivedata

import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel

class MainViewModel: ViewModel() {
   var quoteLiveData = MutableLiveData<String>("This is a fact")

   fun updateQuote(){
      quoteLiveData.value = "This is another fact"
   }
}
```

Binding Adapter

AndroidManifest.xml:

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

build.gradle:

```
plugins {
   //Adding by me
   id 'kotlin-kapt'
}
android {
   //Adding by me
   buildFeatures{
      dataBinding true
   }
}
dependencies {
   //Adding by me
   implementation 'com.github.bumptech.glide:glide:4.15.1'
   kapt 'com.github.bumptech.glide:compiler:4.15.1'
}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<layout 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">
```

```
    <data>
        <variable
            name="post"
            type="com.ghani.bindingadapter.Post" />
    </data>
    <LinearLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        android:gravity="center"
        android:orientation="vertical"
        tools:context=".MainActivity">
        <ImageView
            android:id="@+id/imageView"
            android:layout_width="200dp"
            android:layout_height="200dp"
            android:scaleType="centerCrop"
            imageFromUrl="@{post.url}"
            tools:srcCompat = "@tools:sample/avatars"/>
        <TextView
            style="@style/TextAppearance.AppCompat.Headline"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="16dp"
            android:text="@{post.title}"
            tools:text="Title"/>
        <TextView
            style="@style/TextAppearance.AppCompat.Medium"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginTop="8dp"
            android:text="@{post.description}"
            tools:text="Description"/>
    </LinearLayout>
</layout>
```

MainActivity.kt:

```
package com.ghani.bindingadapter

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import androidx.databinding.DataBindingUtil
import com.ghani.bindingadapter.databinding.ActivityMainBinding

class MainActivity : AppCompatActivity() {
   lateinit var binding:ActivityMainBinding

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)

      binding = DataBindingUtil.setContentView(this,R.layout.activity_main)

     //val post = Post("Introduction to Kotlin","Cheeze code","https://www.youtube.com/watch?v=R-
     X2nM3d2FI&list=PLRKyZvuMYSIO0jLgj8g6sADnD0IBaWaw2&index=4")       //Error occurred

     val post = Post("Introduction to Kotlin","Cheeze
     code","https://www.gardendesign.com/pictures/images/675x529Max/site_3/helianthus-yellow-
     flower-pixabay_11863.jpg")

      binding.post = post
   }
}
```

Adapters.kt:

```
package com.ghani.bindingadapter

import android.widget.ImageView
import androidx.databinding.BindingAdapter
import com.bumptech.glide.Glide

@BindingAdapter("imageFromUrl")
fun ImageView.imageFromUrl(url:String){
      Glide.with(this.context).load(url).error(R.drawable.ic_launcher_background).into(this)
}
```

Post.kt:

```
package com.ghani.bindingadapter

data class Post(val title:String,val description:String,val url:String)
```

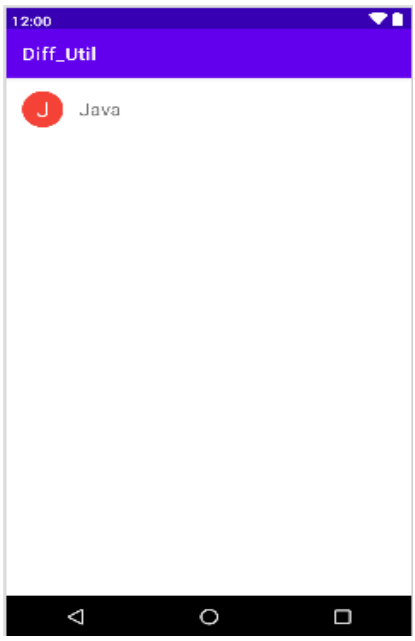
List Adapter(DiffUtil)

item_red.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android">
   <corners android:radius="150dp"></corners>
   <solid android:color="#F44336"></solid>
</shape>
```

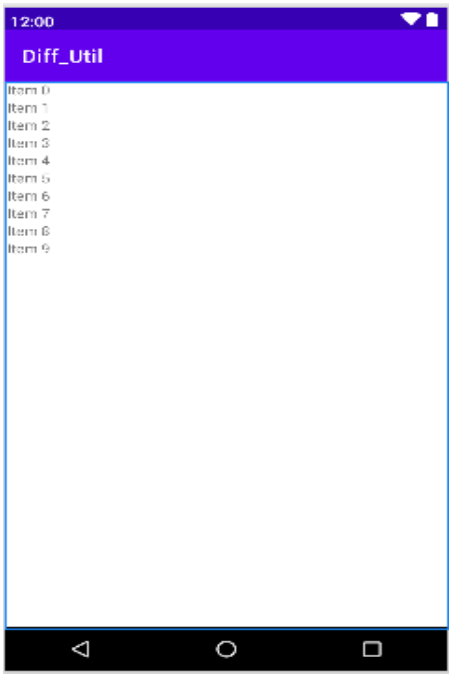
item_view.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="wrap_content"
   android:gravity="center"
   android:padding="16dp">
   <TextView
      android:id="@+id/initial"
      android:layout_width="42dp"
      android:layout_height="42dp"
      android:background="@drawable/item_red"
      android:gravity="center"
      android:text="J"
      android:textSize="24sp"
      android:textColor="@color/white"/>
   <TextView
      android:id="@+id/name"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_gravity="center"
      android:layout_weight="1"
      android:fontFamily="sans-serif"
      android:paddingStart="16dp"
      android:text="Java"
      android:textSize="20sp"/>
</LinearLayout>
```



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.recyclerview.widget.RecyclerView
        android:id="@+id/programmingList"
        android:layout_width="409dp"
        android:layout_height="354dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="1.0"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:layout_constraintVertical_bias="0.0" />

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



MainActivity.kt

```
package com.ghani.diff_util

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import androidx.recyclerview.widget.LinearLayoutManager
import androidx.recyclerview.widget.RecyclerView

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

      val adapter = ProgrammingAdapter()
      val p1 = ProgrammingItem(1,"J","Java")
      val p2 = ProgrammingItem(2,"K","Kotlin")
      val p3 = ProgrammingItem(3,"P","Python")
      adapter.submitList(listOf(p1,p2,p3))

      val recyclerView = findViewById<RecyclerView>(R.id.programmingList)
      recyclerView.layoutManager = LinearLayoutManager(this)
      recyclerView.setHasFixedSize(true)
      recyclerView.adapter = adapter
```

```
        Handler(Looper.getMainLooper()).postDelayed(Runnable{
            val p3 = ProgrammingItem(3,"P","Python")
            val p4 = ProgrammingItem(4,"R","Ruby")
            val p5 = ProgrammingItem(5,"P","Perl")
            val p6 = ProgrammingItem(6,"J","JavaScript")
            adapter.submitList(listOf(p3,p4,p5,p6))
        },9000)
    }
}
```

ProgrammingItem.kt:

```
package com.ghani.diff_util
data class ProgrammingItem(val id:Int,val initial:String,val name:String)
```

ProgrammingAdapter.kt:

```
package com.ghani.diff_util

import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import android.widget.TextView
import androidx.recyclerview.widget.ListAdapter
import androidx.recyclerview.widget.RecyclerView

class ProgrammingAdapter: ListAdapter<ProgrammingItem,
ProgrammingAdapter.ProgrammingViewHolder>(DiffUtil()){
    override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): ProgrammingViewHolder {
        val view = LayoutInflater.from(parent.context).inflate(R.layout.item_view,parent,false)
        return ProgrammingViewHolder(view)
    }
    override fun onBindViewHolder(holder: ProgrammingViewHolder, position: Int) {
        val item = getItem(position)
        holder.bind(item)
    }
    class ProgrammingViewHolder(view: View):RecyclerView.ViewHolder(view){
        val name =  view.findViewById<TextView>(R.id.name)
        val initial =  view.findViewById<TextView>(R.id.initial)

        fun bind(item:ProgrammingItem){
            name.text = item.name
            initial.text = item.initial
        }
    }
```

```
   class DiffUtil:androidx.recyclerview.widget.DiffUtil.ItemCallback<ProgrammingItem>(){
      override fun areItemsTheSame(oldItem: ProgrammingItem, newItem: ProgrammingItem):
Boolean {
         return oldItem.id == newItem.id
      }
      override fun areContentsTheSame(
         oldItem: ProgrammingItem,
         newItem: ProgrammingItem
      ): Boolean {
         return oldItem.id == newItem.id
      }
   }
}
```

Room Database

Why:

- Abstraction over SQLite
- Less Boilerplate
- Compile Time Verification of SQL Queries

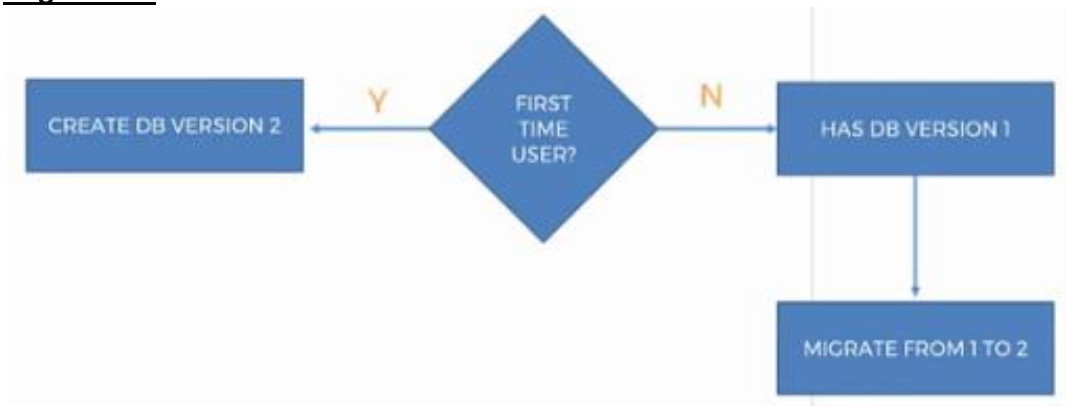
Parts:

- Entities (Tables)
- DAO (Data Access Objects)
- Database
- Type Convertors
- Migrations

Why type converters:

- SQLite only supports –
    1. NULL
    2. INTEGER
    3. REAL
    4. TEXT
    5. BLOB

Migrations:



---

build.gradle:

```
plugins {
   //Adding by me
   id 'kotlin-kapt'
}
dependencies {
   //Adding by me
   def room_version = "2.5.1"
   implementation("androidx.room:room-runtime:$room_version")
   kapt("androidx.room:room-compiler:$room_version")
   implementation("androidx.room:room-ktx:$room_version")
   implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.2")
   implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:1.6.4")
}
```

Contact.kt:

```
package com.ghani.roomdatabase

import androidx.room.Entity
import androidx.room.PrimaryKey
import java.util.*

@Entity(tableName = "contact")
data class Contact(
   @PrimaryKey(autoGenerate = true)
   val id:Long,
   val name:String,
   val phone:String,
   val createdDate:Date,
   val isActive:Int
)
```

ContactDAO.kt:

```
package com.ghani.roomdatabase

import androidx.lifecycle.LiveData
import androidx.room.*

@Dao
interface ContactDAO {
   @Insert
   suspend fun insertContact(contact:Contact)   //suspend fun execute in background

   @Update
   suspend fun updateContact(contact:Contact)

   @Delete
   suspend fun deleteContact(contact:Contact)

   @Query("SELECT * FROM contact")
   fun getContact():LiveData<List<Contact>>    //LiveData execute in background
}
```

ContactDatabase.kt:

```
package com.ghani.roomdatabase

import android.content.Context
import androidx.room.*
import androidx.room.migration.Migration
import androidx.sqlite.db.SupportSQLiteDatabase

@Database(entities = [Contact::class],version = 2)
@TypeConverters(Converters::class)
abstract class ContactDatabase:RoomDatabase() {
   abstract fun contactDao():ContactDAO

   companion object{
      @Volatile      //all the threads can knowing the updated value instantly
      private var INSTANCE:ContactDatabase? = null

      val migration_1_2 = object : Migration(1,2){
         override fun migrate(database: SupportSQLiteDatabase) {
            database.execSQL("ALTER TABLE contact ADD COLUMN isActive NOT NULL INTEGER DEFAULT(1)")
         }
      }
```

```
    fun getDatabase(context: Context):ContactDatabase{
        if (INSTANCE == null){
            synchronized(this){
                INSTANCE = Room
                    .databaseBuilder(context.applicationContext,ContactDatabase::class.java,"contactDB")
                    .addMigrations(migration_1_2)
                    .build()
            }
        }
        return INSTANCE!!
    }

}
}
```

Converters.kt:

```
package com.ghani.roomdatabase

import androidx.room.TypeConverter
import java.util.*

class Converters {
    @TypeConverter
    fun fromDateToLong(value:Date): Long {
        return value.time
    }
    @TypeConverter
    fun fromLongToDate(value:Long): Date {
        return Date(value)
    }
}
```

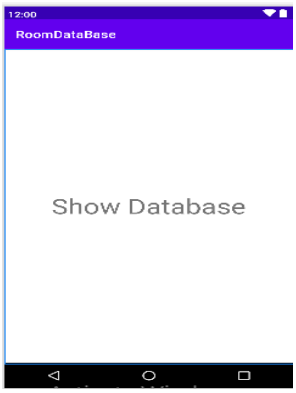
MainActivity.kt:

```
package com.ghani.roomdatabase

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import androidx.lifecycle.Observer
import kotlinx.coroutines.GlobalScope
import kotlinx.coroutines.launch
import java.util.*
```

```
class MainActivity : AppCompatActivity() {

    lateinit var database: ContactDatabase

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

        database = ContactDatabase.getDatabase(this)
        val database2 = ContactDatabase.getDatabase(this)

        GlobalScope.launch {
            database.contactDao().insertContact(Contact(0,"Rony","019387585786", Date(),1))
        }
    }

    fun getData(view: View) {
        database.contactDao().getContact().observe(this, Observer {
            Log.d("Database",it.toString())
        })
    }
}
```

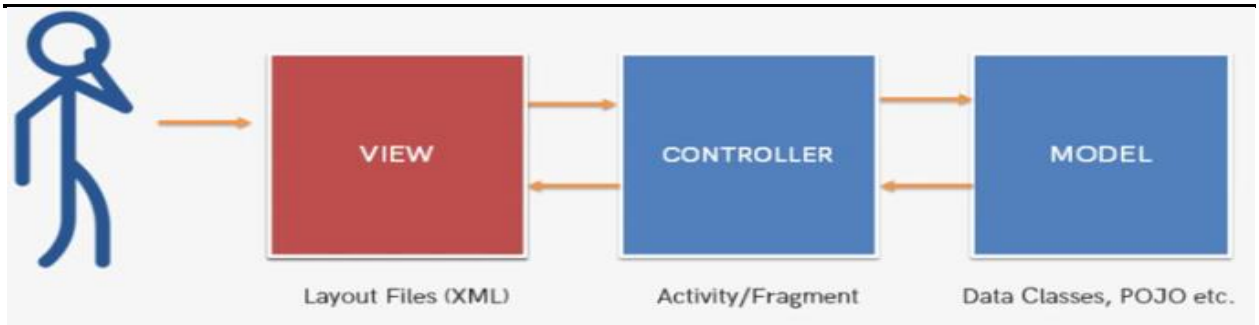
activity_main.xml:

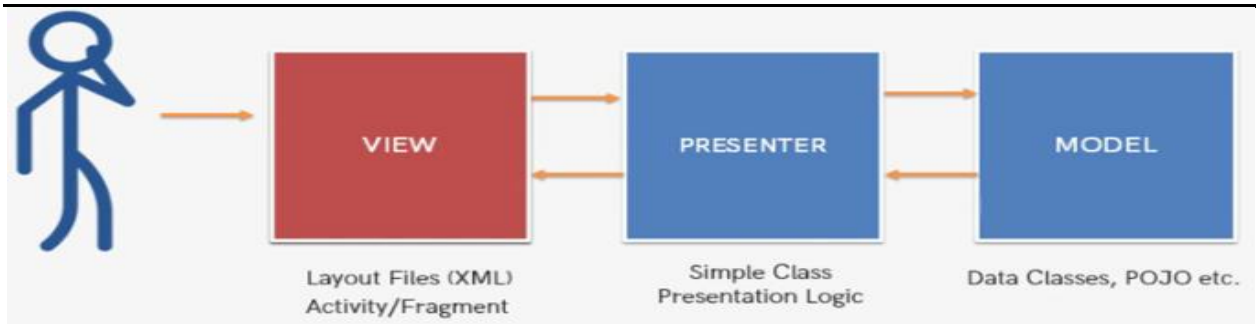
```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".MainActivity">
    <TextView
        android:onClick="getData"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Show Database"
        android:textAllCaps="false"
        android:textSize="40sp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

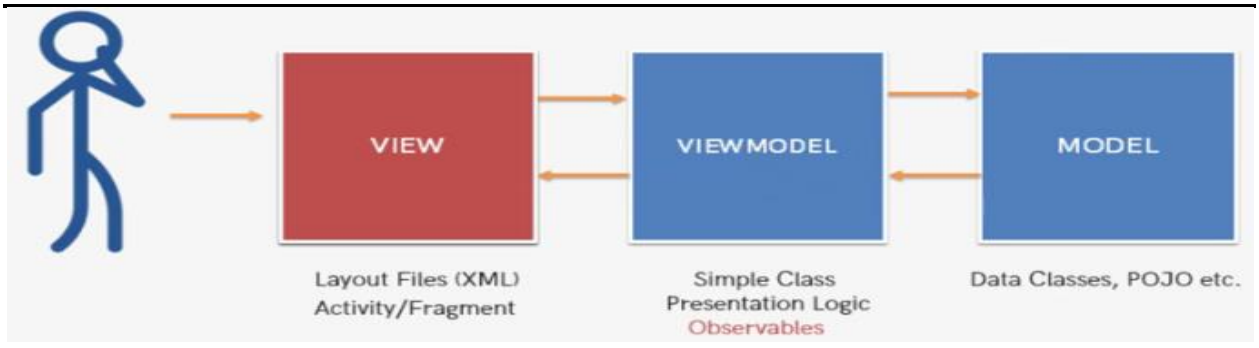


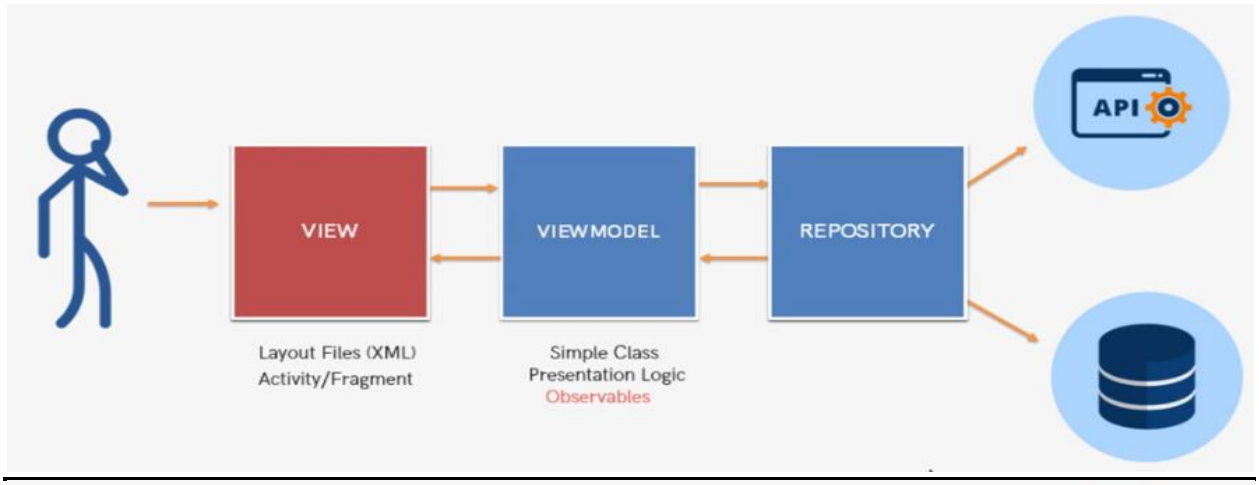
Architecture Pattern:

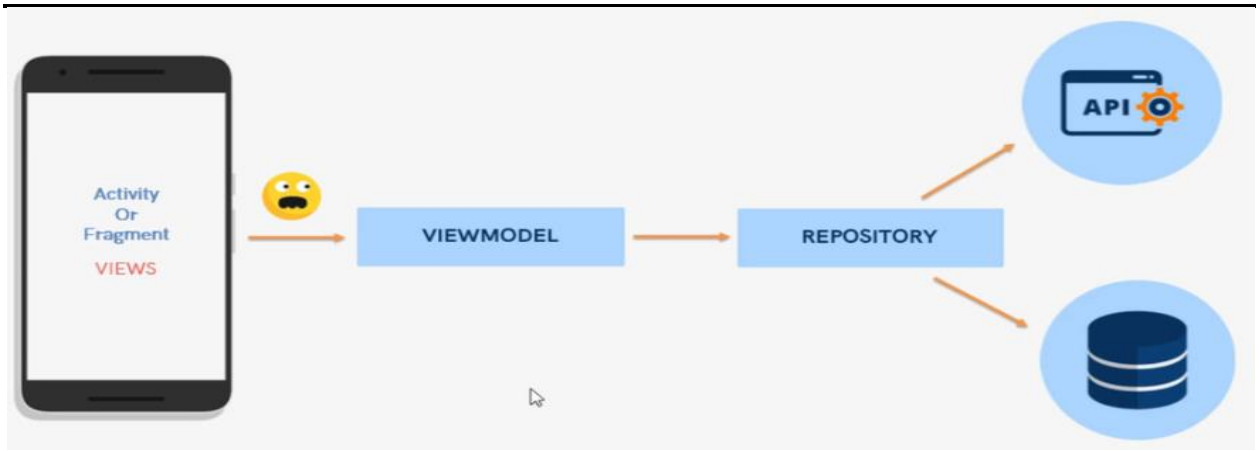
- To achieve this – we implement architecture patterns - MVC, MVP, MVVM etc.
- Architecture patterns mainly focuses on -
  - Separation Of Concerns
  - Unit Testing

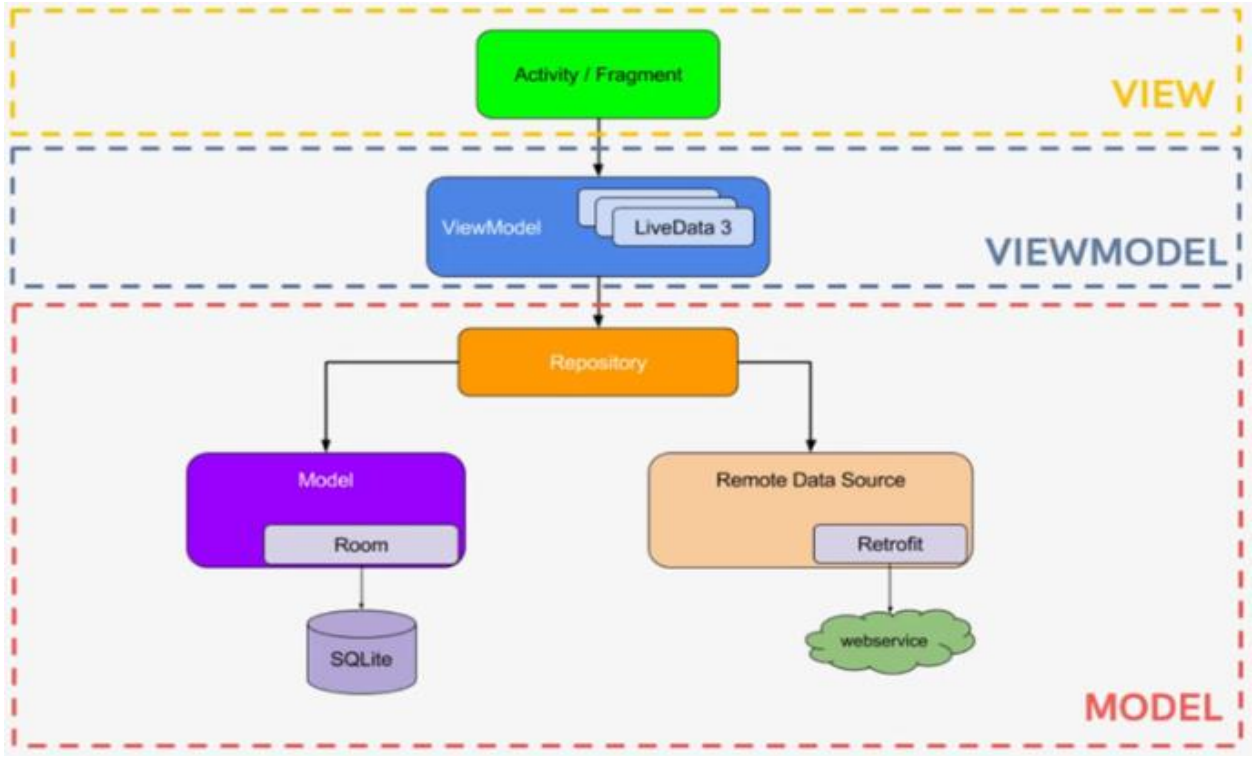







API
VIEW
VIEWMODEL
REPOSITORY
Layout Files (XML)
Activity/Fragment
Simple Class
Presentation Logic
Observables


API
Activity
Or
Fragment
VIEWS
VIEWMODEL
REPOSITORY


Activity / Fragment
VIEW
ViewModel
LiveData 3
VIEWMODEL
Repository
Model
Room
SQLite
Remote Data Source
Retrofit
webservice
MODEL

MODEL - VIEW-VIEWMODEL:

- Room Database and/or Retrofit Setup
- Repository
- LiveData, ViewModel with ViewModelFactory
- Activity/Fragment with Data Binding

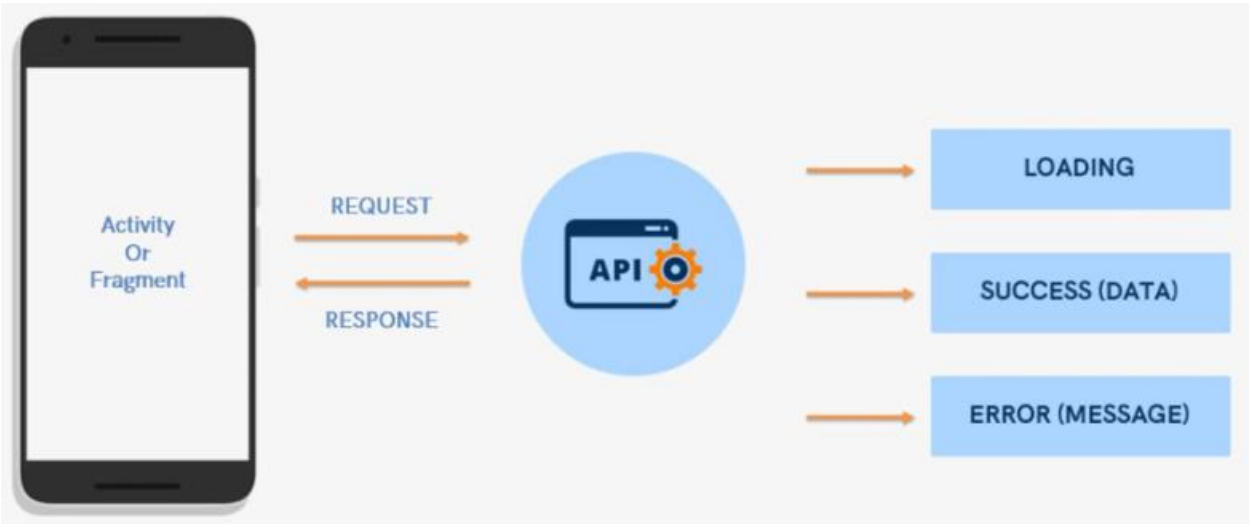
Room Database vs Retrofit:

- Entities (Tables) ---> Data Classes
- DAO (Data Access Objects) ---> Interface
- Database ---> Build Retrofit Object

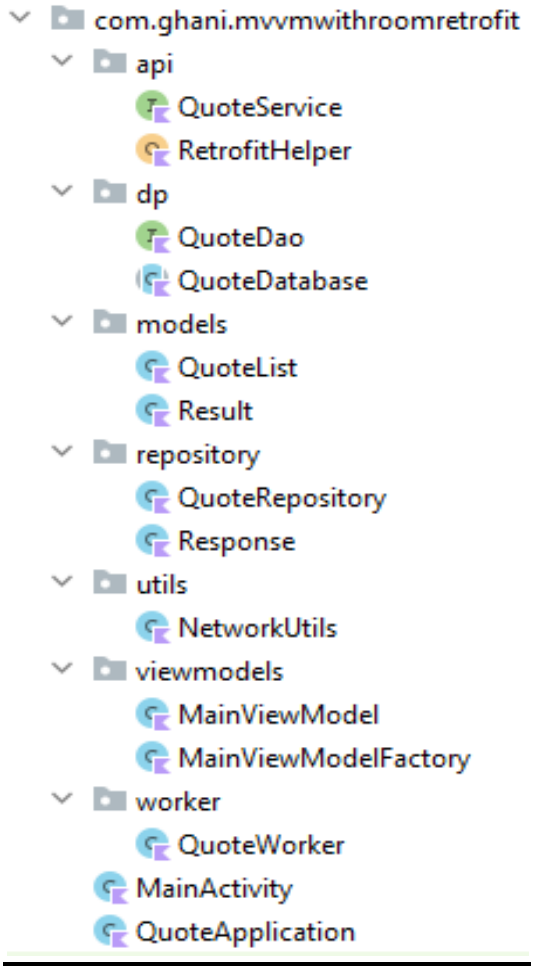
Work Manager:

- Schedule tasks that can run even if app exists or device restarts.
- Simply, run background jobs using Work Manager
- Example - Sync Data From Server

Error Handling:



MVVM with Room & Retrofit:



---

build.gradle:

```
plugins {
   //Adding by me
   id'kotlin-kapt'
}
dependencies {
   //Adding by me
   implementation"org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.1"
   implementation"org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.1"
   def lifecycle_version = "2.5.1"
   implementation"androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
   implementation"androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"
   def room_version = "2.5.1"
   implementation"androidx.room:room-runtime:$room_version"
   kapt"androidx.room:room-compiler:$room_version"
   implementation"androidx.room:room-ktx:$room_version"
```

```
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation'androidx.work:work-runtime:2.7.1'
}
```

AndroidManifest.xml:

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

<application
android:usesCleartextTraffic="true"
</application>
```

QuoteList.kt:

```
package com.ghani.mvvmwithroomretrofit.models

data class QuoteList(
    val count: Int,
    val lastItemIndex: Int,
    val page: Int,
    val results: List<Result>,
    val totalCount: Int,
    val totalPages: Int
)
```

Result.kt:

```
package com.ghani.mvvmwithroomretrofit.models

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity(tableName = "quote")
data class Result(
    @PrimaryKey(autoGenerate = true)
    val quoteId:Int,
    val _id: String,
    val author: String,
    val authorSlug: String,
    val content: String,
    val dateAdded: String,
    val dateModified: String,
    val length: Int,
)
```

QuoteService.kt:

```
package com.ghani.mvvmwithroomretrofit.api

import com.ghani.mvvmwithroomretrofit.models.QuoteList
import retrofit2.Response
import retrofit2.http.GET
import retrofit2.http.Query

interface QuoteService {

    @GET("/quotes")
    suspend fun getQuotes(@Query("page")page:Int): Response<QuoteList>

    //BASE_URL + "/quotes?page=1
}
```

RetrofitHelper.kt:

```
package com.ghani.mvvmwithroomretrofit.api

import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

object RetrofitHelper {
    val BASE_URL = "http://quotable.io/"

    fun getInstance(): Retrofit {
        return Retrofit.Builder()
            .baseUrl(BASE_URL)
            .addConverterFactory(GsonConverterFactory.create())
            .build()
    }
}
```

QuoteDao.kt:

```
package com.ghani.mvvmwithroomretrofit.dp

import androidx.room.*
import com.ghani.mvvmwithroomretrofit.models.Result

@Dao
interface QuoteDao {

   @Insert
   suspend fun addQuotes(quotes:List<Result>)

   @Query("SELECT * FROM quote")
   suspend fun getQuotes():List<Result>
}
```

QuoteDatabase.kt:

```
package com.ghani.mvvmwithroomretrofit.dp

import android.content.Context
import androidx.room.*
import com.ghani.mvvmwithroomretrofit.models.Result

@Database(entities = [Result::class], version = 1)
abstract class QuoteDatabase : RoomDatabase() {

   abstract fun quoteDao(): QuoteDao

   companion object {
      @Volatile
      private var INSTANCE: QuoteDatabase? = null

      fun getDatabase(context: Context): QuoteDatabase {
         if (INSTANCE == null) {
            synchronized(this) {
               INSTANCE = Room
                        .databaseBuilder(context.applicationContext,QuoteDatabase::class.java,"quoteDB")
                        .build()
            }
         }
         return INSTANCE!!
      }
   }
}
```

QuoteRepository.kt:

```
package com.ghani.mvvmwithroomretrofit.repository

import android.content.Context
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import com.ghani.mvvmwithroomretrofit.api.QuoteService
import com.ghani.mvvmwithroomretrofit.dp.QuoteDatabase
import com.ghani.mvvmwithroomretrofit.models.QuoteList
import com.ghani.mvvmwithroomretrofit.utils.NetworkUtils

class QuoteRepository(
    private val quoteService: QuoteService,
    private val quoteDatabase: QuoteDatabase,
    private val applicationContext: Context
) {
    private val quotesLiveData = MutableLiveData<Response<QuoteList>>()
    val quotes:LiveData<Response<QuoteList>>
    get() = quotesLiveData

    suspend fun getQuotes(page:Int){
        if (NetworkUtils.isInternetAvailable(applicationContext)){
            try {
                val result = quoteService.getQuotes(page)
                if (result?.body() != null){
                    quoteDatabase.quoteDao().addQuotes(result.body()!!.results)
                    quotesLiveData.postValue(Response.Success(result.body()))
                }
                else{
                    quotesLiveData.postValue(Response.Error("API error"))
                }
            }
            catch (e:Exception){
                quotesLiveData.postValue(Response.Error(e.message.toString()))
            }
        }
        else{
            try {
                val quote = quoteDatabase.quoteDao().getQuotes()
                val quoteList = QuoteList(1,1,1,quote,1,1)
                quotesLiveData.postValue(Response.Success(quoteList))
            }
            catch (e:Exception){
                quotesLiveData.postValue(Response.Error(e.message.toString()))
```

```
        }
      }
    }
    suspend fun getQuotesBackground(){
      val randomNumber = (Math.random()*10).toInt()
      val result = quoteService.getQuotes(randomNumber)
      if (result?.body() != null){
        quoteDatabase.quoteDao().addQuotes(result.body()!!.results)
      }
    }
}
```

QuoteApplication.kt:

```
package com.ghani.mvvmwithroomretrofit
import android.app.Application
import androidx.work.Constraints
import androidx.work.NetworkType
import androidx.work.PeriodicWorkRequest
import androidx.work.WorkManager
import com.ghani.mvvmwithroomretrofit.api.QuoteService
import com.ghani.mvvmwithroomretrofit.api.RetrofitHelper
import com.ghani.mvvmwithroomretrofit.dp.QuoteDatabase
import com.ghani.mvvmwithroomretrofit.repository.QuoteRepository
import com.ghani.mvvmwithroomretrofit.worker.QuoteWorker
import java.util.concurrent.TimeUnit

class QuoteApplication: Application() {
    lateinit var quoteRepository: QuoteRepository
    override fun onCreate() {
        super.onCreate()
        initialize()
        setupWorker()
    }
    private fun initialize() {
        val quoteService = RetrofitHelper.getInstance().create(QuoteService::class.java)
        val database = QuoteDatabase.getDatabase(applicationContext)
        quoteRepository = QuoteRepository(quoteService,database,applicationContext)
    }
    private fun setupWorker() {
        val constraint = Constraints.Builder().setRequiredNetworkType(NetworkType.CONNECTED).build()
        val workerRequest = PeriodicWorkRequest.Builder(QuoteWorker::class.java, 15, TimeUnit.MINUTES)
            .setConstraints(constraint)
            .build()
        WorkManager.getInstance(this).enqueue(workerRequest)
```

```
    }
}
```

Response.kt:

```
package com.ghani.mvvmwithroomretrofit.repository

sealed class Response<T> (val data:T? = null, val errorMessage:String? = null){
    class Loading<T> :Response<T> ()
    class Success<T> (data: T? = null) :Response<T> (data = data)
    class Error<T> (errorMessage: String? = null) :Response<T> (errorMessage = errorMessage)
}
```

NetworkUtils.kt:

```
package com.ghani.mvvmwithroomretrofit.utils

import android.content.Context
import android.net.ConnectivityManager
import android.net.NetworkCapabilities
import android.os.Build

class NetworkUtils {

    companion object{
        fun isInternetAvailable(context: Context): Boolean {
            (context.getSystemService(Context.CONNECTIVITY_SERVICE) as ConnectivityManager).run {
                if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.M) {
                    return this.getNetworkCapabilities(this.activeNetwork)?.hasCapability(
                        NetworkCapabilities.NET_CAPABILITY_INTERNET
                    ) ?: false
                } else {
                    (@Suppress("DEPRECATION")
                    return this.activeNetworkInfo?.isConnected ?: false)
                }
            }
        }

    }
}
```

QuoteWorker.kt:

```
package com.ghani.mvvmwithroomretrofit.worker

import android.content.Context
import android.util.Log
```

```
import androidx.work.Worker
import androidx.work.WorkerParameters
import com.ghani.mvvmwithroomretrofit.QuoteApplication
import kotlinx.coroutines.CoroutineScope
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch

class QuoteWorker(private val context: Context, params: WorkerParameters): Worker(context, params)
{
   override fun doWork(): Result {
      Log.d("MVVM", "Worker called")
      val repository = (context as QuoteApplication).quoteRepository
      CoroutineScope(Dispatchers.IO).launch {
         repository.getQuotesBackground()
      }
      return Result.success()
   }
}
```

MainViewModelFactory.kt:

```
package com.ghani.mvvmwithroomretrofit.viewmodels

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.ghani.mvvmwithroomretrofit.repository.QuoteRepository

class MainViewModelFactory(private val repository: QuoteRepository):
ViewModelProvider.Factory {
   override fun <T : ViewModel> create(modelClass: Class<T>): T {
      return MainViewModel(repository) as T
   }
}
```

MainViewModel.kt:

```
package com.ghani.mvvmwithroomretrofit.viewmodels

import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.ghani.mvvmwithroomretrofit.models.QuoteList
import com.ghani.mvvmwithroomretrofit.repository.QuoteRepository
import com.ghani.mvvmwithroomretrofit.repository.Response
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
```

```
class MainViewModel(private val repository: QuoteRepository):ViewModel() {
    init {
        viewModelScope.launch(Dispatchers.IO){
            repository.getQuotes(1)
        }
    }
    val quotes: LiveData<Response<QuoteList>>
        get() = repository.quotes
}
```

MainActivity.kt:

```
package com.ghani.mvvmwithroomretrofit
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.Toast
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import com.ghani.mvvmwithroomretrofit.repository.Response
import com.ghani.mvvmwithroomretrofit.viewmodels.MainViewModel
import com.ghani.mvvmwithroomretrofit.viewmodels.MainViewModelFactory

class MainActivity : AppCompatActivity() {
    lateinit var mainViewModel: MainViewModel

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

        val repository = (application as QuoteApplication).quoteRepository
        mainViewModel = ViewModelProvider(this,MainViewModelFactory(repository)).get(MainViewModel::class.java)
        mainViewModel.quotes.observe(this, Observer {
            when(it){
                is Response.Loading -> {}
                is Response.Success -> {
                    it.data?.let {
                        Toast.makeText(this@MainActivity,it.results.size.toString(),Toast.LENGTH_LONG).show()
                    }
                }
                is Response.Error -> {
                    Toast.makeText(this@MainActivity,"Some error occurred",Toast.LENGTH_LONG).show()
                }
            }
        })
    }
}
```

Dependency Injection(Dagger2)

Flow-1,2[UserRepository > EmailService > UserRegistrationService > MainActivity]

Flow-3,4[UserRepository > EmailService > UserRegistrationService > UserRegistrationComponent > MainActivity]

Flow-5,6,7,8,9[UserRepository > EmailService > UserRegistrationService > NotificationServiceModule > UserRepositoryModule > UserRegistrationComponent > MainActivity]

Flow-10[MainActivity]

Flow-11,12[EmailService > UserRegistrationComponent > MainActivity]

Flow-13[EmailService > MainActivity]

Flow-14[MainActivity]

Flow-15[UserApplication > MainActivity]

Flow-16[EmailService > NotificationServiceModule > UserRepository > UserRepositoryModule > UserRegistrationComponent > AnalyticsService(> AnalyticsModule > AppComponent > UserApplication > MainActivity]

Flow-17[UserRegistrationComponent > AppComponent > MainActivity]

Flow-18[UserRegistrationComponent > NotificationServiceModule > AppComponent > MainActivity]

Flow-19,20[UserRegistrationComponent > AppComponent > MainActivity]

Dependency Injection(Dagger2)

Goal:

(1)Unit testing
(2)Single responsibility
(3)Lifetime of these objects
(4)Extensible

Concept-

Consumer (@Inject)

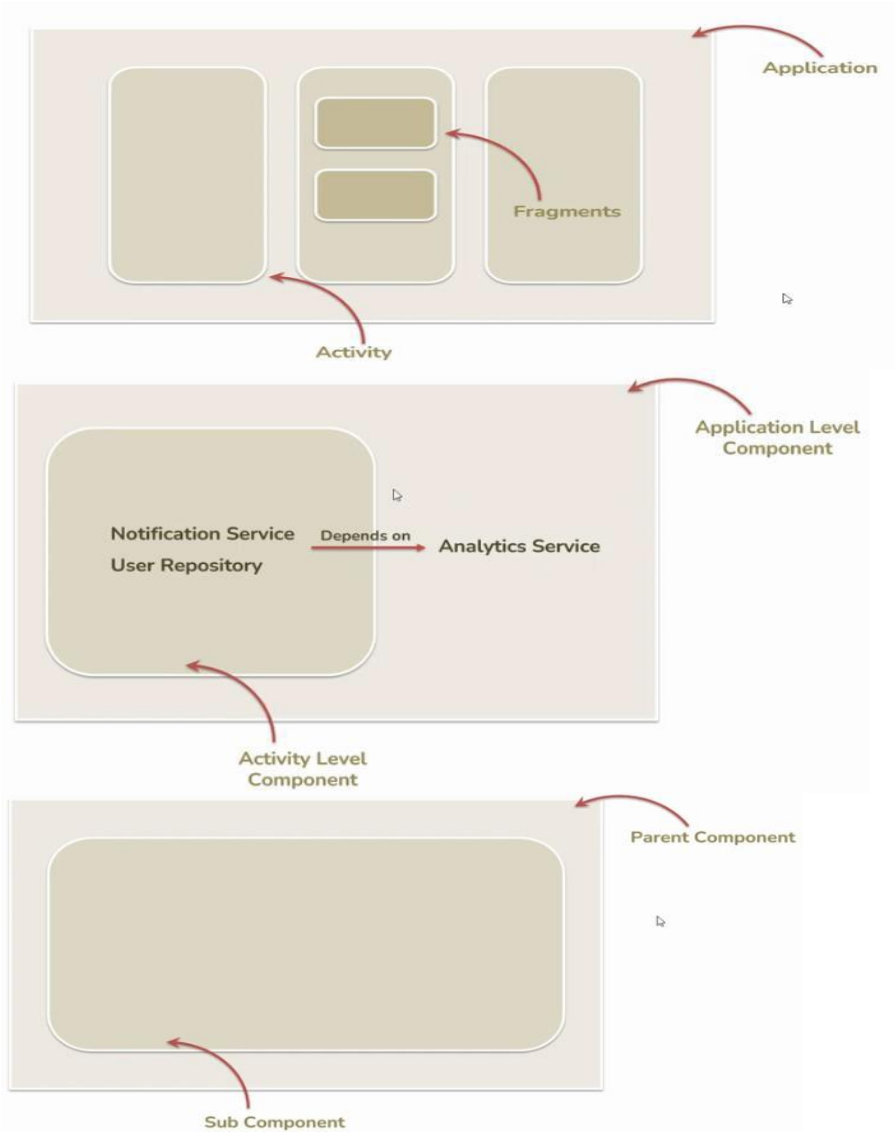
Producer (@Module,@Provides,@Binds)

Connecter (@component)

Singleton:

Just one instance i.e. single object for the whole application.

HTTP Client, Database Connection etc.



build.gradle:

```
plugins {
   //Adding by me
   id 'kotlin-kapt'  }
dependencies  {
   //Adding by me
   implementation 'com.google.dagger:dagger:2.44'
   kapt 'com.google.dagger:dagger-compiler:2.44'  }
```

AndroidManifest.xml:

```
<application
   android:name=".UserApplication">
</application>
```

ActivityScope.kt:

```
package com.ghani.dependencyinjection_1
import java.lang.annotation.Documented
import java.lang.annotation.Retention
import java.lang.annotation.RetentionPolicy
import javax.inject.Scope
@Scope
@Documented
@Retention(RetentionPolicy.RUNTIME)
annotation class ActivityScope()
```

ApplicationScope.kt:

```
package com.ghani.dependencyinjection_1
import java.lang.annotation.Documented
import java.lang.annotation.Retention
import java.lang.annotation.RetentionPolicy
import javax.inject.Scope
@Scope
@Documented
@Retention(RetentionPolicy.RUNTIME)
annotation class ApplicationScope()
```

MessageQualifier.kt:

```
package com.ghani.dependencyinjection_1
import java.lang.annotation.Documented
import java.lang.annotation.Retention
import java.lang.annotation.RetentionPolicy
import javax.inject.Qualifier
@Qualifier
@Documented
@Retention(RetentionPolicy.RUNTIME)
annotation class MessageQualifier()
```

MainActivity.kt:

```
package com.ghani.dependencyinjection_1

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import dagger.internal.MapFactory.builder
import dagger.internal.SetFactory.builder
import java.util.stream.DoubleStream.builder
import javax.inject.Inject

class MainActivity : AppCompatActivity() {
    @Inject
    lateinit var userRegistrationService:UserRegistrationService

    /*
    //Flow-10,11 (Checking singleton)
    @Inject
    lateinit var emailService:EmailService
    @Inject
    lateinit var emailService1:EmailService
    */

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

        //Flow-1
        //val  userRegistrationService  =  UserRegistrationService()
        /*
        //Flow-2
        val userRepository = UserRepository()
        val emailService = EmailService()
        val  userRegistrationService  =  UserRegistrationService(userRepository,emailService)
        */
        //val component1 = DaggerUserRegistrationComponent.builder().build()
        //val component2 = DaggerUserRegistrationComponent.builder().
                                notificationServiceModule(NotificationServiceModule(3)).build()
        //val component3 = DaggerUserRegistrationComponent.factory().create(3)
        /*
        //Flow-12,13  (Checking  singleton)
        val emailService = component3.getEmailService()
        val  emailService1  = component3.getEmailService()
        */
```

```
        /*
        //Flow-14 (Checking singleton)
        val component4 = DaggerUserRegistrationComponent.factory().create(3)
        val emailService = component4.getEmailService()
        val component5 = DaggerUserRegistrationComponent.factory().create(3)
        val emailService1 = component5.getEmailService()
        */
        /*
        //Flow-15 (Checking singleton)
        val component6 = (application as UserApplication).userRegistrationComponent
        val emailService = component6.getEmailService()
        val component7 = (application as UserApplication).userRegistrationComponent
        val emailService1 = component7.getEmailService()
        */
        //Flow-3
        //val userRegistrationService = component1.getUserRegistrationService()
        //Flow-4,5,6,7
        //component1.inject(this)
        //Flow-8
        //component2.inject(this)
        //Flow-9
        //component3.inject(this)

        val appComponent = (application as UserApplication).appComponent
        //Flow-16
        //val userRegistrationComponent = DaggerUserRegistrationComponent.factory()
                                          .create(3,appComponent)
         //Flow-17
        //val userRegistrationComponent = appComponent
                                          .getUserRegistrationComponentFactory().create(3)
        //Flow-18
        //val userRegistrationComponent = appComponent.getUserRegistrationComponent()
        //Flow-19
        //val userRegistrationComponent = appComponent
                                          .getUserRegistrationComponentBuilder()
                                          .retryCount(3).build()
        //Flow-20
        val userRegistrationComponent = DaggerUserRegistrationComponent.builder()
                                        .appComponent(appComponent).retryCount(3).build()

        userRegistrationComponent.inject(this)
        userRegistrationService.registerUser("cheezycode@gmail.com","11111")
    }
}
```

UserApplication.kt:

```
package com.ghani.dependencyinjection_1
import android.app.Application

/*
//Flow-15
class UserApplication: Application() {
   lateinit var userRegistrationComponent: UserRegistrationComponent
   override fun onCreate(){
      super.onCreate()
      userRegistrationComponent = DaggerUserRegistrationComponent.factory().create(3)
   }
}
*/
//Flow-16
class UserApplication: Application() {
   lateinit var appComponent: AppComponent
   override fun onCreate(){
      super.onCreate()
      appComponent = DaggerAppComponent.builder().build()
   }
}
```

AppComponent.kt:

```
package com.ghani.dependencyinjection_1
import dagger.BindsInstance
import dagger.Component
import javax.inject.Singleton

//Flow-16,20
@Singleton
@Component(modules = [AnalyticsModule::class])
interface AppComponent {
   fun getAnalyticsService(): AnalyticsService
}

/*
//Flow-17
@Singleton
@Component(modules = [AnalyticsModule::class])
interface AppComponent {
   fun getUserRegistrationComponentFactory() : UserRegistrationComponent.Factory
}
*/
```

```
/*
//Flow-18
@Singleton
@Component(modules = [AnalyticsModule::class])
interface AppComponent {
   fun  getUserRegistrationComponent() :  UserRegistrationComponent
}
*/

/*
//Flow-19
@Singleton
@Component(modules = [AnalyticsModule::class])
interface AppComponent {
   fun  getUserRegistrationComponentBuilder() :  UserRegistrationComponent.Builder
}
*/
```

UserRegistrationComponent.kt:

```
package  com.ghani.dependencyinjection_1

import dagger.BindsInstance
import dagger.Component
import dagger.Subcomponent
import javax.inject.Singleton

/*
//Flow-3
@Component
interface  UserRegistrationComponent  {
   fun getUserRegistrationService() : UserRegistrationService
   fun getEmailService() : EmailService
}
*/

/*
//Flow-4
@Component
interface  UserRegistrationComponent  {
   fun  inject(mainActivity:  MainActivity)      //fun name does not matter, but data type matter
}
*/
```

```
/*
//Flow-5,6,7,8
@Component(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

}
*/


/*
//Flow-9
@Component(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

   @Component.Factory
   interface Factory{
      fun create(@BindsInstance retryCount:Int):UserRegistrationComponent
      //@BindsInstance use for pass value on runtime
   }
}
*/


/*
//Flow-11
@Singleton
@Component(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

   @Component.Factory
   interface Factory{
      fun create(@BindsInstance retryCount:Int):UserRegistrationComponent
   }
}
*/
```

```
/*
//Flow-12
@Singleton
@Component(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {
   fun inject(mainActivity: MainActivity)
   fun getEmailService(): EmailService

   @Component.Factory
   interface Factory{
      fun create(@BindsInstance retryCount:Int):UserRegistrationComponent
   }
}
*/

/*
//Flow-16
@ActivityScope
@Component(dependencies = [AppComponent::class],modules =
[UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {
   fun inject(mainActivity: MainActivity)

   @Component.Factory
   interface Factory{
      fun create(@BindsInstance retryCount:Int,appComponent: AppComponent):UserRegistrationComponent
   }
}
*/

/*
//Flow-17
@ActivityScope
@Subcomponent(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {
   fun inject(mainActivity: MainActivity)

   @Subcomponent.Factory
   interface Factory{
      fun create(@BindsInstance retryCount:Int):UserRegistrationComponent
   }
}
*/
```

```
/*
//Flow-18
@ActivityScope
@Subcomponent(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

}
*/

/*
//Flow-19
@ActivityScope
@Subcomponent(modules = [UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

   @Subcomponent.Builder
   interface Builder{
      fun build():UserRegistrationComponent
      fun retryCount(@BindsInstance retryCount:Int):Builder
    }
}
*/

//Flow-20
@ActivityScope
@Component(dependencies = [AppComponent::class],modules =
[UserRepositoryModule::class,NotificationServiceModule::class])
interface UserRegistrationComponent {

   fun inject(mainActivity: MainActivity)

   @Component.Builder
   interface Builder{
      fun build():UserRegistrationComponent
      fun retryCount(@BindsInstance retryCount:Int):Builder
      fun appComponent(appComponent: AppComponent):Builder
   }
}
```

UserRepositoryModule.kt:

```
package com.ghani.dependencyinjection_1

import dagger.Binds
import dagger.Module
import dagger.Provides
import javax.inject.Singleton

/*
//Flow-5,6,7,8,9
@Module
abstract class UserRepositoryModule {
   @Binds
   abstract fun getSQLRepository(sqlRepository: SQLRepository):UserRepository
}
*/

//Flow-16
@Module
abstract class UserRepositoryModule {
   @ActivityScope
   @Binds
   abstract fun getSQLRepository(sqlRepository: SQLRepository):UserRepository
}
```

NotificationServiceModule.kt:

```
package com.ghani.dependencyinjection_1

import dagger.Module
import dagger.Provides
import javax.inject.Named

/*
//Flow-5
@Module
class NotificationServiceModule() {
   @Provides
   fun getMessageService(): NotificationService {
      return MessageService()
   }
}
*/
```

```
/*
//Flow-6
@Module
class  NotificationServiceModule()  {

   @MessageQualifier
   @Provides
   fun getMessageService(): NotificationService {
      return MessageService()
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
*/


/*
//Flow-7
@Module
class  NotificationServiceModule()  {

   @MessageQualifier
   @Provides
   fun getMessageService(): NotificationService {    //pass value through component
      return MessageService(3)
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
*/
```

```
 /*
//Flow-8
@Module
class NotificationServiceModule(private val retryCount:Int) {  //pass value through module

   @MessageQualifier
   @Provides
   fun getMessageService():NotificationService{
      return MessageService(retryCount)
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
*/


/*
//Flow-9
@Module
class  NotificationServiceModule()  {

   @MessageQualifier
   @Provides
   fun getMessageService(retryCount:Int):NotificationService{      //pass value through
component
      return MessageService(retryCount)
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
*/
```

```
/*
//Flow-16
@Module
class  NotificationServiceModule()  {

   @ActivityScope
   @MessageQualifier
   @Provides
   fun getMessageService(retryCount:Int):NotificationService{
      return MessageService(retryCount)
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
*/



//Flow-18
@Module
class  NotificationServiceModule()  {

   @ActivityScope
   @MessageQualifier
   @Provides
   fun getMessageService():NotificationService{
      return MessageService(3)
   }

   @Named("email")
   @Provides
   fun getEmailService(emailService:EmailService):NotificationService{
      return emailService
   }
}
```

AnalyticsModule.kt:

```
package com.ghani.dependencyinjection_1

import dagger.Module
import dagger.Provides
import javax.inject.Singleton

//Flow-16
@Module
class AnalyticsModule {

    @Singleton
    @Provides
    fun getAnalyticsService():AnalyticsService {
        return Mixpanel()
    }
}
```

UserRegistrationService.kt:

```
package com.ghani.dependencyinjection_1

import javax.inject.Inject
import javax.inject.Named
/*
//Flow-1
class UserRegistrationService {

    private val userRepository = UserRepository()
    private val emailService = EmailService()

    fun registerUser(email: String, password: String) {

        userRepository.saveUser(email, password)
        emailService.send(email, "no-reply@cheezycode.com", "User Registered")
    }
}
*/
```

```
/*
//Flow-2
class UserRegistrationService(private val userRepository:UserRepository,
private val emailService:EmailService) {
   fun registerUser(email: String, password: String) {
      userRepository.saveUser(email, password)
      emailService.send(email,  "no-reply@cheezycode.com",  "User  Registered")
   }
}
*/
/*
//Flow-3,4
class  UserRegistrationService @Inject  constructor(private val  userRepository:UserRepository,
private val emailService:EmailService) {
   fun registerUser(email: String, password: String) {
      userRepository.saveUser(email, password)
      emailService.send(email,  "no-reply@cheezycode.com",  "User  Registered")
   }
}
*/
/*
//Flow-5
class UserRegistrationService @Inject constructor(
   private val userRepository: UserRepository,
   private val notificationService: NotificationService
) {
   fun registerUser(email: String, password: String) {
      userRepository.saveUser(email, password)
      notificationService.send(email,  "no-reply@cheezycode.com",  "User  Registered")
   }
}
*/
//Flow-6,7,8,9
class UserRegistrationService @Inject constructor(
   private val userRepository: UserRepository,
   @MessageQualifier  private val  notificationService:  NotificationService
) {
   fun registerUser(email: String, password: String) {
      userRepository.saveUser(email, password)
      notificationService.send(email,  "no-reply@cheezycode.com",  "User  Registered")
   }
}
```

UserRepository.kt:

```
package com.ghani.dependencyinjection_1

import android.content.ContentValues.TAG
import android.util.Log
import javax.inject.Inject
import javax.inject.Singleton


/*
//Flow-1,2
class UserRepository {
   fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in DB")
   }
}
*/

/*
//Flow-3,4
class UserRepository @Inject constructor() {
   fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in DB")
   }
}
*/

/*
//Flow- 5,6,7,8,9
interface UserRepository {
   fun saveUser(email:String, password:String)
}
class SQLRepository @Inject constructor():UserRepository{
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in DB")
   }
}
class FirebaseRepository:UserRepository{
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in Firebase")
   }
}
*/
```

```
//Flow- 16
interface UserRepository {
   fun saveUser(email:String, password:String)
}
@ActivityScope
class SQLRepository @Inject constructor(val analyticsService: AnalyticsService):UserRepository{
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in DB")
      analyticsService.trackEvent("Save user","Create")
   }
}
class FirebaseRepository(val analyticsService: AnalyticsService):UserRepository{
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User saved in Firebase")
      analyticsService.trackEvent("Save user","Create")
   }
}
```

EmailService.kt:

```
package com.ghani.dependencyinjection_1

import android.content.ContentValues.TAG
import android.util.Log
import javax.inject.Inject
import javax.inject.Singleton

/*
//Flow-1,2
class EmailService {
   fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Email sent")
   }
}
*/

/*
//Flow-3,4
class EmailService @Inject constructor() {
   fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Email sent")
   }
}
*/
```

```
/*
//Flow-5,6
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

class  MessageService  :  NotificationService  {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent")
   }
}
*/




/*
//Flow-7,8,9
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

class MessageService(private val retryCount:Int) : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent - Retry Count $retryCount")
   }
}
*/
```

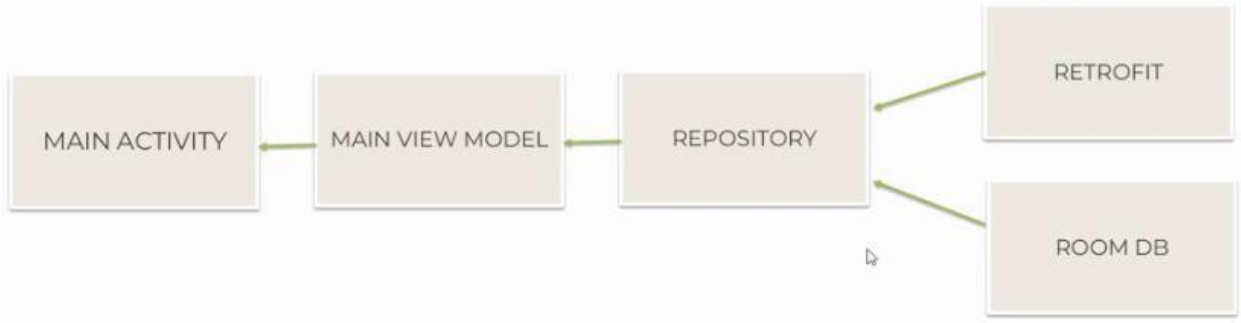
```
/*
//Flow-11
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

@Singleton
class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

class MessageService(private val retryCount:Int) : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent - Retry Count $retryCount")
   }
}
*/



/*
//Flow-12
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

class MessageService(private val retryCount:Int) : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent - Retry Count $retryCount")
   }
}
*/
```

```
/*
//Flow-13
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

@Singleton
class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

class MessageService(private val retryCount:Int) : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent - Retry Count $retryCount")
   }
}
*/




//Flow-16
interface  NotificationService  {
   fun send(to: String, from: String, body: String?)
}

class EmailService @Inject constructor() : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG,  "Email  sent")
   }
}

@ActivityScope
class MessageService(private val retryCount:Int) : NotificationService {
   override fun send(to: String, from: String, body: String?) {
      Log.d(TAG, "Message sent - Retry Count $retryCount")
   }
}
```

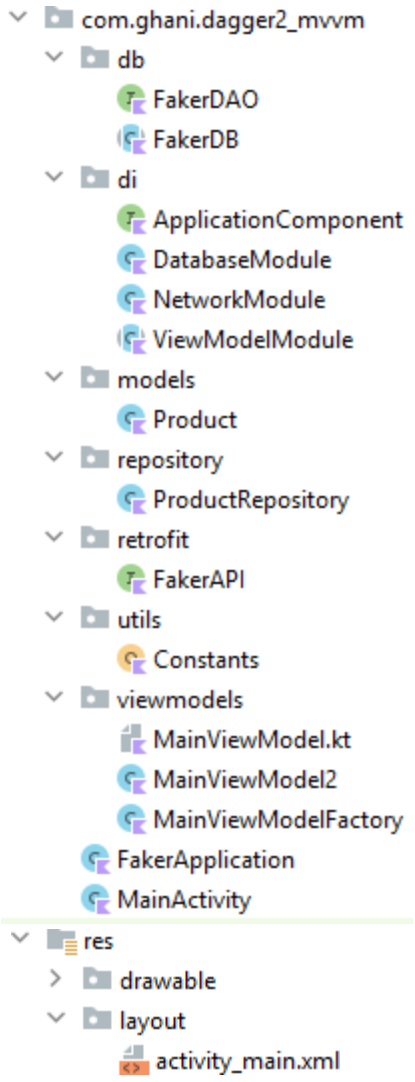
AnalyticsService.kt:

```
package com.ghani.dependencyinjection_1

import android.content.ContentValues.TAG
import android.util.Log

//Flow-16
interface AnalyticsService {
   fun trackEvent(eventName:String, eventType:String)
}

class Mixpanel:AnalyticsService{
   override fun trackEvent(eventName: String, eventType: String) {
      Log.d(TAG,"Mixpanel - $eventName - $eventType")
   }
}

class FirebaseAnalytics:AnalyticsService{
   override fun trackEvent(eventName: String, eventType: String) {
      Log.d(TAG,"FirebaseAnalytics - $eventName - $eventType")
   }
}
```

MVVM



API address link: fakestoreapi.com/products



build.gradle:

```
plugins {
    id 'kotlin-kapt'
}
android {
    kotlinOptions {
        freeCompilerArgs = ['-Xjvm-default = enable']
    }
}
dependencies {
    //Adding by me
    def dagger_version = "2.44"
    implementation "com.google.dagger:dagger:$dagger_version"
    kapt "com.google.dagger:dagger-compiler:$dagger_version"

    implementation"org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.1"
    implementation"org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.1"

    def lifecycle_version = "2.5.1"
    implementation"androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
    implementation"androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"

    def retrofit_version = "2.9.0"
    implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
    implementation  "com.squareup.retrofit2:converter-gson:$retrofit_version"

    def room_version = "2.5.1"
    implementation"androidx.room:room-runtime:$room_version"
    implementation"androidx.room:room-ktx:$room_version"
    kapt"androidx.room:room-compiler:$room_version"

    def coroutines_version = "1.5.1"
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:$coroutines_version")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:$coroutines_version")
}
```

AndroidManifest.xml:

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
    <uses-permission   android:name="android.permission.INTERNET"></uses-permission>
    <application
        android:usesCleartextTraffic="true"
        android:name=".FakerApplication"
    </application>
</manifest>
```

Product.kt:

```
package com.ghani.dagger2_mvvm.models

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity
data class Product(
    val category: String,
    val description: String,
    @PrimaryKey(autoGenerate = false)
    val id: Int,
    val image: String,
    val price: Double,
    val title: String
)
```

Constants.kt:

```
package com.ghani.dagger2_mvvm.utils

object Constants {
    const val BASE_url = "http://fakestoreapi.com"
}
```

FakerAPI.kt:

```
package com.ghani.dagger2_mvvm.retrofit

import com.ghani.dagger2_mvvm.models.Product
import retrofit2.Response
import retrofit2.http.GET

interface FakerAPI {

    @GET("products")
    suspend fun getProducts(): Response<List<Product>>

}
```

NetworkModule.kt:

```
package com.ghani.dagger2_mvvm.di

import com.ghani.dagger2_mvvm.retrofit.FakerAPI
import com.ghani.dagger2_mvvm.utils.Constants
import dagger.Module
import dagger.Provides
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import javax.inject.Singleton

@Module
class NetworkModule {
   @Singleton
   @Provides
   fun providesRetrofit():Retrofit{
      return Retrofit.Builder().baseUrl(Constants.BASE_url)
         .addConverterFactory(GsonConverterFactory.create())
         .build()
   }
   @Singleton
   @Provides
   fun providesFakerAPI(retrofit: Retrofit):FakerAPI{
      return retrofit.create(FakerAPI::class.java)
   }
}
```

FakerDAO.kt:

```
package com.ghani.dagger2_mvvm.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.ghani.dagger2_mvvm.models.Product

@Dao
interface FakerDAO {
   @Insert (onConflict = OnConflictStrategy.REPLACE)

 suspend fun addProducts(products:List<Product>)    //parameter--------take product
   @Query("Select * From Product")
   suspend fun getProducts(): List<Product>  //return type -------give product
}
```

FakerDB.kt:

```
package com.ghani.dagger2_mvvm.db

import androidx.room.Database
import androidx.room.RoomDatabase
import com.ghani.dagger2_mvvm.models.Product

@Database(entities = [Product::class],version = 1)
abstract class FakerDB:RoomDatabase() {

   abstract fun getFakerDAO():FakerDAO

}
```

DatabaseModule.kt:

```
package com.ghani.dagger2_mvvm.di

import android.content.Context
import androidx.room.Room
import com.ghani.dagger2_mvvm.db.FakerDB
import dagger.Module
import dagger.Provides
import javax.inject.Singleton

@Module
class DatabaseModule {

   @Singleton
   @Provides
   fun provideFakerDB(context: Context): FakerDB {
      return Room.databaseBuilder(context, FakerDB::class.java, "FakerDB").build()
   }
}
```

ApplicationComponent.kt:

```
package com.ghani.dagger2_mvvm.di

import android.content.Context
import androidx.lifecycle.ViewModel
import com.ghani.dagger2_mvvm.MainActivity
import dagger.BindsInstance
import dagger.Component
import javax.inject.Singleton

@Singleton
@Component(modules=[NetworkModule::class,DatabaseModule::class,ViewModelModule::class])
interface ApplicationComponent {

    fun inject (mainActivity: MainActivity)

    //fun getMap(): Map<String, ViewModel>
    fun getMap(): Map<Class<*>, ViewModel>

    @Component.Factory  //Factory --> Give something to application when this application run
    interface Factory{
        fun create (@BindsInstance context: Context): ApplicationComponent
    }
}
```

FakerApplication.kt:

```
package com.ghani.dagger2_mvvm

import android.app.Application
import com.ghani.dagger2_mvvm.di.ApplicationComponent
import com.ghani.dagger2_mvvm.di.DaggerApplicationComponent

class FakerApplication: Application() {
    lateinit var applicationComponent: ApplicationComponent

    override fun onCreate() {
        super.onCreate()
        applicationComponent = DaggerApplicationComponent.factory().create(this)
    }

}
```

ProductRepository.kt:

```
package  com.ghani.dagger2_mvvm.repository

import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import com.ghani.dagger2_mvvm.db.FakerDB
import com.ghani.dagger2_mvvm.models.Product
import com.ghani.dagger2_mvvm.retrofit.FakerAPI
import javax.inject.Inject

class ProductRepository @Inject constructor(private val fakerAPI: FakerAPI,private val fakerDB: FakerDB)
{
   private val _products = MutableLiveData<List<Product>>()

   val products: LiveData<List<Product>>
   get() = _products

   suspend fun getProducts(){
      val result = fakerAPI.getProducts()
      if (result.isSuccessful && result.body() != null){
         fakerDB.getFakerDAO().addProducts(result.body()!!)
         _products.postValue(result.body())
      }
   }
}
```

MainViewModel.kt:

```
package  com.ghani.dagger2_mvvm.viewmodels

import android.util.Log
import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.ghani.dagger2_mvvm.models.Product
import com.ghani.dagger2_mvvm.repository.ProductRepository
import kotlinx.coroutines.launch
import javax.inject.Inject

class MainViewModel  @Inject constructor(private val  repository:  ProductRepository,  private  val
randomize: Randomize):ViewModel() {

   val productsLiveData: LiveData<List<Product>>
   get() = repository.products
```

```
    init {
        viewModelScope.launch{
            repository.getProducts()
        }
    }
}
class Randomize @Inject constructor (){
    fun doAction(){
        Log.d("Cheeze  Code","Random  Action")
    }
}
```

MainViewModel2.kt:

```
package  com.ghani.dagger2_mvvm.viewmodels

import androidx.lifecycle.ViewModel
import javax.inject.Inject

class  MainViewModel2  @Inject constructor(private val  randomize:  Randomize):  ViewModel() {
    init {
        randomize.doAction()
    }
}
```

ViewModelModule.kt:

```
package  com.ghani.dagger2_mvvm.di

import androidx.lifecycle.ViewModel
import com.ghani.dagger2_mvvm.viewmodels.MainViewModel
import com.ghani.dagger2_mvvm.viewmodels.MainViewModel2
import dagger.Binds
import dagger.Module
import  dagger.multibindings.ClassKey
import  dagger.multibindings.IntoMap
import  dagger.multibindings.StringKey

@Module
abstract  class  ViewModelModule  {

    @Binds
    //@StringKey("mainViewModel")
    @ClassKey(MainViewModel::class)
    @IntoMap
    abstract  fun  mainViewModel(mainViewModel:  MainViewModel):  ViewModel
```

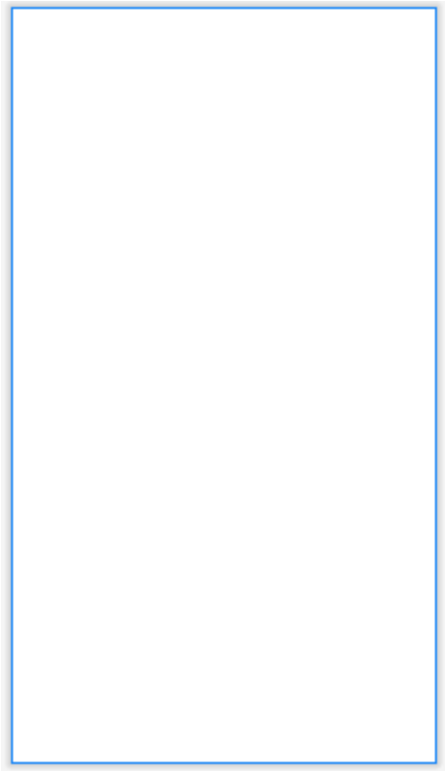
```
    @Binds
    //@StringKey(mainViewModel2)
    @ClassKey(MainViewModel2::class)
    @IntoMap
    abstract fun mainViewModel2(mainViewModel2: MainViewModel2): ViewModel
}
```

MainViewModelFactory.kt:

```
package com.ghani.dagger2_mvvm.viewmodels

import androidx.lifecycle.ViewModel
import androidx.lifecycle.ViewModelProvider
import com.ghani.dagger2_mvvm.repository.ProductRepository
import javax.inject.Inject

/*class MainViewModelFactory @Inject constructor(private val repository:
ProductRepository,private val randomize: Randomize):ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return MainViewModel(repository,randomize) as T
    }
}*/

/*class MainViewModelFactory @Inject constructor(private val mainViewModel:
MainViewModel):ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return mainViewModel as T
    }
}*/

class MainViewModelFactory @Inject constructor(private val map:
Map<Class<*>,@JvmSuppressWildcards ViewModel>):ViewModelProvider.Factory {
    override fun <T : ViewModel> create(modelClass: Class<T>): T {
        return map[modelClass] as T
    }
}
```

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/products"
      android:layout_width="0dp"
      android:layout_height="0dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />

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



MainActivity.kt:

```
package com.ghani.dagger2_mvvm

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import com.ghani.dagger2_mvvm.db.FakerDB
import com.ghani.dagger2_mvvm.viewmodels.MainViewModel
import com.ghani.dagger2_mvvm.viewmodels.MainViewModelFactory
import javax.inject.Inject

class MainActivity : AppCompatActivity() {
    lateinit var mainViewModel: MainViewModel

    @Inject
    lateinit var mainViewModelFactory: MainViewModelFactory

    @Inject
    lateinit var fakerDB1: FakerDB

    @Inject
    lateinit var fakerDB2: FakerDB

    private val products : TextView
    get() = findViewById(R.id.products)

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

        (application as FakerApplication).applicationComponent.inject(this)

        //val map = (application as FakerApplication).applicationComponent.getMap()  //show inside the map

        mainViewModel = ViewModelProvider(this,mainViewModelFactory).get(MainViewModel::class.java)

        mainViewModel.productsLiveData.observe(this, Observer{
            products.text = it.joinToString { x -> x.title + "\n\n" }
        })
    }
}
```

Hilt Dependency Injection

HILT

- Hilt is a dependency injection library for Android
- Standard way of implementing DI in Android
- Built on top of Dagger 2
- Hilt generates Dagger code for you.

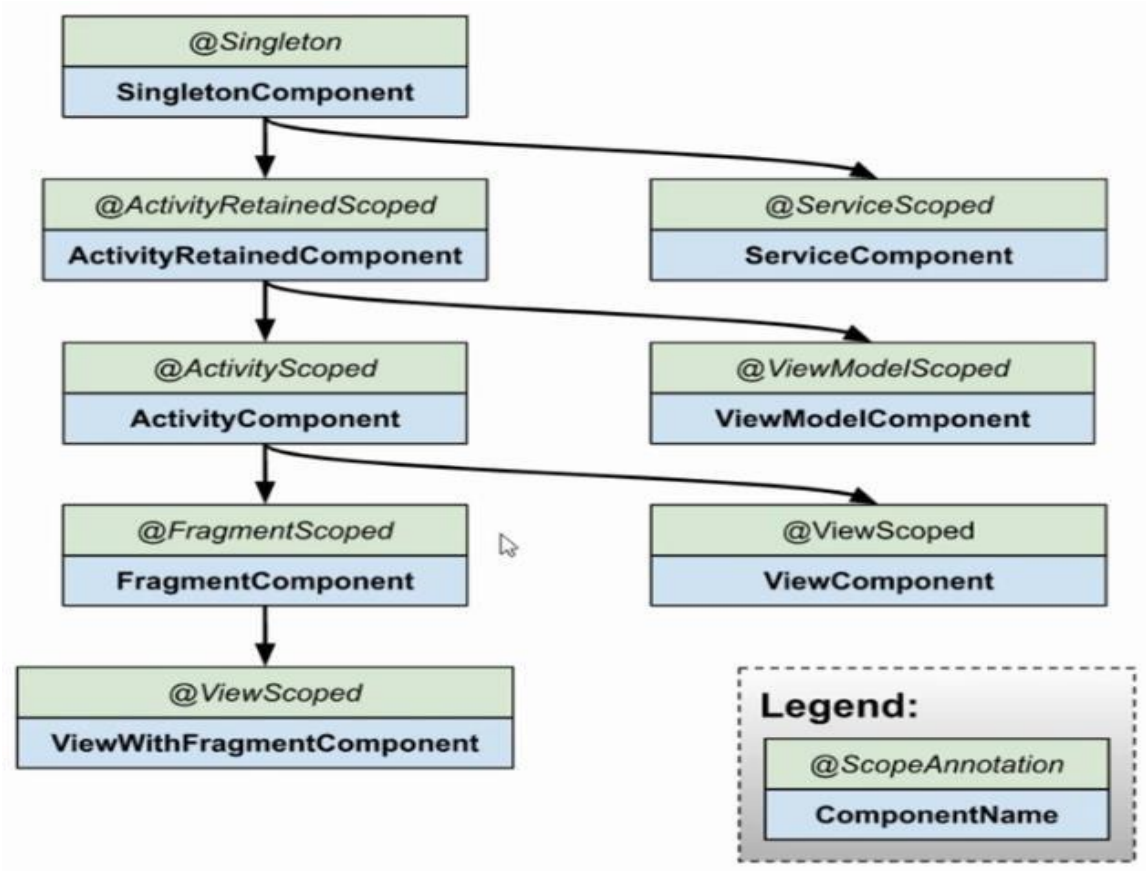
DAGGER 2

- Scopes – Singleton Scope, Activity Scope, Fragment Scope
- Components – Application Component, Activity Component etc.
- Component Dependencies – Sub Components, Dependency Attribute
- Runtime bindings for Application Context, Activity Context.
- Modules for providing dependencies.

HILT

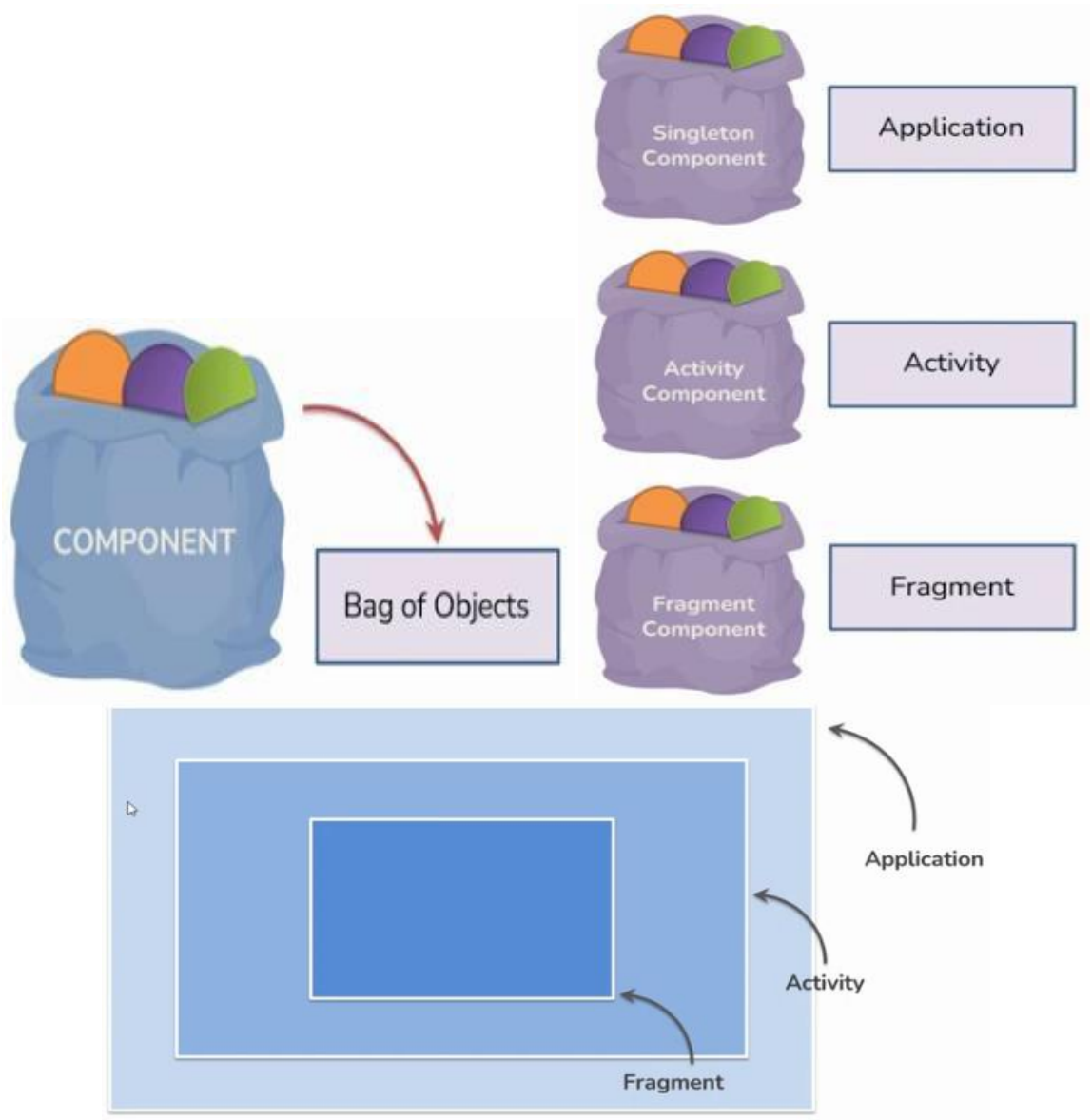
- ~~Scopes – Singleton Scope, Activity Scope, Fragment Scope~~
- ~~Components – Application Component, Activity Component etc.~~
- ~~Component Dependencies – Sub Components, Dependency Attribute~~
- ~~Runtime bindings for Application Context, Activity Context.~~
- Modules for providing dependencies.

Hilt component	Injector for
`SingletonComponent`	`Application`
`ActivityRetainedComponent`	N/A
`ViewModelComponent`	`ViewModel`
`ActivityComponent`	`Activity`
`FragmentComponent`	`Fragment`
`ViewComponent`	`View`
`ViewWithFragmentComponent`	`View` annotated with `@WithFragmentBindings`
`ServiceComponent`	`Service`



Android class	Generated component	Scope
`Application`	`SingletonComponent`	`@Singleton`
`Activity`	`ActivityRetainedComponent`	`@ActivityRetainedScoped`
`ViewModel`	`ViewModelComponent`	`@ViewModelScoped`
`Activity`	`ActivityComponent`	`@ActivityScoped`
`Fragment`	`FragmentComponent`	`@FragmentScoped`
`View`	`ViewComponent`	`@ViewScoped`
`View` annotated with `@WithFragmentBindings`	`ViewWithFragmentComponent`	`@ViewScoped`
`Service`	`ServiceComponent`	`@ServiceScoped`

Android component	Default bindings
`SingletonComponent`	`Application`
`ActivityRetainedComponent`	`Application`
`ViewModelComponent`	`SavedStateHandle`
`ActivityComponent`	`Application`, `Activity`
`FragmentComponent`	`Application`, `Activity`, `Fragment`
`ViewComponent`	`Application`, `Activity`, `View`
`ViewWithFragmentComponent`	`Application`, `Activity`, `Fragment`, `View`
`ServiceComponent`	`Application`, `Service`



Program Flow

Flow-1[UserRepository > UserApplication]

Flow-2[UserRepository > MainActivity> UserApplication]

Flow-3[UserRepository >MainFragment>activity_main.xml> MainActivity> UserApplication]

Flow-4[LoggerService>UserRepository >MainFragment>activity_main.xml> MainActivity> UserApplication]

Flow-5,6,7,8,9[UserRepository >UserModule>MainFragment>activity_main.xml> MainActivity> UserApplication]

build.gradle(Project):

```
plugins {
   //Adding by me
   id("com.google.dagger.hilt.android") version "2.44" apply false
}
```

build.gradle(Module):

```
plugins {
   //Adding by me
   id 'kotlin-kapt'
   id'com.google.dagger.hilt.android'
}

//Adding by me
compileOptions {
   sourceCompatibility = JavaVersion.VERSION_1_8
   targetCompatibility = JavaVersion.VERSION_1_8
}

//Adding by me
kapt {
   correctErrorTypes = true
}

dependencies {
   //Adding by me
   implementation("com.google.dagger:hilt-android:2.44")
   kapt("com.google.dagger:hilt-android-compiler:2.44")
}
```

AndroidManifest.xml:

```
<application

   android:name=".UserApplication"

</application>
```

FirebaseQualifier.kt:

```
package com.ghani.hilt_di

import java.lang.annotation.Documented
import java.lang.annotation.Retention
import java.lang.annotation.RetentionPolicy
import javax.inject.Qualifier

@Qualifier
@Documented
@Retention(RetentionPolicy.RUNTIME)
annotation class FirebaseQualifier()
```

LoggerService.kt:

```
package com.ghani.hilt_di

import android.util.Log
import javax.inject.Inject

//Flow-4
class LoggerService @Inject constructor() {
   fun log(message:String){
      Log.d(TAG,message)
   }
}
```

UserRepository.kt:

```
package com.ghani.hilt_di

import android.util.Log
import javax.inject.Inject

/*
//Flow-1,2,3
const val TAG = "MyCheezeCode"

class UserRepository @Inject constructor() {
   fun saveUser(email:String,password:String){
      Log.d(TAG,"User save in DB")
   }
}
*/
```

```
/*
//Flow-4
const val TAG = "MyCheezeCode"

class UserRepository @Inject constructor(val loggerService: LoggerService) {    //constructor
injection
   fun saveUser(email:String,password:String){
      loggerService.log("User save in DB")
   }
}
*/

//Flow-5,6,7,8,9
const val TAG = "MyCheezeCode"

interface UserRepository {
   fun saveUser(email: String, password: String)
}
class SQLRepository @Inject constructor():UserRepository {
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User save in DB")
   }
}
class FirebaseRepository:UserRepository {
   override fun saveUser(email:String, password:String){
      Log.d(TAG,"User save in Firebase")
   }
}
```

UserModule.kt:

```
package com.ghani.hilt_di

import dagger.Binds
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.components.ActivityComponent
import dagger.hilt.android.components.FragmentComponent
import dagger.hilt.components.SingletonComponent
import javax.inject.Named
import javax.inject.Singleton
```

```
/*
//Flow-5
@InstallIn(FragmentComponent::class)  //for Fragment level
@Module
class UserModule {

   @Provides
   fun providesUserRepository():UserRepository{
      return FirebaseRepository()
   }

}
*/


/*
//Flow-6
@InstallIn(SingletonComponent::class)  //for application level
@Module
class UserModule {

   @Provides
   fun providesUserRepository():UserRepository{
      return FirebaseRepository()
   }

}
*/


/*
//Flow-7
@InstallIn(ActivityComponent::class)
@Module
abstract class UserModule {

   @Binds
   abstract fun providesUserRepository(sqlRepository:SQLRepository):UserRepository
}
*/
```

```
/*
//Flow-8
@InstallIn(ActivityComponent::class)
@Module
class UserModule {

   @Provides
   @Named("sql")
   fun sqlRepository():UserRepository{
      return SQLRepository()
   }

   @Provides
   @Named("firebase")
   fun providesUserRepository():UserRepository{
      return FirebaseRepository()
   }

}
*/


//Flow-9
@InstallIn(ActivityComponent::class)
@Module
class UserModule {

   @Provides
   @Named("sql")
   fun sqlRepository():UserRepository{
      return SQLRepository()
   }

   @Provides
   @FirebaseQualifier
   fun providesUserRepository():UserRepository{
      return FirebaseRepository()
   }

}
```

MainFragment.kt:

```
package com.ghani.hilt_di

import android.os.Bundle
import androidx.fragment.app.Fragment
import android.view.LayoutInflater
import android.view.View
import android.view.ViewGroup
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

/*
//Flow-3,4,5
@AndroidEntryPoint
class MainFragment : Fragment() {

   @Inject
   lateinit var userRepository:UserRepository      //field injection

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

      userRepository.saveUser("test@gmail.com","1111111")

      return inflater.inflate(R.layout.fragment_main, container, false)
   }
}
*/

//Flow-6,7,8,9
class MainFragment : Fragment() {

   override fun onCreateView(
      inflater: LayoutInflater, container: ViewGroup?,
      savedInstanceState: Bundle?,
   ): View? {
      return inflater.inflate(R.layout.fragment_main, container, false)
   }

}
```

activity_main.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout 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">

  <fragment
     android:id="@+id/fragment"
     android:name="com.ghani.hilt_di.MainFragment"
     android:layout_height="match_parent"
     android:layout_width="match_parent"/>

</LinearLayout>
```

MainActivity.kt:

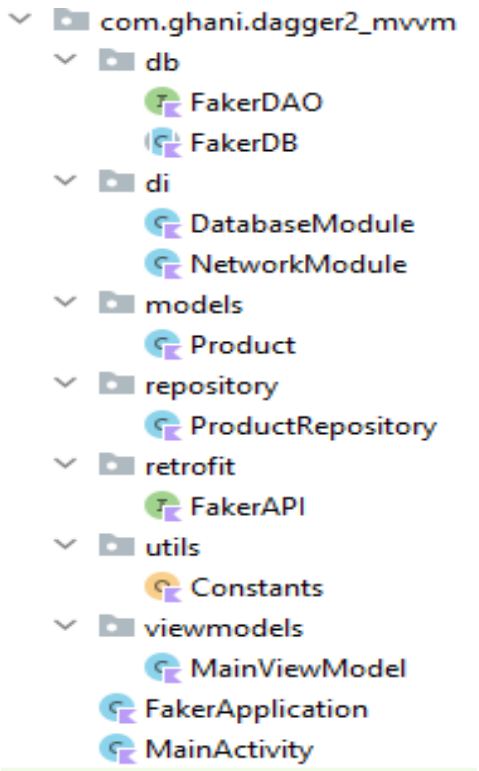
```
package com.ghani.hilt_di

import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject
import javax.inject.Named

/*
//Flow-2,6,7
@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

   @Inject
   lateinit var userRepository:UserRepository

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

      userRepository.saveUser("test@gmail.com","1111111")

   }
}
*/
```

```
/*
//Flow-3,4,5
@AndroidEntryPoint     //because it takes fragment
class MainActivity : AppCompatActivity() {
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContentView(R.layout.activity_main)
   }
}
*/
/*
//Flow-8
@AndroidEntryPoint
class  MainActivity  :  AppCompatActivity()  {

   @Inject
   @Named("firebase")
   lateinit var userRepository:UserRepository

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

      userRepository.saveUser("test@gmail.com","1111111")

   }
}
*/
//Flow-9
@AndroidEntryPoint
class  MainActivity  :  AppCompatActivity()  {

   @Inject
   @FirebaseQualifier
   lateinit var userRepository:UserRepository

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

      userRepository.saveUser("test@gmail.com","1111111")

   }
}
```

UserApplication.kt:

```
package com.ghani.hilt_di
import android.app.Application
import dagger.hilt.android.HiltAndroidApp
import javax.inject.Inject
/*
//Flow-1
@HiltAndroidApp
class UserApplication: Application() {
   @Inject
   lateinit var userRepository:UserRepository

   override fun onCreate() {
      super.onCreate()
      userRepository.saveUser("test@gmail.com","1111111")
   }
}*/

//Flow-2,3,4,5,6,7,8,9
@HiltAndroidApp
class UserApplication: Application() {
}
```

Hilt MVVM



build.gradle(Project):

```
plugins {
   //Adding by me
   id("com.google.dagger.hilt.android") version "2.44" apply false
}
```

build.gradle(Module):

```
plugins {
   //Adding by me
   id 'kotlin-kapt'
   id'com.google.dagger.hilt.android'
}

//Adding by me
compileOptions {
   sourceCompatibility = JavaVersion.VERSION_1_8
   targetCompatibility = JavaVersion.VERSION_1_8
}

//Adding by me
kapt {
   correctErrorTypes = true
}

dependencies {
   //Adding by me
   def hilt_version = "2.44"
   implementation"com.google.dagger:hilt-android:$hilt_version"
   kapt"com.google.dagger:hilt-android-compiler:$hilt_version"

   implementation"org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.1"
   implementation"org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.1"

   def lifecycle_version = "2.5.1"
   implementation"androidx.lifecycle:lifecycle-viewmodel-ktx:$lifecycle_version"
   implementation"androidx.lifecycle:lifecycle-livedata-ktx:$lifecycle_version"

   def retrofit_version = "2.9.0"
   implementation "com.squareup.retrofit2:retrofit:$retrofit_version"
   implementation "com.squareup.retrofit2:converter-gson:$retrofit_version"
```

```
    def room_version = "2.5.1"
    implementation"androidx.room:room-runtime:$room_version"
    implementation"androidx.room:room-ktx:$room_version"
    kapt"androidx.room:room-compiler:$room_version"

    def coroutines_version = "1.5.1"
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-core:$coroutines_version")
    implementation("org.jetbrains.kotlinx:kotlinx-coroutines-android:$coroutines_version")
}
```

AndroidManifest.xml:

```
<manifest>

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

    <application

        android:usesCleartextTraffic="true"
        android:name=".FakerApplication">

    </application>

</manifest>
```

Product.kt:

```
package com.ghani.dagger2_mvvm.models

import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity
data class Product(
    val category: String,
    val description: String,
    @PrimaryKey(autoGenerate = false)
    val id: Int,
    val image: String,
    val price: Double,
    val title: String
)
```

Constants.kt:

```
package com.ghani.dagger2_mvvm.utils

object Constants {
   const val BASE_url = "http://fakestoreapi.com"
}
```

FakerAPI.kt:

```
package com.ghani.dagger2_mvvm.retrofit

import com.ghani.dagger2_mvvm.models.Product
import retrofit2.Response
import retrofit2.http.GET

interface FakerAPI {

   @GET("products")
   suspend fun getProducts(): Response<List<Product>>

}
```

NetworkModule.kt:

```
package com.ghani.dagger2_mvvm.di

import com.ghani.dagger2_mvvm.retrofit.FakerAPI
import com.ghani.dagger2_mvvm.utils.Constants
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import javax.inject.Singleton

@InstallIn(SingletonComponent::class)
@Module
class NetworkModule {

   @Singleton
   @Provides
   fun providesRetrofit():Retrofit{
      return Retrofit.Builder().baseUrl(Constants.BASE_url)
```

```
        .addConverterFactory(GsonConverterFactory.create())
        .build()
    }

    @Singleton
    @Provides
    fun providesFakerAPI(retrofit: Retrofit):FakerAPI{
        return retrofit.create(FakerAPI::class.java)
    }

}
```

FakerDAO.kt:

```
package com.ghani.dagger2_mvvm.db

import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.ghani.dagger2_mvvm.models.Product

@Dao
interface FakerDAO {

    @Insert (onConflict = OnConflictStrategy.REPLACE)
    suspend fun addProducts(products:List<Product>)     //parameter ------- take product

    @Query("Select * From Product")
    suspend fun getProducts(): List<Product>   //return type-------give product
}
```

FakerDB.kt:

```
package com.ghani.dagger2_mvvm.db
import androidx.room.Database
import androidx.room.RoomDatabase
import com.ghani.dagger2_mvvm.models.Product

@Database(entities = [Product::class],version = 1)
abstract class FakerDB:RoomDatabase() {

    abstract fun getFakerDAO():FakerDAO
}
```

DatabaseModule.kt:

```
package com.ghani.dagger2_mvvm.di

import android.content.Context
import androidx.room.Room
import com.ghani.dagger2_mvvm.db.FakerDB
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@InstallIn(SingletonComponent::class)
@Module
class DatabaseModule {

    @Singleton
    @Provides
    fun provideFakerDB(@ApplicationContext context: Context): FakerDB {
        return Room.databaseBuilder(context, FakerDB::class.java, "FakerDB").build()
    }
}
```

FakerApplication.kt:

```
package com.ghani.dagger2_mvvm

import android.app.Application
import dagger.hilt.android.HiltAndroidApp

@HiltAndroidApp
class FakerApplication: Application() {
    override fun onCreate() {
        super.onCreate()
    }
}
```

ProductRepository.kt:

```
package com.ghani.dagger2_mvvm.repository
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import com.ghani.dagger2_mvvm.db.FakerDB
import com.ghani.dagger2_mvvm.models.Product
import com.ghani.dagger2_mvvm.retrofit.FakerAPI
import javax.inject.Inject

class ProductRepository @Inject constructor(private val fakerAPI: FakerAPI,private val fakerDB:
FakerDB){
   private val _products = MutableLiveData<List<Product>>()
   val products: LiveData<List<Product>>
   get() = _products

   suspend fun getProducts(){
      val result = fakerAPI.getProducts()
      if (result.isSuccessful && result.body() != null){
         fakerDB.getFakerDAO().addProducts(result.body()!!)
         _products.postValue(result.body())
      }
   }
}
```

MainViewModel.kt:

```
package com.ghani.dagger2_mvvm.viewmodels
import androidx.lifecycle.LiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.ghani.dagger2_mvvm.models.Product
import com.ghani.dagger2_mvvm.repository.ProductRepository
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.launch
import javax.inject.Inject

@HiltViewModel     //for this,No need to create factory
class MainViewModel @Inject constructor(private val repository: ProductRepository):ViewModel() {
   val productsLiveData: LiveData<List<Product>>
   get() = repository.products
   init {
      viewModelScope.launch{
         repository.getProducts()
      }
   }
}
```

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/products"
      android:layout_width="0dp"
      android:layout_height="0dp"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintLeft_toLeftOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

MainActivity.kt:

```
package com.ghani.dagger2_mvvm
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.widget.TextView
import androidx.lifecycle.Observer
import androidx.lifecycle.ViewModelProvider
import com.ghani.dagger2_mvvm.viewmodels.MainViewModel
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

   lateinit var mainViewModel: MainViewModel

   private val products : TextView
   get() = findViewById(R.id.products)

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

      mainViewModel = ViewModelProvider(this).get(MainViewModel::class.java)
      mainViewModel.productsLiveData.observe(this, Observer{
         products.text = it.joinToString { x -> x.title + "\n\n" }
      })
   }
}
```