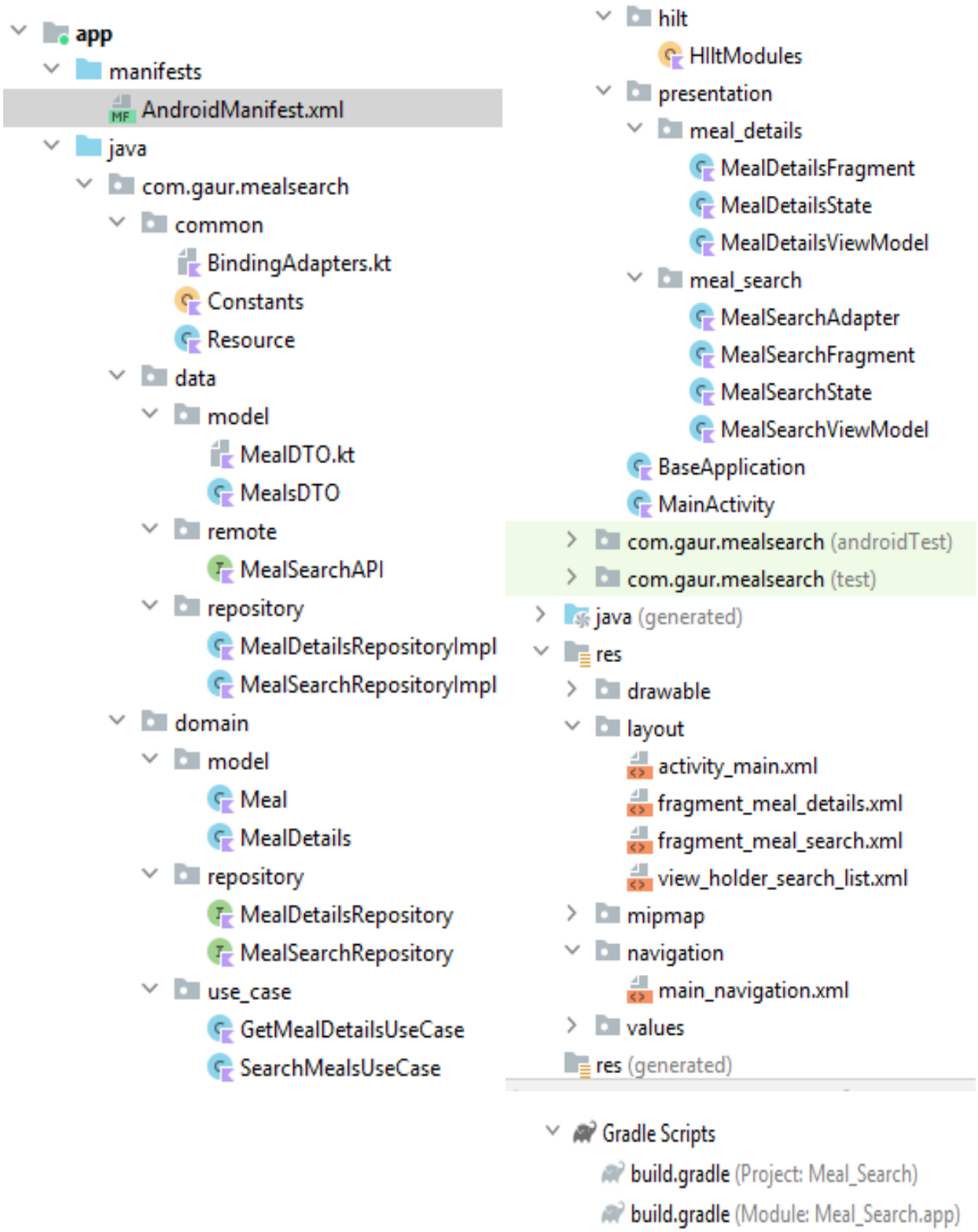
Clean Architecture



AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
   package="com.gaur.mealsearch">
   <uses-permission android:name="android.permission.INTERNET" />
   <application
      android:name=".BaseApplication"
      android:allowBackup="true"
      android:icon="@mipmap/ic_launcher"
      android:label="@string/app_name"
      android:roundIcon="@mipmap/ic_launcher_round"
      android:supportsRtl="true"
      android:theme="@style/Theme.MealSearch">
      <activity
         android:name=".MainActivity"
         android:exported="true">
         <intent-filter>
            <!--Follow1(Step1)-->
            <action android:name="android.intent.action.MAIN" />
            <category android:name="android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
   </application>
</manifest>
```

BindingAdapters.kt:

```
package com.gaur.mealsearch.common
import.....

//Follow3(Step24)
@BindingAdapter("urlToImage")
fun urlToImage(view: ImageView, s: String?) {
   val options = RequestOptions.placeholderOf(R.drawable.loading).error(R.drawable.error)
   Glide.with(view).setDefaultRequestOptions(options).load(s ?: "").into(view)
}
```

Constants.kt:

```
package com.gaur.mealsearch.common

object Constants {
   const val BASE_URL = "https://www.themealdb.com/"
}
```

Resource.kt:

```
package com.gaur.mealsearch.common

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

MealDTO.kt:

```
package com.gaur.mealsearch.data.model
import.....................................

data class MealDTO(
   val dateModified: String?,
   val idMeal: String,
   val strArea: String?,
   val strCategory: String?,
   val strCreativeCommonsConfirmed: String?,
   val strDrinkAlternate: String?,
   val strImageSource: String?,
   val strIngredient1: String?,
   val strIngredient10: String?,
   val strIngredient11: String?,
   val strIngredient12: String?,
   val strIngredient13: String?,
   val strIngredient14: String?,
   val strIngredient15: String?,
   val strIngredient16: String?,
   val strIngredient17: String?,
   val strIngredient18: String?,
   val strIngredient19: String?,
   val strIngredient2: String?,
   val strIngredient20: String?,
   val strIngredient3: String?,
   val strIngredient4: String?,
   val strIngredient5: String?,
   val strIngredient6: String?,
   val strIngredient7: String?,
   val strIngredient8: String?,
   val strIngredient9: String?,
   val strInstructions: String?,
   val strMeal: String?,
   val strMealThumb: String?,
```

```
    val strMeasure1: String?,
    val strMeasure10: String?,
    val strMeasure11: String?,
    val strMeasure12: String?,
    val strMeasure13: String?,
    val strMeasure14: String?,
    val strMeasure15: String?,
    val strMeasure16: String?,
    val strMeasure17: String?,
    val strMeasure18: String?,
    val strMeasure19: String?,
    val strMeasure2: String?,
    val strMeasure20: String?,
    val strMeasure3: String?,
    val strMeasure4: String?,
    val strMeasure5: String?,
    val strMeasure6: String?,
    val strMeasure7: String?,
    val strMeasure8: String?,
    val strMeasure9: String?,
    val strSource: String?,
    val strTags: String?,
    val strYoutube: String?
)

fun MealDTO.toDomainMeal(): Meal {
    return Meal(
        id = this.idMeal,
        name = this.strMeal ?: "",
        image = this.strMealThumb ?: ""
    )
}

fun MealDTO.toDomainMealDetails(): MealDetails {
    return MealDetails(
        name = this.strMeal ?: "",
        image = this.strMealThumb ?: "",
        instructions = this.strInstructions ?: "",
        category = this.strCategory ?: "",


        ingredient1 = this.strIngredient1 ?: "",
        ingredient2 = this.strIngredient2 ?: "",
        ingredient3 = this.strIngredient3 ?: "",
```

```
        ingredient4 = this.strIngredient4 ?: "",
        ingredient5 = this.strIngredient5 ?: "",
        ingredient6 = this.strIngredient6 ?: "",
        ingredient7 = this.strIngredient7 ?: "",
        ingredient8 = this.strIngredient8 ?: "",
        ingredient9 = this.strIngredient9 ?: "",
        ingredient10 = this.strIngredient10 ?: "",
        ingredient11 = this.strIngredient11 ?: "",
        ingredient12 = this.strIngredient12 ?: "",
        ingredient13 = this.strIngredient13 ?: "",
        ingredient14 = this.strIngredient14 ?: "",
        ingredient15 = this.strIngredient15 ?: "",
        ingredient16 = this.strIngredient16 ?: "",
        ingredient17 = this.strIngredient17 ?: "",
        ingredient18 = this.strIngredient18 ?: "",
        ingredient19 = this.strIngredient19 ?: "",
        ingredient20 = this.strIngredient20 ?: "",

        measure1 = this.strMeasure1 ?: "",
        measure2 = this.strMeasure2 ?: "",
        measure3 = this.strMeasure3 ?: "",
        measure4 = this.strMeasure4 ?: "",
        measure5 = this.strMeasure5 ?: "",
        measure6 = this.strMeasure6 ?: "",
        measure7 = this.strMeasure7 ?: "",
        measure8 = this.strMeasure8 ?: "",
        measure9 = this.strMeasure9 ?: "",
        measure10 = this.strMeasure10 ?: "",
        measure11 = this.strMeasure11 ?: "",
        measure12 = this.strMeasure12 ?: "",
        measure13 = this.strMeasure13 ?: "",
        measure14 = this.strMeasure14 ?: "",
        measure15 = this.strMeasure15 ?: "",
        measure16 = this.strMeasure16 ?: "",
        measure17 = this.strMeasure17 ?: "",
        measure18 = this.strMeasure18 ?: "",
        measure19 = this.strMeasure19 ?: "",
        measure20 = this.strMeasure20 ?: "",
        )
}
```

MealsDTO.kt:

```
package com.gaur.mealsearch.data.model

data class MealsDTO(
   val meals: List<MealDTO>?
)
```

MealSearchAPI.kt:

```
package com.gaur.mealsearch.data.remote
import .....

interface MealSearchAPI {
   //User give meals name(s) & take mealsList(MealsDTO) //Follow2,3(Step11)
   @GET("api/json/v1/1/search.php")
   suspend fun getSearchMealList(
      @Query("s") query: String
   ): MealsDTO

   //Follow4(Step12)
   //User gives mealId(i) & take mealsList(MealsDTO)
   @GET("api/json/v1/1/lookup.php")
   suspend fun getMealDetails(
      @Query("i") i: String
   ): MealsDTO
}
```

MealDetailsRepositoryImpl.kt:

```
package com.gaur.mealsearch.data.repository
import .....

class MealDetailsRepositoryImpl(private val mealSearchAPI: MealSearchAPI) :
MealDetailsRepository {
   //User gives mealId(id) & take mealsList(MealsDTO)
   //Follow4(Step10,14)
   override suspend fun getMealDetails(id: String): MealsDTO {
      //User gives mealId(id) & take mealsList(MealsDTO)
      //Follow4(Step11,13)
      return mealSearchAPI.getMealDetails(id)
   }
}
```

MealSearchRepositoryImpl.kt:

```
package com.gaur.mealsearch.data.repository
import .....................................................

class MealSearchRepositoryImpl(private val mealSearchAPI: MealSearchAPI) :
MealSearchRepository {
   //User give meals name(s) & take mealsList(MealsDTO) //Follow2,3(Step9,13)
   override suspend fun getMealSearch(s: String): MealsDTO {
      //User give meals name(s) & take mealsList(MealsDTO) //Follow2,3(Step10,12)
      return mealSearchAPI.getSearchMealList(s)
   }
}
```

Meal.kt:

```
package com.gaur.mealsearch.domain.model

data class Meal(
   val id: String,
   val name: String,
   val image: String
)
```

MealDetails.kt:

```
package com.gaur.mealsearch.domain.model

class MealDetails(
   val name: String,
   val image: String,
   val category: String,
   val instructions: String,

   val ingredient1: String,
   val ingredient2: String,
   val ingredient3: String,
   val ingredient4: String,
   val ingredient5: String,
   val ingredient6: String,
   val ingredient7: String,
   val ingredient8: String,
   val ingredient9: String,
   val ingredient10: String,
   val ingredient11: String,
   val ingredient12: String,
```

```
    val ingredient13: String,
    val ingredient14: String,
    val ingredient15: String,
    val ingredient16: String,
    val ingredient17: String,
    val ingredient18: String,
    val ingredient19: String,
    val ingredient20: String,

    val measure1: String,
    val measure2: String,
    val measure3: String,
    val measure4: String,
    val measure5: String,
    val measure6: String,
    val measure7: String,
    val measure8: String,
    val measure9: String,
    val measure10: String,
    val measure11: String,
    val measure12: String,
    val measure13: String,
    val measure14: String,
    val measure15: String,
    val measure16: String,
    val measure17: String,
    val measure18: String,
    val measure19: String,
    val measure20: String,
) {
}
```

MealDetailsRepository.kt:

```
package com.gaur.mealsearch.domain.repository
import com.gaur.mealsearch.data.model.MealsDTO

interface MealDetailsRepository {
    //Follow4(Step9,15)
    //User gives mealId(id) & take mealsList(MealsDTO)
    suspend fun getMealDetails(id:String):MealsDTO
}
```

MealSearchRepository.kt:

```
package com.gaur.mealsearch.domain.repository
import .....

interface MealSearchRepository {
   //User give meals name(s) & take mealsList(MealsDTO) //Follow2,3(Step8,14)
   suspend fun getMealSearch(s:String): MealsDTO
}
```

GetMealDetailsUseCase.kt:

```
package com.gaur.mealsearch.domain.use_case
import .....

class GetMealDetailsUseCase @Inject constructor(private val repository:
MealDetailsRepository) {

   //Follow4(Step7)
   operator fun invoke(id: String): Flow<Resource<List<MealDetails>>> = flow {
      try {
         emit(Resource.Loading())
         //User gives mealId(id) & take mealsList(data)
         //Follow4(Step8)
         val data = repository.getMealDetails(id)
         val domainData =
         if (!data.meals.isNullOrEmpty()) data.meals.map { it -> it.toDomainMealDetails() }
         else emptyList()
         //User takes mealsList(data) //Follow4(Step16)
         emit(Resource.Success(data = domainData))
      } catch (e: HttpException) {
         emit(Resource.Error(message = e.localizedMessage ?: "An Unknown error occurred"))
      } catch (e: IOException) {
         emit(Resource.Error(message = e.localizedMessage ?: "Check Connectivity"))
      } catch (e: Exception) {
      }
   }
}
```

SearchMealsUseCase.kt:

```
package com.gaur.mealsearch.domain.use_case
import .....

class SearchMealsUseCase @Inject constructor(private val repository: MealSearchRepository) {
   //User give meals name(q) //Follow2,3(Step6)
   operator fun invoke(q: String): Flow<Resource<List<Meal>>> = flow {
```

```
        try {
            emit(Resource.Loading())
            //User give meals name(q) //Follow2,3(Step7)
            val rawData = repository.getMealSearch(q)
            val domainData =
                if (rawData.meals != null) rawData.meals.map { it -> it.toDomainMeal() }
              else emptyList()
            //User take mealsList(data) //Follow2,3(Step15)
            emit(Resource.Success(data = domainData))
        } catch (e: HttpException) {
            emit(Resource.Error(message = e.localizedMessage ?: "An Unknown error occurred"))
        } catch (e: IOException) {
            emit(Resource.Error(message = e.localizedMessage ?: "Check Connectivity"))
        } catch (e: Exception) {
        }
    }
}
```

HIltModules.kt:

```
package com.gaur.mealsearch.hilt
import .....

@InstallIn(SingletonComponent::class)
@Module
object HIltModules {

    @Provides
    @Singleton
    fun provideMealSearchAPI(): MealSearchAPI {
        return Retrofit.Builder().baseUrl(Constants.BASE_URL)
            .addConverterFactory(GsonConverterFactory.create()).build()
            .create(MealSearchAPI::class.java)
    }

    @Provides
    fun provideMealSearchRepository(mealSearchAPI: MealSearchAPI): MealSearchRepository {
        return MealSearchRepositoryImpl(mealSearchAPI)
    }

    @Provides
    fun provideMealDetails(searchMealSearchAPI: MealSearchAPI): MealDetailsRepository {
        return MealDetailsRepositoryImpl(searchMealSearchAPI)
    }
}
```

MealDetailsFragment.kt:

```
package com.gaur.mealsearch.presentation.meal_details
import .....

@AndroidEntryPoint
class MealDetailsFragment : Fragment() {
   private var _binding: FragmentMealDetailsBinding? = null
   val binding: FragmentMealDetailsBinding
      get() = _binding!!

   private val viewModel: MealDetailsViewModel by viewModels()
   private val args: MealDetailsFragmentArgs by navArgs()

   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
   }
   override fun onCreateView(
      inflater: LayoutInflater, container: ViewGroup?,
      savedInstanceState: Bundle?
   ): View? {
      _binding = FragmentMealDetailsBinding.inflate(inflater, container, false)
      return _binding?.root
   }
   override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
      args.mealId?.let {
         //User gives mealId(it) //Follow4(Step4)
         viewModel.getMealDetails(it)
      }
      lifecycle.coroutineScope.launchWhenCreated {
         //User takes mealsList(mealDetails)
         //Follow4(Step19)
         viewModel.mealDetails.collect {
            if (it.isLoading) {
            }
            if (it.error.isNotBlank()) {
               Toast.makeText(requireContext(),it.error,Toast.LENGTH_SHORT).show()
            }
            it.data?.let {
               //User set mealsList(it)
               //Follow4(Step20)
               binding.mealDetails = it
            }
         }
      }
```

```
        //Follow5(Step1F)
        binding.detailsBackArrow.setOnClickListener {
            findNavController().popBackStack()
        }
    }
}
```

MealDetailsState.kt:

```
package com.gaur.mealsearch.presentation.meal_details
import com.gaur.mealsearch.domain.model.MealDetails

data class MealDetailsState(
    val isLoading: Boolean = false,
    val data: MealDetails? = null,
    val error: String = ""
) {
}
```

MealDetailsViewModel.kt:

```
package com.gaur.mealsearch.presentation.meal_details
import .....

@HiltViewModel
class MealDetailsViewModel @Inject constructor(private val mealDetailsUseCase:
GetMealDetailsUseCase) :
    ViewModel() {
    private val _mealDetails = MutableStateFlow<MealDetailsState>(MealDetailsState())
    //User takes mealsList(mealDetails)
    //Follow4(Step18)
    val mealDetails: StateFlow<MealDetailsState> = _mealDetails

    //Follow4(Step5)
    fun getMealDetails(id: String) {
        //User gives mealId(id)
        //Follow4(Step6)
        mealDetailsUseCase(id).onEach {
            when (it) {
                is Resource.Loading -> {
                    _mealDetails.value = MealDetailsState(isLoading = true)
                }
                is Resource.Error -> {
                    _mealDetails.value = MealDetailsState(error = it.message ?: "")
                }
```

```
            is Resource.Success -> {
                //User takes mealsList(_mealDetails)
                // Follow4(Step17)
                _mealDetails.value = MealDetailsState(data = it.data?.get(0))
            }
        }
    }.launchIn(viewModelScope)
  }
}
```

MealSearchAdapter.kt:

```
package com.gaur.mealsearch.presentation.meal_search
import .....

class MealSearchAdapter : RecyclerView.Adapter<MealSearchAdapter.MyViewHolder>() {

  private var listener :((Meal)->Unit)?=null
  // ((Meal)->Unit) is a function that takes Meal & return Unit(Void).
  //var listener  initialize by null.

  var list = mutableListOf<Meal>()

  //Follow3(Step20)
  fun setContentList(list: MutableList<Meal>) {
    this.list = list
    notifyDataSetChanged()
  }

  /* notifyDataSetChanged() tells the ListView that the data has been modified;
  and to show the new data,
  the ListView must be redrawn. */


  class MyViewHolder(val viewHolder: ViewHolderSearchListBinding) :
    RecyclerView.ViewHolder(viewHolder.root)

  override fun onCreateViewHolder(
    parent: ViewGroup,
    viewType: Int
  ): MealSearchAdapter.MyViewHolder {
    val binding =
      ViewHolderSearchListBinding.inflate(LayoutInflater.from(parent.context), parent, false)
    return MyViewHolder(binding)
  }
```

```
    //When we call fun itemClickListener() from any scope/fragment, then l initialize listener.
    fun itemClickListener(l:(Meal)->Unit){
        listener= l
    }

    override fun onBindViewHolder(holder: MealSearchAdapter.MyViewHolder, position: Int) {
        //Follow3(Step21)
        holder.viewHolder.meal = this.list[position]

       /* When someone click any of meal from this list,
          listener?.let{} return that particular position's meal
          to searchAdapter.itemClickListener{} in class MealSearchFragment.*/
        //Follow4(Step1)
        holder.viewHolder.root.setOnClickListener {
            listener?.let {
                it(this.list[position])
            }
        }
    }

    override fun getItemCount(): Int {
        return this.list.size
    }
}
```

MealSearchFragment.kt:

```
package com.gaur.mealsearch.presentation.meal_search
import .....................................................

@AndroidEntryPoint
class MealSearchFragment : Fragment() {

    private val searchAdapter = MealSearchAdapter()

    private val viewModel: MealSearchViewModel by viewModels()

    private var _binding: FragmentMealSearchBinding? = null
    val binding: FragmentMealSearchBinding
        get() = _binding!!

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

```
override fun onCreateView(
    inflater: LayoutInflater, container: ViewGroup?,
    savedInstanceState: Bundle?
): View? {
    _binding = FragmentMealSearchBinding.inflate(inflater, container, false)
    return _binding?.root
}
override fun onViewCreated(view: View, savedInstanceState: Bundle?) {
    binding.mealSearchRecycler.apply {
        adapter = searchAdapter
    }
    binding.mealSearchView.setOnQueryTextListener(object :
   SearchView.OnQueryTextListener {
        //User give meals name(s) //Follow2,3(Step2)
        override fun onQueryTextSubmit(s: String?): Boolean {
            s?.let {
                //User give meals name(it) //Follow2,3(Step3)
                viewModel.getSearchMeals(it)
            }
            return false
        }
        override fun onQueryTextChange(p0: String?): Boolean {
            return false
        }
    })

    lifecycle.coroutineScope.launchWhenCreated {
        //User take mealsList(mealSearchList) //Follow2,3(Step18)
        viewModel.mealSearchList.collect {
            if (it.isLoading) {
                binding.nothingFound.visibility = View.GONE
                binding.progressMealSearch.visibility = View.VISIBLE
            }
            if (it.error.isNotBlank()) {
                binding.nothingFound.visibility = View.GONE
                binding.progressMealSearch.visibility = View.GONE
                Toast.makeText(requireContext(), it.error, Toast.LENGTH_SHORT).show()
            }

            it.data?.let {
                if (it.isEmpty()) {
                    //Follow2(Step19)
                    binding.nothingFound.visibility = View.VISIBLE
                }
```

```
                binding.progressMealSearch.visibility = View.GONE
                //Set mealsList in recyclerView's adapter.
                //Follow3(Step19)
                searchAdapter.setContentList(it.toMutableList())
            }
        }
    }
    /*It takes meal from holder.viewHolder.root.setOnClickListener{}
    in class MealSearchAdapter & navigate by
    it.id/meal.id to MealDetailsFragment.*/
    //Follow4(Step2)
    searchAdapter.itemClickListener {
        findNavController().navigate(
            MealSearchFragmentDirections.actionMealSearchFragmentToMealDetailsFragment(
                it.id
            )
        )
    }
  }
}
```

MealSearchState.kt:

```
package com.gaur.mealsearch.presentation.meal_search
import com.gaur.mealsearch.domain.model.Meal

data class MealSearchState(
  val isLoading: Boolean = false,
  val data: List<Meal>? = null,
  val error: String = ""
)
```

MealSearchViewModel.kt:

```
package com.gaur.mealsearch.presentation.meal_search
import .....

@HiltViewModel
class MealSearchViewModel @Inject constructor(private val mealSearchMealsUseCase:
SearchMealsUseCase): ViewModel(){

  private val _mealSearchList = MutableStateFlow<MealSearchState>(MealSearchState())
  //User take mealsList(mealSearchList) //Follow2,3(Step17)
  val mealSearchList: StateFlow<MealSearchState> = _mealSearchList
```

```
  //User give meals name(s) //Follow2,3(Step4)
  fun getSearchMeals(s: String) {
    //User give meals name(s) //Follow2,3(Step5)
    mealSearchMealsUseCase(s).onEach {
      when (it) {
        is Resource.Loading -> {
          _mealSearchList.value = MealSearchState(isLoading = true)
        }
        is Resource.Success -> {
          //User take mealsList(data) //Follow2,3(Step16)
          _mealSearchList.value = MealSearchState(data = it.data)
        }
        is Resource.Error -> {
          _mealSearchList.value = MealSearchState(error = it.message ?: "")
        }
      }
    }.launchIn(viewModelScope)
  }
}
```

BaseApplication.kt:

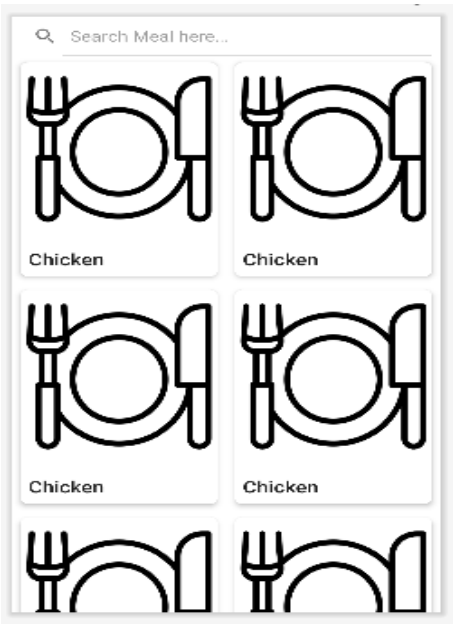
```
package com.gaur.mealsearch
import .....

@HiltAndroidApp
class BaseApplication : Application() {
}
```

MainActivity.kt:

```
package com.gaur.mealsearch
import.....

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {
  override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    //Follow1(Step2)
    setContentView(R.layout.activity_main)
  }
}
```

activity_main.xml:

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

   <fragment
      android:id="@+id/fragment_container"
      android:name="androidx.navigation.fragment.NavHostFragment"
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      app:defaultNavHost="true"
      app:navGraph="@navigation/main_navigation" /> <!--Follow1(Step3)-->

</RelativeLayout>
```



fragment_meal_details.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>
      <import type="android.view.View" />
      <!--Follow4(Step21F)-->
      <variable
         name="mealDetails"
         type="com.gaur.mealsearch.domain.model.MealDetails" />
   </data>

   <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:isScrollContainer="true"
      android:orientation="vertical"
      android:scrollbars="vertical"
      tools:context=".presentation.meal_details.MealDetailsFragment">
      <RelativeLayout
         android:layout_width="match_parent"
         android:layout_height="wrap_content">
         <ImageView
            android:id="@+id/details_back_arrow"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginStart="8dp"
            android:layout_centerVertical="true"
            app:tint="@color/black"
            android:layout_marginEnd="8dp"
            android:src="@drawable/ic_baseline_arrow_back_24" />
         <TextView
            style="@style/TextAppearance.AppCompat.Title"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:layout_toEndOf="@+id/details_back_arrow"
            android:gravity="center_vertical"
            android:minHeight="55dp"
            android:text="@{mealDetails.name}"
            android:textColor="@color/black"
            tools:text="Chicken Handi" />
      </RelativeLayout>
```

```
<androidx.core.widget.NestedScrollView
   android:layout_width="match_parent"
   android:layout_height="wrap_content">

   <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical"
      android:paddingStart="12dp"
      android:paddingEnd="12dp"
      android:paddingBottom="12dp">
      <ImageView
         android:layout_width="match_parent"
         android:layout_height="320dp"
         android:layout_marginTop="12dp"
         android:layout_marginBottom="32dp"
         android:src="@drawable/loading"
         app:urlToImage="@{mealDetails.image}" />
      <TextView
         style="@style/TextAppearance.MaterialComponents.Body1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:text="Instructions : "
         android:textColor="@color/black"
         android:textSize="18sp"
         android:textStyle="bold" />
      <TextView
         style="@style/TextAppearance.MaterialComponents.Body2"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginTop="8dp"
         android:layout_marginBottom="12dp"
         android:text="@{mealDetails.instructions}"
         tools:text="This is my instructions" />
      <TextView
         style="@style/TextAppearance.MaterialComponents.Body1"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginTop="12dp"
         android:text="Required Items : "
         android:textColor="@color/black"
         android:textSize="18sp"
         android:textStyle="bold" />

      <androidx.appcompat.widget.LinearLayoutCompat
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:layout_marginTop="8dp"
```

```
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient1.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient1}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure1}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient2.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient2}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure2}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
```

```
android:visibility="@{mealDetails.ingredient3.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient3}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.ingredient3}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient4.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient4}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure4}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient5.length()==0?View.GONE:View.VISIBLE}">
    <TextView
```

```
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient5}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure5}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient6.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient6}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure6}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient7.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
```

```
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient7}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure7}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient8.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient8}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure8}"
        tools:text="2 tea spoon" />
 </androidx.appcompat.widget.LinearLayoutCompat>

 <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient9.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
```

```
        android:text="@{mealDetails.ingredient9}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure9}"
        tools:text="2 tea spoon" />
  </androidx.appcompat.widget.LinearLayoutCompat>

  <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient10.length()==0?View.GONE:View.VISIBLE}">
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient10}"
        tools:text="Corriander" />
    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure10}"
        tools:text="2 tea spoon" />
  </androidx.appcompat.widget.LinearLayoutCompat>

  <androidx.appcompat.widget.LinearLayoutCompat
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:padding="4dp"
android:visibility="@{mealDetails.ingredient11.length()==0?View.GONE:View.VISIBLE}">

    <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient11}"
```

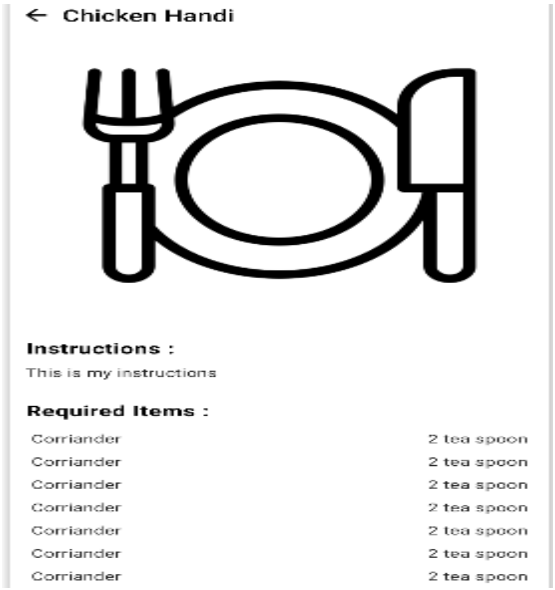
```
            tools:text="Corriander" />
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure11}"
            tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="4dp"
android:visibility="@{mealDetails.ingredient12.length()==0?View.GONE:View.VISIBLE}">

        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@{mealDetails.ingredient12}"
            tools:text="Corriander" />
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure12}"
            tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="4dp"
android:visibility="@{mealDetails.ingredient13.length()==0?View.GONE:View.VISIBLE}">
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@{mealDetails.ingredient13}"
            tools:text="Corriander" />
```

```
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure13}"
            tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="4dp"
android:visibility="@{mealDetails.ingredient14.length()==0?View.GONE:View.VISIBLE}">
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@{mealDetails.ingredient14}"
            tools:text="Corriander" />
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure14}"
            tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:orientation="horizontal"
        android:padding="4dp"
android:visibility="@{mealDetails.ingredient15.length()==0?View.GONE:View.VISIBLE}">
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@{mealDetails.ingredient15}"
            tools:text="Corriander" />
        <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure15}"
        tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:padding="4dp"
android:visibility="@{mealDetails.ingredient16.length()==0?View.GONE:View.VISIBLE}">
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient16}"
        tools:text="Corriander" />
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure16}"
        tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:padding="4dp"
android:visibility="@{mealDetails.ingredient17.length()==0?View.GONE:View.VISIBLE}">
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient17}"
        tools:text="Corriander" />
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

```
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure17}"
        tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:padding="4dp"
android:visibility="@{mealDetails.ingredient18.length()==0?View.GONE:View.VISIBLE}">


      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient18}"
        tools:text="Corriander" />
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:gravity="end"
        android:text="@{mealDetails.measure18}"
        tools:text="2 tea spoon" />
    </androidx.appcompat.widget.LinearLayoutCompat>

    <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="horizontal"
      android:padding="4dp"
android:visibility="@{mealDetails.ingredient19.length()==0?View.GONE:View.VISIBLE}">
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_weight="1"
        android:text="@{mealDetails.ingredient19}"
        tools:text="Corriander" />
      <TextView
        style="@style/TextAppearance.MaterialComponents.Body2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```

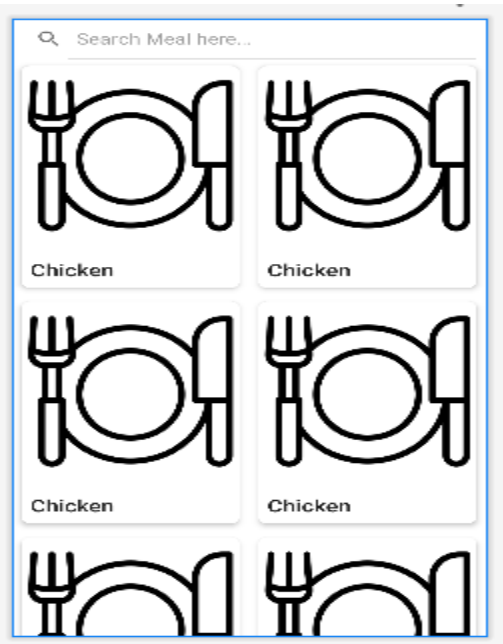
```
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure19}"
            tools:text="2 tea spoon" />
        </androidx.appcompat.widget.LinearLayoutCompat>
        <androidx.appcompat.widget.LinearLayoutCompat
          android:layout_width="match_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal"
          android:padding="4dp"
      android:visibility="@{mealDetails.ingredient20.length()==0?View.GONE:View.VISIBLE}">
          <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:text="@{mealDetails.ingredient20}"
            tools:text="Corriander" />
          <TextView
            style="@style/TextAppearance.MaterialComponents.Body2"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_weight="1"
            android:gravity="end"
            android:text="@{mealDetails.measure20}"
            tools:text="2 tea spoon" />
        </androidx.appcompat.widget.LinearLayoutCompat>
      </androidx.appcompat.widget.LinearLayoutCompat>
    </androidx.core.widget.NestedScrollView>
  </androidx.appcompat.widget.LinearLayoutCompat>
</layout>
```



fragment_meal_search.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>
   </data>

   <RelativeLayout
      android:layout_width="match_parent"
      android:layout_height="match_parent"
      android:orientation="vertical"
      tools:context=".presentation.meal_search.MealSearchFragment">

      <androidx.appcompat.widget.LinearLayoutCompat
         android:layout_width="match_parent"
         android:layout_height="wrap_content"
         android:orientation="vertical">

         <SearchView
            android:id="@+id/meal_search_view"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:iconifiedByDefault="false"
            android:queryHint="Search Meal here..." />
       <!--Follow1(Step5F)--> <!--Follow2,3(Step1)-->

         <androidx.recyclerview.widget.RecyclerView
            android:id="@+id/meal_search_recycler"
            android:layout_width="match_parent"
            android:layout_height="match_parent"
            app:layoutManager="androidx.recyclerview.widget.GridLayoutManager"
            app:spanCount="2"
            tools:listitem="@layout/view_holder_search_list" />

      </androidx.appcompat.widget.LinearLayoutCompat>

      <ProgressBar
         android:id="@+id/progress_meal_search"
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_centerInParent="true"
         android:visibility="gone" />
```

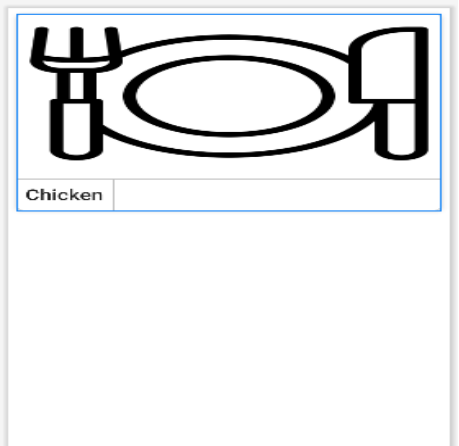
```
    <TextView
        android:id="@+id/nothing_found"
        style="@style/TextAppearance.MaterialComponents.Body1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_centerInParent="true"
        android:text="Nothing Found"
        android:textColor="@color/black"
        android:visibility="gone"/> <!--Follow2(Step20F)-->
  </RelativeLayout>
</layout>
```



view_holder_search_list.xml:

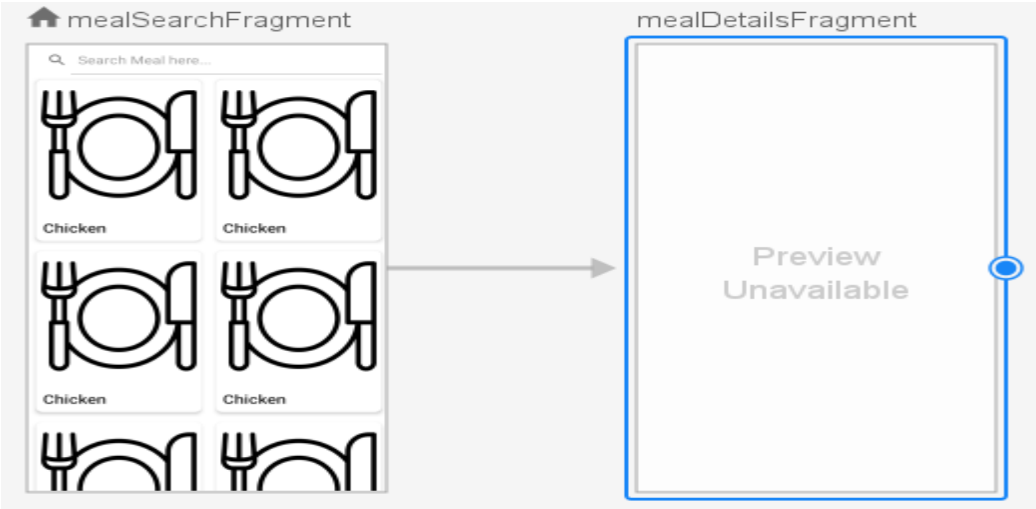
```
<?xml version="1.0" encoding="utf-8"?>
<layout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  xmlns:tools="http://schemas.android.com/tools">

  <data>
    <!--Follow3(Step22)-->
    <variable
      name="meal"
      type="com.gaur.mealsearch.domain.model.Meal" />
  </data>
```

```
  <com.google.android.material.card.MaterialCardView
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_margin="8dp"
    android:orientation="vertical"
    app:cardCornerRadius="8dp"
    app:cardElevation="4dp">
    <androidx.appcompat.widget.LinearLayoutCompat
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:orientation="vertical">
      <ImageView
        android:id="@+id/view_holder_image"
        android:layout_width="match_parent"
        android:layout_height="220dp"
        android:scaleType="fitXY"
        android:src="@color/black"
        app:urlToImage="@{meal.image}"
        tools:src="@drawable/loading" />
         <!--Follow3(Step23)-->
      <TextView
        android:id="@+id/view_holder_item_name"
        style="@style/TextAppearance.AppCompat.Title"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:padding="8dp"
        android:text="@{meal.name}"
        tools:text="Chicken" />
        <!--Follow3(Step25F)-->
    </androidx.appcompat.widget.LinearLayoutCompat>
  </com.google.android.material.card.MaterialCardView>
</layout>
```



main_navigation.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<navigation xmlns:android="http://schemas.android.com/apk/res/android"
   xmlns:app="http://schemas.android.com/apk/res-auto"
   xmlns:tools="http://schemas.android.com/tools"
   android:id="@+id/main_navigation"
   app:startDestination="@id/mealSearchFragment">

   <fragment
      android:id="@+id/mealSearchFragment"
      android:name="com.gaur.mealsearch.presentation.meal_search.MealSearchFragment"
      android:label="fragment_meal_search"
      tools:layout="@layout/fragment_meal_search"> <!--Follow1(Step4)-->
      <!--Follow4(Step3)-->
      <action
         android:id="@+id/action_mealSearchFragment_to_mealDetailsFragment"
         app:destination="@id/mealDetailsFragment" />
   </fragment>
   <fragment
      android:id="@+id/mealDetailsFragment"
      android:name="com.gaur.mealsearch.presentation.meal_details.MealDetailsFragment"
      android:label="MealDetailsFragment">
      <argument
         android:name="meal_id"
         android:defaultValue="@null"
         app:argType="string"
         app:nullable="true" />
   </fragment>
</navigation>
```



build.gradle(Project):

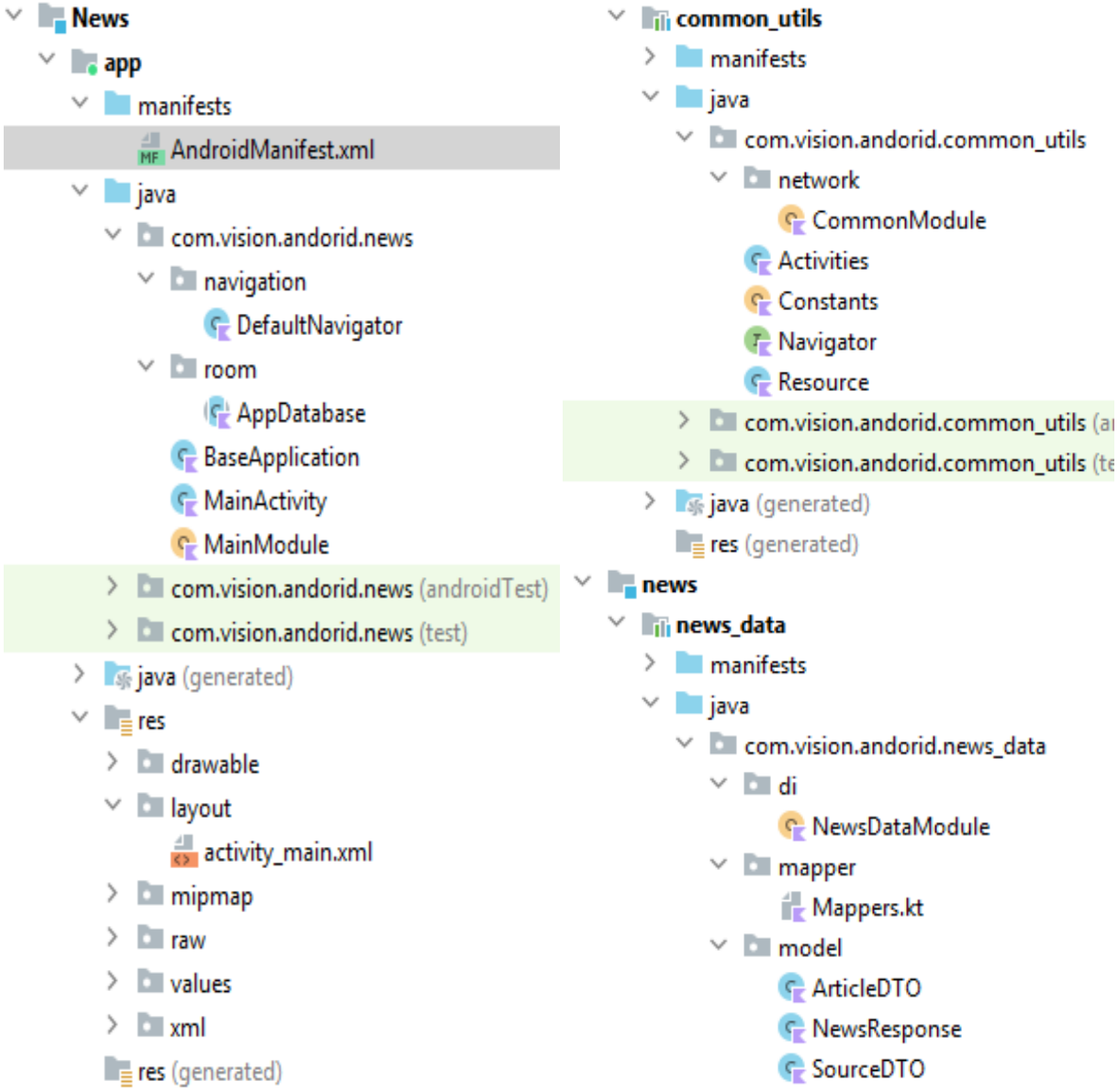
```
// Top-level build file where you can add configuration options common to all sub-
projects/modules.
buildscript {
   repositories {
      google()
      mavenCentral()
   }
   dependencies {
      classpath "com.android.tools.build:gradle:7.0.2"
      classpath "org.jetbrains.kotlin:kotlin-gradle-plugin:1.5.31"
      classpath "androidx.navigation:navigation-safe-args-gradle-plugin:2.4.0-alpha09"
      classpath "com.google.dagger:hilt-android-gradle-plugin:2.38.1"
  // NOTE: Do not place your application dependencies here; they belong in the individual module
build.gradle files
   }
}
task clean(type: Delete) {
   delete rootProject.buildDir
}
```

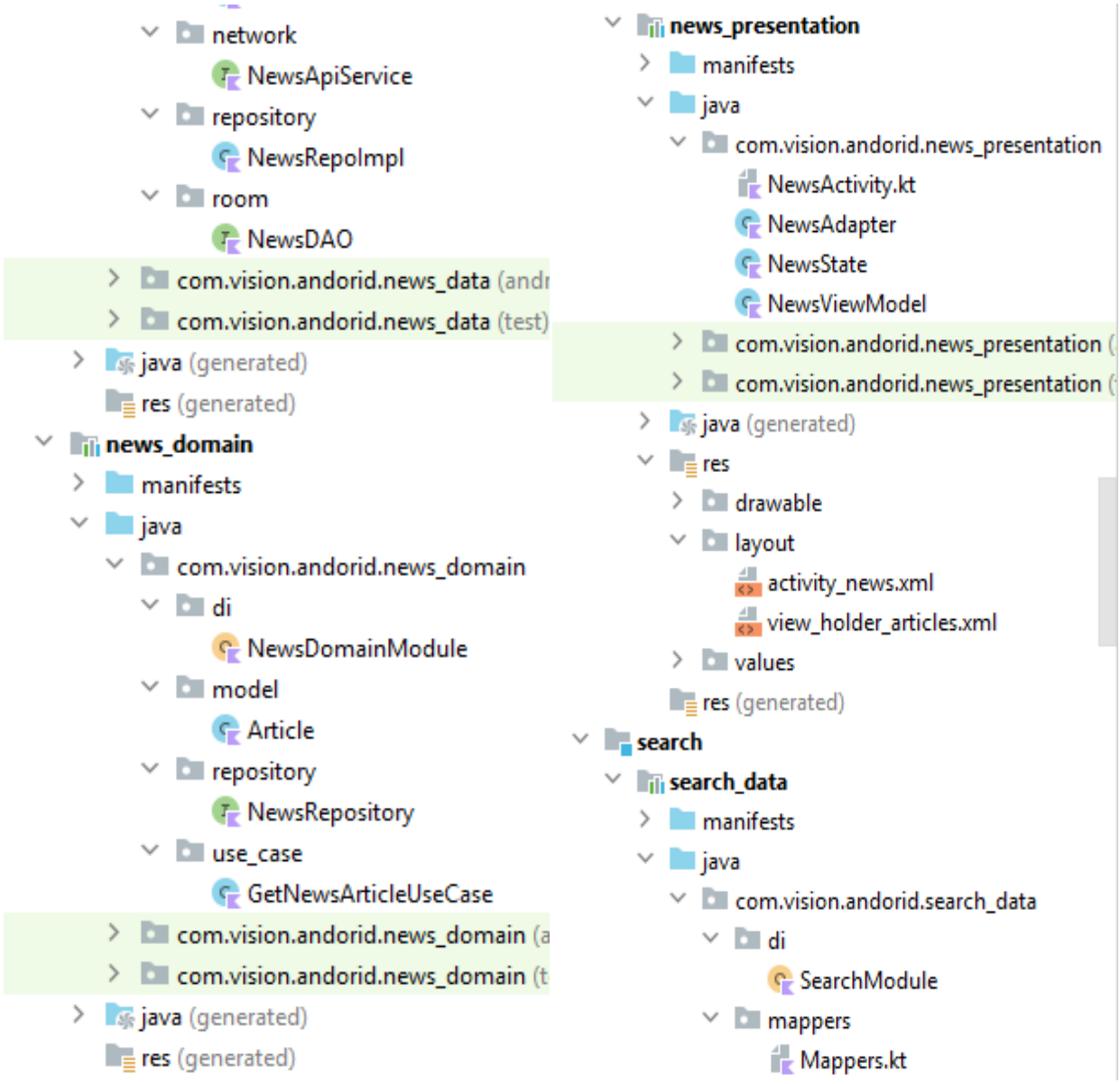
build.gradle(Module):

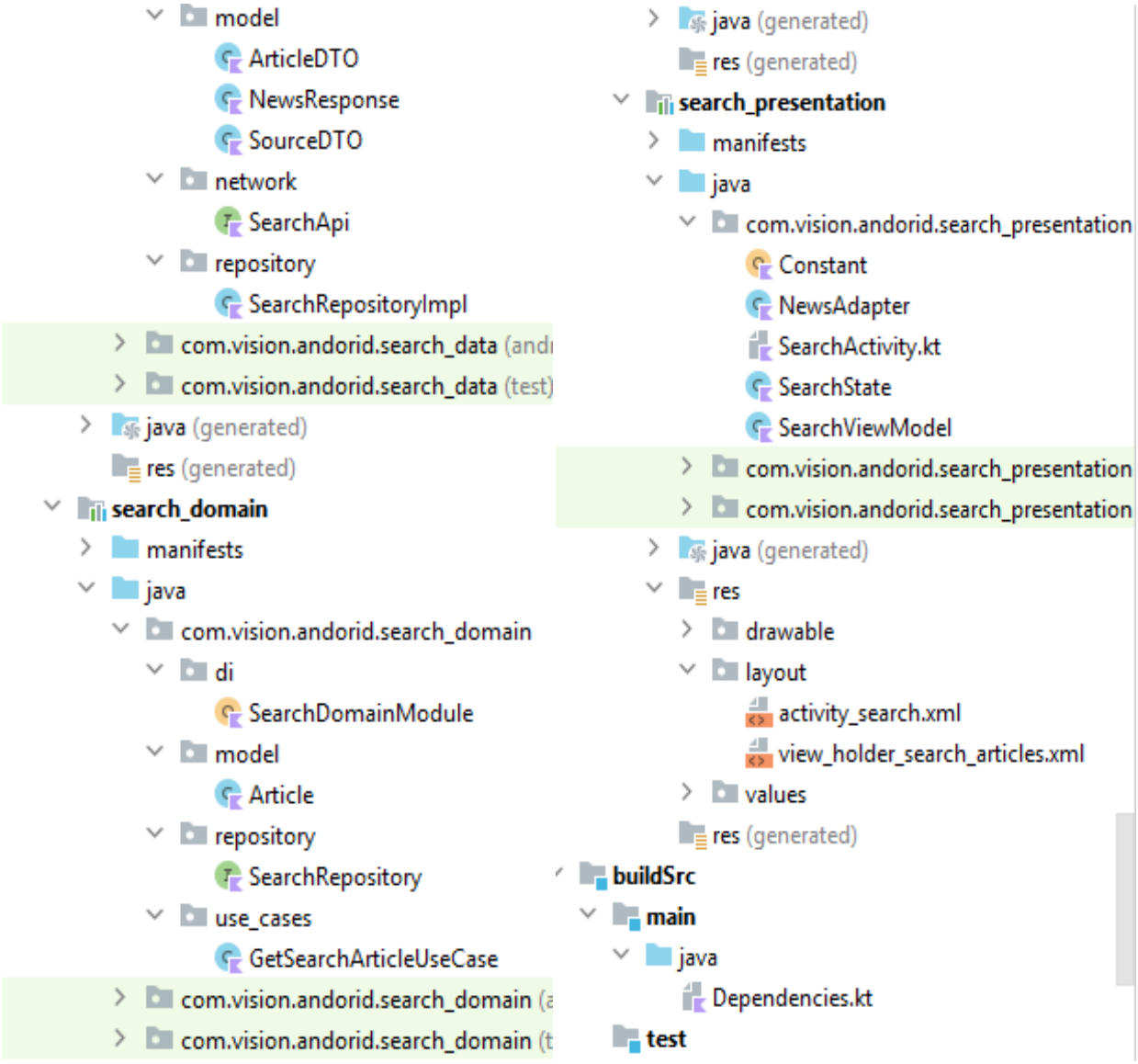
```
plugins {
   id 'com.android.application'
   id 'kotlin-android'
   id 'kotlin-kapt'
   id 'androidx.navigation.safeargs.kotlin'
   id 'dagger.hilt.android.plugin'
   id 'kotlin-parcelize'
}
android {
   compileSdk 31
   defaultConfig {
      applicationId "com.gaur.mealsearch"
      minSdk 23
      targetSdk 31
      versionCode 1
      versionName "1.0"
      testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
   }
   buildTypes {
      release {
         minifyEnabled false
         proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
      }
```

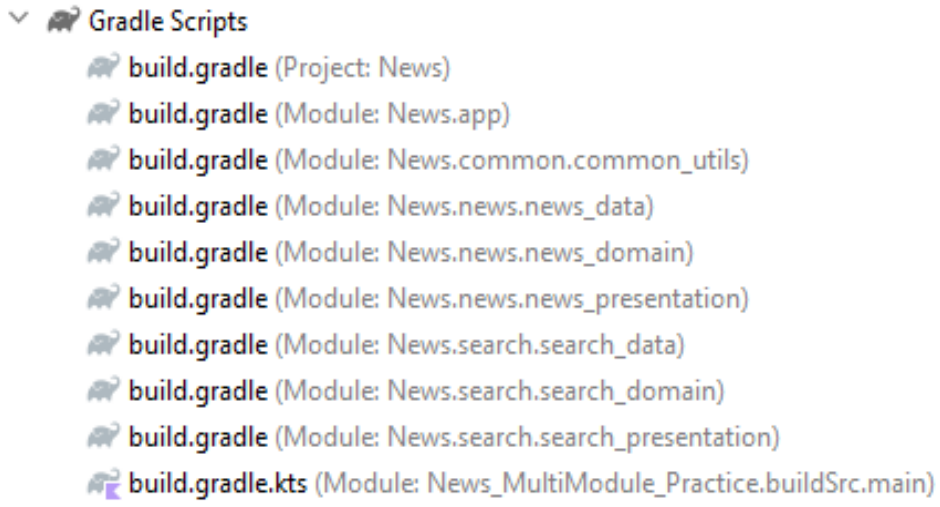
```
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
    kotlinOptions {
        jvmTarget = '1.8'
    }
    buildFeatures{
        dataBinding=true
    }
}
dependencies {
    implementation 'androidx.core:core-ktx:1.6.0'
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'
    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    testImplementation 'junit:junit:4.+'
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    // Retrofit
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'com.squareup.okhttp3:okhttp:4.9.0'
    implementation 'com.squareup.retrofit2:converter-scalars:2.1.0'
    // Coroutines
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-core:1.5.0'
    implementation 'org.jetbrains.kotlinx:kotlinx-coroutines-android:1.5.0'
    // Coroutine Lifecycle Scopes
    implementation "androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1"
    implementation "androidx.lifecycle:lifecycle-runtime-ktx:2.3.1"
    // Navigation Components
    implementation "androidx.navigation:navigation-fragment-ktx:2.4.0-alpha09"
    implementation "androidx.navigation:navigation-ui-ktx:2.4.0-alpha09"
    // Glide
    implementation 'com.github.bumptech.glide:glide:4.12.0'
    kapt 'com.github.bumptech.glide:compiler:4.12.0'
    // Activity KTX for viewModels()
    implementation "androidx.activity:activity-ktx:1.3.1"
    //Dagger - Hilt
    implementation "com.google.dagger:hilt-android:2.38.1"
    kapt "com.google.dagger:hilt-android-compiler:2.38.1"
    implementation "androidx.hilt:hilt-lifecycle-viewmodel:1.0.0-alpha03"
    kapt "androidx.hilt:hilt-compiler:1.0.0"
}
```

Modularization




network
NewsApiService
repository
NewsRepoImpl
room
NewsDAO
com.vision.andorid.news_data (andr
com.vision.andorid.news_data (test)
java (generated)
res (generated)
news_domain
manifests
java
com.vision.andorid.news_domain
di
NewsDomainModule
model
Article
repository
NewsRepository
use_case
GetNewsArticleUseCase
com.vision.andorid.news_domain (a
com.vision.andorid.news_domain (t
java (generated)
res (generated)
news_presentation
manifests
java
com.vision.andorid.news_presentation
NewsActivity.kt
NewsAdapter
NewsState
NewsViewModel
com.vision.andorid.news_presentation (
com.vision.andorid.news_presentation (
java (generated)
res
drawable
layout
activity_news.xml
view_holder_articles.xml
values
res (generated)
search
search_data
manifests
java
com.vision.andorid.search_data
di
SearchModule
mappers
Mappers.kt


model
ArticleDTO
NewsResponse
SourceDTO
network
SearchApi
repository
SearchRepositoryImpl
com.vision.andorid.search_data (andı
com.vision.andorid.search_data (test)
java (generated)
res (generated)
search_domain
manifests
java
com.vision.andorid.search_domain
di
SearchDomainModule
model
Article
repository
SearchRepository
use_cases
GetSearchArticleUseCase
com.vision.andorid.search_domain (a
com.vision.andorid.search_domain (t
java (generated)
res (generated)
search_presentation
manifests
java
com.vision.andorid.search_presentation
Constant
NewsAdapter
SearchActivity.kt
SearchState
SearchViewModel
com.vision.andorid.search_presentation
com.vision.andorid.search_presentation
java (generated)
res
drawable
layout
activity_search.xml
view_holder_search_articles.xml
values
res (generated)
buildSrc
main
java
Dependencies.kt
test


Gradle Scripts
build.gradle (Project: News)
build.gradle (Module: News.app)
build.gradle (Module: News.common.common_utils)
build.gradle (Module: News.news.news_data)
build.gradle (Module: News.news.news_domain)
build.gradle (Module: News.news.news_presentation)
build.gradle (Module: News.search.search_data)
build.gradle (Module: News.search.search_domain)
build.gradle (Module: News.search.search_presentation)
build.gradle.kts (Module: News_MultiModule_Practice.buildSrc.main)

AndroidManifest.xml:

```
<uses-permission android:name="android.permission.INTERNET"/>
<!--Part-3(step-5f).-->
<application
   android:name=".BaseApplication"
```

DefaultNavigator.kt:

```
package com.vision.andorid.news.navigation
import com.vision.andorid.common_utils.Activities
import com.vision.andorid.common_utils.Navigator
import com.vision.andorid.news_presentation.GoToNewsActivity
import com.vision.andorid.search_presentation.GoToSearchActivity

class DefaultNavigator : Navigator.Provider {
   //Part-4(step-7)
   override fun getActivities(activities: Activities): Navigator {
      return when (activities) {
         Activities.NewsActivity -> {
            GoToNewsActivity
         }
         Activities.SearchActivity -> {
            GoToSearchActivity
         }
      }
   }
}
```

AppDatabase.kt:

```
package com.vision.andorid.news.room
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
import com.vision.andorid.news_data.room.NewsDAO
import com.vision.andorid.news_domain.model.Article

//Part-6(step-3)
@Database(entities = [Article::class], version = 1, exportSchema = false)
abstract class AppDatabase : RoomDatabase() {
   companion object {
      fun getInstance(context: Context): AppDatabase {
         return Room.databaseBuilder(context, AppDatabase::class.java, "app_db")
            .fallbackToDestructiveMigration().build()
      }
```

```
    }
    abstract fun getNewsDao(): NewsDAO
}
```

BaseApplication.kt:

```
package com.vision.andorid.news
import android.app.Application
import dagger.hilt.android.HiltAndroidApp

//Part-3(step-4).
@HiltAndroidApp
class BaseApplication : Application() {
}
```

MainActivity.kt:

```
package com.vision.andorid.news
import android.os.Bundle
import android.os.Handler
import android.os.Looper
import androidx.appcompat.app.AppCompatActivity
import androidx.core.content.ContextCompat
import com.vision.andorid.common_utils.Activities
import com.vision.andorid.common_utils.Navigator
import com.vision.andorid.news.databinding.ActivityMainBinding
import dagger.hilt.android.AndroidEntryPoint
import javax.inject.Inject

@AndroidEntryPoint
class MainActivity : AppCompatActivity() {

    @Inject
    lateinit var provider:Navigator.Provider

    private var _binding:ActivityMainBinding?=null
    private val binding:ActivityMainBinding
    get() = _binding!!

    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        window.statusBarColor = ContextCompat.getColor(this, R.color.white)
        _binding = ActivityMainBinding.inflate(layoutInflater)
        setContentView(binding.root)
```

```
        //Part-4(step-5) for navigation.
        //Part-5(step-1) for news module.
        //Part-7f(step-1) for search module.
        Handler(Looper.myLooper()!!).postDelayed({
            provider.getActivities(Activities.NewsActivity).navigate(this)
            finish()
        }, 1500)
    }
}
//Runnable means code for run.
```

MainModule.kt:

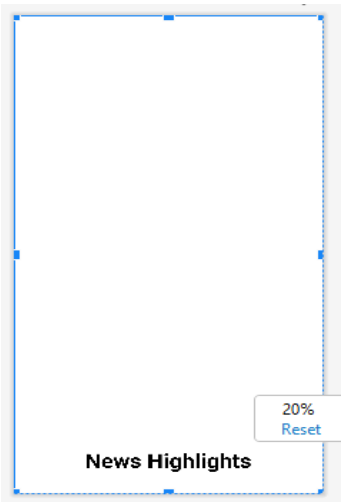
```
package com.vision.andorid.news
import android.content.Context
import com.vision.andorid.common_utils.Navigator
import com.vision.andorid.news.navigation.DefaultNavigator
import com.vision.andorid.news.room.AppDatabase
import com.vision.andorid.news_data.room.NewsDAO
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import javax.inject.Singleton

@InstallIn(SingletonComponent::class)
@Module
object MainModule {

    //Part-4(step-6)
    @Provides
    @Singleton
    fun provideProvider(): Navigator.Provider {
        return DefaultNavigator()
    }

    //Part-6(step-4)
    @Provides
    @Singleton
    fun provideNewsDatabase(@ApplicationContext context: Context): AppDatabase {
        return AppDatabase.getInstance(context)
    }
```

```
    @Provides
    fun provideNewsDAO(appDatabase: AppDatabase): NewsDAO {
        return appDatabase.getNewsDao()
    }

}

//AppDatabase type var/val can be inject.
```

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>
    </data>

    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        tools:context=".MainActivity">

        <!--Part-2(step-3).-->
        <com.airbnb.lottie.LottieAnimationView
            android:id="@+id/lottie_anim"
            android:layout_width="200dp"
            android:layout_height="200dp"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            app:lottie_autoPlay="true"
            app:lottie_loop="false"
            app:lottie_rawRes="@raw/news_animation"/>
            <!--Part-2(step-4).-->

        <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="News Highlights"
            android:textColor="@color/black"
            android:textSize="30sp"
            android:textStyle="bold"
```

```
        android:layout_marginBottom="30dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent" />

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



CommonModule.kt:

```
package com.vision.andorid.common_utils.network
import com.vision.andorid.common_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
object CommonModule {

    @Provides
    @Singleton
    fun provideRetrofit(): Retrofit {
        return Retrofit.Builder().baseUrl(Constants.BASE_URL)
            .addConverterFactory(GsonConverterFactory.create()).build()
    }
}
```

Activities.kt:

```
package com.vision.andorid.common_utils

//Part-4(step-3)
sealed class Activities{
   object NewsActivity:Activities()
   object SearchActivity:Activities()
}
```

Constants.kt:

```
package com.vision.andorid.common_utils

object Constants {
   const val BASE_URL="https://newsapi.org/v2/"
   const val API_KEY=""
   const val CATEGORY="business"
   const val COUNTRY="us"
}
```

Navigator.kt:

```
package com.vision.andorid.common_utils
import android.app.Activity

//Part-4(step-4)
interface Navigator {
   fun navigate(activity:Activity)
   interface Provider{
      fun getActivities(activities: Activities):Navigator
   }
}
```

Resource.kt:

```
package com.vision.andorid.common_utils

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

NewsDataModule.kt:

```
package com.vision.andorid.news_data.di
import com.vision.andorid.news_data.network.NewsApiService
import com.vision.andorid.news_data.repository.NewsRepoImpl
import com.vision.andorid.news_data.room.NewsDAO
import com.vision.andorid.news_domain.repository.NewsRepository
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import retrofit2.Retrofit

@InstallIn(SingletonComponent::class)
@Module
object NewsDataModule {
   @Provides
   fun provideNewsApiService(retrofit: Retrofit): NewsApiService {
      return retrofit.create(NewsApiService::class.java)
   }
   //Part-6(step-6f)
   @Provides
   fun provideNewsRepository(newsApiService: NewsApiService,newsDAO:
NewsDAO):NewsRepository{
      return NewsRepoImpl(newsApiService,newsDAO)
   }
}
/*I think,If we create obj. of NewsRepoImpl,
then we find obj. of NewsRepository,
because NewsRepoImpl extend NewsRepository.*/
```

Mappers.kt:

```
package com.vision.andorid.news_data.mapper
import com.vision.andorid.news_data.model.ArticleDTO
import com.vision.andorid.news_domain.model.Article

fun ArticleDTO.toDomainArticle(): Article {
   return Article(
      author = this.author?:"",
      content = this.content?:"",
      description = this.description?:"",
      title = this.title?:"",
      urlToImage = this.urlToImage?:""
   )
}
```

ArticleDTO.kt:

```
package com.vision.andorid.news_data.model

data class ArticleDTO(
   val author: String?,
   val content: String?,
   val description: String?,
   val publishedAt: String?,
   val source: SourceDTO?,
   val title: String?,
   val url: String?,
   val urlToImage: String?
)
```

NewsResponse.kt:

```
package com.vision.andorid.news_data.model

data class NewsResponse(
   val articles: List<ArticleDTO>,
   val status: String,
   val totalResults: Int
)
```

SourceDTO.kt:

```
package com.vision.andorid.news_data.model

data class SourceDTO(
   val id: String,
   val name: String
)
```

NewsApiService.kt:

```
package com.vision.andorid.news_data.network
import com.vision.andorid.common_utils.Constants
import com.vision.andorid.news_data.model.NewsResponse
import retrofit2.http.GET
import retrofit2.http.Query

interface NewsApiService {

 //https://newsapi.org/v2/topheadlines?country=us&category=business&apiKey=your_api_key

   @GET("top-headlines")
   suspend fun getNewsArticles( //Part-5(step-7,8)
      @Query("country") country:String,
```

```
    @Query("category") category: String=Constants.CATEGORY,
    @Query("apiKey") apiKey:String = Constants.API_KEY
  ):NewsResponse
}
```

NewsRepoImpl.kt:

```
package com.vision.andorid.news_data.repository
import com.vision.andorid.news_data.mapper.toDomainArticle
import com.vision.andorid.news_data.network.NewsApiService
import com.vision.andorid.news_data.room.NewsDAO
import com.vision.andorid.news_domain.model.Article
import com.vision.andorid.news_domain.repository.NewsRepository

class NewsRepoImpl(private val newsApiService: NewsApiService, private val newsDAO:
NewsDAO): NewsRepository {
  override suspend fun getNewsArticle(): List<Article> {
    return try {
      //Part-5(step-6)
      val temp = newsApiService.getNewsArticles(country = "us").articles.map {
      it.toDomainArticle() }
      newsDAO.insertList(temp)          // Part-6(step-5)
      newsDAO.getNewsArticle()          //Part-5(step-9.1)
    } catch (e: Exception) {
       newsDAO.getNewsArticle()         //Part-5(step-9.2)
    }
  }
}
//return try work for both (try & catch) & return value from last line.
```

NewsDAO.kt:

```
package com.vision.andorid.news_data.room
import androidx.room.Dao
import androidx.room.Insert
import androidx.room.OnConflictStrategy
import androidx.room.Query
import com.vision.andorid.news_domain.model.Article

@Dao    //Part-6(step-2)
interface NewsDAO {
  @Insert(onConflict = OnConflictStrategy.REPLACE)
  suspend fun insertList(list:List<Article>)
  @Query("SELECT * FROM ARTICLE")
  suspend fun getNewsArticle():List<Article>
```

NewsDomainModule.kt:

```
package com.vision.andorid.news_domain.di
import com.vision.andorid.news_domain.repository.NewsRepository
import com.vision.andorid.news_domain.use_case.GetNewsArticleUseCase
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent

@InstallIn(SingletonComponent::class)
@Module
object NewsDomainModule {

   @Provides
   fun provideGetNewsUseCase(newsRepository: NewsRepository):GetNewsArticleUseCase{
      return GetNewsArticleUseCase(newsRepository)
   }
}
```

Article.kt:

```
package com.vision.andorid.news_domain.model
import androidx.room.Entity
import androidx.room.PrimaryKey

@Entity   //Part-6(step-1)
data class Article(
   val author: String,
   val content: String,
   val description: String,
   @PrimaryKey(autoGenerate = false)
   val title: String,
   val urlToImage: String
)
```

NewsRepository.kt:

```
package com.vision.andorid.news_domain.repository
import com.vision.andorid.news_domain.model.Article

interface NewsRepository {
   //Part-5(step-5,10)
   suspend fun getNewsArticle():List<Article>
 }
```

GetNewsArticleUseCase.kt:

```
package com.vision.andorid.news_domain.use_case
import com.vision.andorid.common_utils.Resource
import com.vision.andorid.news_domain.model.Article
import com.vision.andorid.news_domain.repository.NewsRepository
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow

class GetNewsArticleUseCase (private val newsRepository:NewsRepository) {
   operator fun invoke():Flow<Resource<List<Article>>> = flow {
      emit(Resource.Loading())
      try {
         emit(Resource.Success(data=newsRepository.getNewsArticle()))  //Part-5(step-4,11)
      }catch (e:Exception){
         emit(Resource.Error(message = e.message.toString()))
      }
   }
}
```

NewsActivity.kt:

```
package com.vision.andorid.news_presentation
import android.app.Activity
import android.content.Intent
import android.os.Bundle
import android.view.View
import android.widget.Toast
import androidx.activity.viewModels
import androidx.appcompat.app.AppCompatActivity
import androidx.lifecycle.lifecycleScope
import com.vision.andorid.common_utils.Activities
import com.vision.andorid.common_utils.Navigator
import com.vision.andorid.news_presentation.databinding.ActivityNewsBinding
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest
import javax.inject.Inject

@AndroidEntryPoint
class NewsActivity : AppCompatActivity() {
   companion object {     //Part-4(step-9f)
      fun launchActivity(activity: Activity) {
         val intent = Intent(activity, NewsActivity::class.java)
         activity.startActivity(intent)
      }
   }
```

```
@Inject
lateinit var provider:Navigator.Provider

private var _binding: ActivityNewsBinding? = null
private val binding: ActivityNewsBinding
   get() = _binding!!

//Part-5(step-2)
private val newsViewModel: NewsViewModel by viewModels()
private val newsAdapter = NewsAdapter()

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

   initView()
   setObservers()
}

private fun initView() {
   binding.rvArticles.adapter = newsAdapter
   //Part-7f(step-2)
   binding.ivGoToSearch.setOnClickListener {
      provider.getActivities(Activities.SearchActivity).navigate(this)
   }
}

private fun setObservers() {
   lifecycleScope.launchWhenStarted {
      //Part-5(step-14)
      newsViewModel.newsArticle.collectLatest {
         if (it.isLoading) {
            binding.progressBar.visibility= View.VISIBLE
         }
         if (it.error.isNotBlank()) {
            binding.progressBar.visibility= View.GONE
            Toast.makeText(this@NewsActivity, it.error, Toast.LENGTH_LONG).show()
         }
         it.data?.let {
            binding.progressBar.visibility= View.GONE
            newsAdapter.setData(it) //Part-5(step-15)
         }
      }
```

```
    }
  }
}

//Part-4(step-8)
object GoToNewsActivity: Navigator {
  override fun navigate(activity: Activity) {
    NewsActivity.launchActivity(activity)
  }
}
```

NewsAdapter.kt:

```
package com.vision.andorid.news_presentation
import android.view.LayoutInflater
import android.view.ViewGroup
import android.widget.ImageView
import androidx.recyclerview.widget.RecyclerView
import androidx.swiperefreshlayout.widget.CircularProgressDrawable
import com.bumptech.glide.Glide
import com.vision.andorid.news_domain.model.Article
import com.vision.andorid.news_presentation.databinding.ViewHolderArticlesBinding

class NewsAdapter : RecyclerView.Adapter<NewsAdapter.MyViewHolder>() {

  private var list = listOf<Article>()
  //Part-5(step-16f)
  fun setData(list: List<Article>) {
    this.list = list
    notifyItemInserted(this.list.lastIndex)
  }

  inner class MyViewHolder(val viewDataBinding: ViewHolderArticlesBinding) :
    RecyclerView.ViewHolder(viewDataBinding.root)

  override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
    val binding =
      ViewHolderArticlesBinding.inflate(LayoutInflater.from(parent.context), parent, false)
    return MyViewHolder(binding)
  }

  override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
    holder.viewDataBinding.apply {
      val item = list[position]
```

```
            ivArticle.loadImage(item.urlToImage)
            tvHeadlines.text = item.title
            tvContent.text = item.content
        }
    }

    override fun getItemCount(): Int {
        return this.list.size
    }

    fun ImageView.loadImage(url: String) {
        val circularProgressDrawable = CircularProgressDrawable(this.context)
        circularProgressDrawable.strokeWidth = 5f
        circularProgressDrawable.centerRadius = 30f
        circularProgressDrawable.start()
        Glide.with(this).load(url).placeholder(circularProgressDrawable)
            .error(com.google.android.material.R.drawable.mtrl_ic_error).into(this)
    }
}
```

NewsState.kt:

```
package com.vision.andorid.news_presentation
import com.vision.andorid.news_domain.model.Article

data class NewsState(
    val isLoading:Boolean=false,
    val error:String="",
    val data:List<Article>?=null
)
```

NewsViewModel.kt:

```
package com.vision.andorid.news_presentation
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.vision.andorid.common_utils.Resource
import com.vision.andorid.news_domain.use_case.GetNewsArticleUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import javax.inject.Inject
```

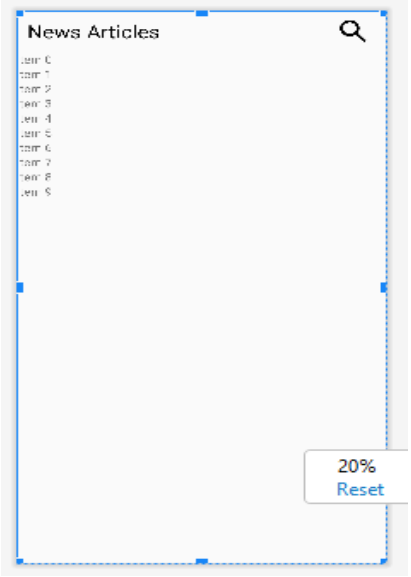
```
@HiltViewModel
class NewsViewModel @Inject constructor(private val getNewsArticleUseCase:
GetNewsArticleUseCase) :
   ViewModel() {

   private val _newsArticle = MutableStateFlow(NewsState())
   //Part-5(step-13)
   val newsArticle: StateFlow<NewsState> = _newsArticle

   init {
      //Part-5(step-3.1)
      getNewsArticles() //init{} work when object of this class created.
   }
   //Part-5(step-3.2)
   fun getNewsArticles() {
      //Part-5(step-3.3)
      getNewsArticleUseCase().onEach {
         when (it) {
            is Resource.Loading -> {
               _newsArticle.value = NewsState(isLoading = true)
            }
            is Resource.Error -> {
               _newsArticle.value = NewsState(error = it.message)
            }
            is Resource.Success -> {
               //Part-5(step-12)
               _newsArticle.value = NewsState(data = it.data)
            }
         }
      }.launchIn(viewModelScope)
   }
}
```

activity_news.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>

   </data>
```

```
<androidx.constraintlayout.widget.ConstraintLayout
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   tools:context=".NewsActivity">

   <androidx.constraintlayout.widget.ConstraintLayout
      android:id="@+id/constraintLayout"
      android:layout_width="match_parent"
      android:layout_height="55dp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintStart_toStartOf="parent"
      app:layout_constraintTop_toTopOf="parent">

      <androidx.appcompat.widget.AppCompatTextView
         android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginStart="12dp"
         android:text="News Articles"
         android:textColor="@color/black"
         android:textSize="24sp"
         app:layout_constraintBottom_toBottomOf="parent"
         app:layout_constraintStart_toStartOf="parent"
         app:layout_constraintTop_toTopOf="parent" />

      <ImageView
         android:layout_width="40dp"
         android:layout_height="40dp"
         android:id="@+id/ivGoToSearch"
         android:src="@drawable/ic_baseline_search_24"
         app:layout_constraintBottom_toBottomOf="parent"
         android:layout_marginEnd="16dp"
         app:layout_constraintEnd_toEndOf="parent"
         app:layout_constraintTop_toTopOf="parent" />

   </androidx.constraintlayout.widget.ConstraintLayout>

   <androidx.recyclerview.widget.RecyclerView
      android:id="@+id/rvArticles"
      android:layout_width="match_parent"
      android:layout_height="0dp"
      app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
      app:layout_constraintBottom_toBottomOf="parent"
      app:layout_constraintTop_toBottomOf="@+id/constraintLayout"
      app:layout_constraintVertical_bias="0.0"
```

```
        tools:layout_editor_absoluteX="0dp" />

    <ProgressBar
        android:id="@+id/progressBar"
        android:visibility="gone"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="@+id/rvArticles"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

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



view_holder_articles.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>
  </data>

  <androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="wrap_content">
```

```
    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tvHeadlines"
        style="@style/TextAppearance.MaterialComponents.Headline5"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="8dp"
        android:paddingVertical="12dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        tools:text="Headline 5" />
    <androidx.appcompat.widget.AppCompatImageView
        android:id="@+id/ivArticle"
        android:layout_width="match_parent"
        android:layout_height="300dp"
        android:scaleType="centerCrop"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/tvHeadlines" />
    <androidx.appcompat.widget.AppCompatTextView
        android:id="@+id/tvContent"
        style="@style/TextAppearance.MaterialComponents.Body1"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:paddingHorizontal="8dp"
        android:paddingVertical="12dp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/ivArticle"
        tools:text="Headline 5" />
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

SearchModule.kt:

```
package com.vision.andorid.search_data.di
import com.vision.andorid.search_data.network.SearchApi
import com.vision.andorid.search_data.repository.SearchRepositoryImpl
import com.vision.andorid.search_domain.repository.SearchRepository
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent
import retrofit2.Retrofit

@InstallIn(SingletonComponent::class)
@Module
object SearchModule {

   @Provides
   fun provideSearchApi(retrofit: Retrofit):SearchApi{
      return retrofit.create(SearchApi::class.java)
   }
   @Provides
   fun provideSearchRepo(searchApi: SearchApi):SearchRepository{
      return SearchRepositoryImpl(searchApi)
   }
}
//SearchApi will receive from above.
```

Mappers.kt:

```
package com.vision.andorid.search_data.mappers
import com.vision.andorid.search_data.model.ArticleDTO
import com.vision.andorid.search_domain.model.Article

fun ArticleDTO.toDomainArticle(): Article {
   return Article(
      author = this.author ?: "",
      content = this.content ?: "",
      description = this.description ?: "",
      title = this.title ?: "",
      urlToImage = this.urlToImage ?: ""
   )
}
```

ArticleDTO.kt:

```
package com.vision.andorid.search_data.model

data class ArticleDTO(
   val author: String?,
   val content: String?,
   val description: String?,
   val publishedAt: String?,
   val source: SourceDTO?,
   val title: String?,
   val url: String?,
   val urlToImage: String?
)
```

NewsResponse.kt:

```
package com.vision.andorid.search_data.model

data class NewsResponse(
   val articles: List<ArticleDTO>,
   val status: String,
   val totalResults: Int
)
```

SourceDTO.kt:

```
package com.vision.andorid.search_data.model

data class SourceDTO(
   val id: String,
   val name: String
)
```

SearchApi.kt:

```
package com.vision.andorid.search_data.network
import com.vision.andorid.search_data.model.NewsResponse
import retrofit2.http.GET
import retrofit2.http.QueryMap

interface SearchApi {
// https://newsapi.org/v2/everything?q=apple&from=2022-10-14&to=2022-10-
14&sortBy=popularity&apiKey=API_KEY

   @GET("everything")    //Part-7f(step-8 & 9)
   suspend fun getSearchArticles(
      @QueryMap map:MutableMap<String,String>
   ) : NewsResponse
}
```

SearchRepositoryImpl.kt:

```
package com.vision.andorid.search_data.repository
import android.util.Log
import com.vision.andorid.search_data.mappers.toDomainArticle
import com.vision.andorid.search_data.network.SearchApi
import com.vision.andorid.search_domain.model.Article
import com.vision.andorid.search_domain.repository.SearchRepository

class SearchRepositoryImpl(private val searchApi: SearchApi) : SearchRepository {
   override suspend fun getSearchArticles(map: MutableMap<String, String>): List<Article> {
     //Part-7f(step-7 & 10)
     return searchApi.getSearchArticles(map).articles.map { it.toDomainArticle() }
   }
}
```

SearchDomainModule.kt:

```
package com.vision.andorid.search_domain.di
import com.vision.andorid.search_domain.repository.SearchRepository
import com.vision.andorid.search_domain.use_cases.GetSearchArticleUseCase
import dagger.Module
import dagger.Provides
import dagger.hilt.InstallIn
import dagger.hilt.components.SingletonComponent

@InstallIn(SingletonComponent::class)
@Module
object SearchDomainModule {

   @Provides
   fun provideSearchUseCase(searchRepository: SearchRepository):GetSearchArticleUseCase{
      return GetSearchArticleUseCase(searchRepository)
   }
}
//SearchRepository will receive from SearchModule.
```

Article.kt:

```
package com.vision.andorid.search_domain.model
data class Article(
   val author: String,
   val content: String,
   val description: String,
   val title: String,
   val urlToImage: String    //Part-7f(step-16)
)
```

SearchRepository.kt:

```
package com.vision.andorid.search_domain.repository
import com.vision.andorid.search_domain.model.Article

interface SearchRepository {
   //Part-7f(step-6 & 11)
   suspend fun getSearchArticles(map:MutableMap<String,String>):List<Article>
}
```

GetSearchArticleUseCase.kt:

```
package com.vision.andorid.search_domain.use_cases
import android.util.Log
import com.vision.andorid.common_utils.Resource
import com.vision.andorid.search_domain.model.Article
import com.vision.andorid.search_domain.repository.SearchRepository
import kotlinx.coroutines.flow.Flow
import kotlinx.coroutines.flow.flow

class GetSearchArticleUseCase (private val searchRepository: SearchRepository) {

   operator fun invoke(map:MutableMap<String,String>):Flow<Resource<List<Article>>> = flow
{
      emit(Resource.Loading())
      try {
         //Part-7f(step-5 & 12)
         emit(Resource.Success(searchRepository.getSearchArticles(map)))
      }catch (e:Exception){
         emit(Resource.Error(e.message.toString()))
      }
   }
}

/*
Specifying an invoke operator on a class allows it to be
called on any instances of the class without a method name.
*/
```

Constant.kt:

```
package com.vision.andorid.search_presentation

object Constant {
   // https://newsapi.org/v2/everything?q=apple&from=2022-10-14&to=2022-10-
   14&sortBy=popularity&apiKey=API_KEY

   const val START_DATE="from"
   const val END_DATE="to"
   const val apiKey="apiKey"
   const val QUERY="q"
   const val KEY=""
}
```

NewsAdapter.kt:

```
package com.vision.andorid.search_presentation
import android.view.LayoutInflater
import android.view.ViewGroup
import android.widget.ImageView
import androidx.recyclerview.widget.RecyclerView
import androidx.swiperefreshlayout.widget.CircularProgressDrawable
import com.bumptech.glide.Glide
import com.vision.andorid.search_domain.model.Article
import com.vision.andorid.search_presentation.databinding.ViewHolderSearchArticlesBinding

class NewsAdapter : RecyclerView.Adapter<NewsAdapter.MyViewHolder>() {

   private var list = listOf<Article>()

   //Part-7f(step-16f)
   fun setData(list: List<Article>) {
      this.list = list
      notifyItemInserted(this.list.lastIndex)
   }

   inner class MyViewHolder(val viewDataBinding: ViewHolderSearchArticlesBinding) :
      RecyclerView.ViewHolder(viewDataBinding.root)

   override fun onCreateViewHolder(parent: ViewGroup, viewType: Int): MyViewHolder {
      val binding = ViewHolderSearchArticlesBinding.inflate(LayoutInflater.from(parent.context),
     parent, false)
      return MyViewHolder(binding)
   }
```

```
    override fun onBindViewHolder(holder: MyViewHolder, position: Int) {
        holder.viewDataBinding.apply {
            val item = list[position]
            ivArticle.loadImage(item.urlToImage)
            tvHeadlines.text = item.title
            tvContent.text = item.content
        }
    }

    override fun getItemCount(): Int {
        return this.list.size
    }

    fun ImageView.loadImage(url: String) {
        val circularProgressDrawable = CircularProgressDrawable(this.context)
        circularProgressDrawable.strokeWidth = 5f
        circularProgressDrawable.centerRadius = 30f
        circularProgressDrawable.start()
        Glide.with(this).load(url).placeholder(circularProgressDrawable)
            .error(com.google.android.material.R.drawable.mtrl_ic_error).into(this)
    }
}
```

SearchActivity.kt:

```
package com.vision.andorid.search_presentation
import android.app.Activity
import android.content.Intent
import androidx.appcompat.app.AppCompatActivity
import android.os.Bundle
import android.util.Log
import android.view.View
import android.widget.Toast
import androidx.activity.viewModels
import androidx.core.widget.doAfterTextChanged
import androidx.lifecycle.lifecycleScope
import androidx.recyclerview.widget.LinearLayoutManager
import com.google.android.material.datepicker.MaterialDatePicker
import com.vision.andorid.common_utils.Navigator
import com.vision.andorid.search_presentation.databinding.ActivitySearchBinding
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.flow.collectLatest
import java.text.SimpleDateFormat
```

```
@AndroidEntryPoint
class SearchActivity : AppCompatActivity() {
    companion object{
        fun launchActivity(activity: Activity){
            val intent = Intent(activity,SearchActivity::class.java)
            activity.startActivity(intent)
        }
    }
    private var _binding:ActivitySearchBinding?=null
    val binding:ActivitySearchBinding
    get()=_binding!!

    private val viewModel:SearchViewModel by viewModels()
    private val newsAdapter = NewsAdapter()

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

        initView()
        setObserver()
    }
    private fun setObserver() {
        lifecycleScope.launchWhenStarted {
            //Part-7f(step-14)
            viewModel.searchArticles.collectLatest {
                if(it.isLoading){
                    binding.progressBar.visibility= View.VISIBLE
                }
                if(it.error.isNotBlank()){
                    binding.progressBar.visibility= View.GONE
                    Toast.makeText(this@SearchActivity,it.error,Toast.LENGTH_LONG).show()
                }
                it.data?.let {
                    binding.progressBar.visibility= View.GONE
                    //Part-7f(step-15)
                    newsAdapter.setData(it)
                }
            }
        }

    }
```

```
    private fun initView() {
        binding.rvSearch.adapter = newsAdapter
        binding.searchTitle.doAfterTextChanged {
            val map = mutableMapOf<String,String>()
            map[Constant.apiKey]=Constant.KEY
            map[Constant.QUERY]=it.toString()
            //Part-7f(step-3.1)
            viewModel.getSearchArticles(map)
        }
        binding.ivRange.setOnClickListener {
            val datePicker = MaterialDatePicker.Builder.dateRangePicker().build()
            datePicker.show(this.supportFragmentManager,"range picker")

           datePicker.addOnPositiveButtonClickListener {
              val start = changeDateFormat(it.first)
              val end = changeDateFormat(it.second)

              val map = mutableMapOf<String,String>()
              map[Constant.apiKey]=Constant.KEY
              map[Constant.QUERY]=binding.searchTitle.text.toString()
              map[Constant.START_DATE]=start
              map[Constant.END_DATE]=end
              //Part-7f(step-3.2)
              viewModel.getSearchArticles(map)
           }
        }
    }

    fun changeDateFormat(long:Long?):String{
        return try {
            val simpleDateFormat= SimpleDateFormat("yyyy-MM-dd")
            simpleDateFormat.format(long)
        }catch (e:Exception){
             ""
        }
    }
}

object GoToSearchActivity: Navigator {
    override fun navigate(activity: Activity) {
        SearchActivity.launchActivity(activity)
    }
}
```

SearchState.kt:

```
package com.vision.andorid.search_presentation
import com.vision.andorid.search_domain.model.Article

data class SearchState(
   val isLoading:Boolean=false,
   val error:String="",
   val data:List<Article>?=null
)
```

SearchViewModel.kt:

```
package com.vision.andorid.search_presentation
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.vision.andorid.common_utils.Resource
import com.vision.andorid.search_domain.use_cases.GetSearchArticleUseCase
import dagger.hilt.android.lifecycle.HiltViewModel
import kotlinx.coroutines.flow.MutableStateFlow
import kotlinx.coroutines.flow.StateFlow
import kotlinx.coroutines.flow.launchIn
import kotlinx.coroutines.flow.onEach
import javax.inject.Inject

@HiltViewModel
class SearchViewModel @Inject constructor(private val getSearchArticleUseCase:
GetSearchArticleUseCase) :
   ViewModel() {

   private val _searchArticles = MutableStateFlow(SearchState())
   //Part-7f(step-13.2)
   val searchArticles: StateFlow<SearchState> = _searchArticles

   fun getSearchArticles(map: MutableMap<String, String>) {
      //Part-7f(step-4)
      getSearchArticleUseCase(map).onEach {

         when (it) {
            is Resource.Loading -> {
               _searchArticles.value = SearchState(isLoading = true)
            }
            is Resource.Error -> {
               _searchArticles.value = SearchState(error = it.message)
```

```
            }
            is Resource.Success -> {
                //Part-7f(step-13.1)
                _searchArticles.value = SearchState(data = it.data)
            }
        }
    }.launchIn(viewModelScope)
  }

}
```

activity_search.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>

  </data>

  <androidx.constraintlayout.widget.ConstraintLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".SearchActivity">

    <androidx.constraintlayout.widget.ConstraintLayout
      android:id="@+id/constraintLayout"
      android:layout_width="match_parent"
      android:layout_height="55dp"
      app:layout_constraintEnd_toEndOf="parent"
      app:layout_constraintTop_toTopOf="parent">

      <EditText
        android:id="@+id/searchTitle"
        android:layout_width="0dp"
        android:layout_height="match_parent"
        android:layout_marginEnd="8dp"
        android:background="@null"
        android:hint="Search here..."
        android:paddingHorizontal="8dp"
        app:layout_constraintEnd_toStartOf="@+id/ivRange"
        app:layout_constraintStart_toStartOf="parent" />
```

```
        <ImageView
            android:id="@+id/ivRange"
            android:layout_width="35dp"
            android:layout_height="35dp"
            android:layout_marginEnd="8dp"
            android:src="@drawable/ic_baseline_calendar_today_24"
            app:layout_constraintBottom_toBottomOf="parent"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent" />
    </androidx.constraintlayout.widget.ConstraintLayout>
    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rvSearch"
        android:layout_width="match_parent"
        tools:listitem="@layout/view_holder_search_articles"
        android:layout_height="0dp"
        app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/constraintLayout" />
    <ProgressBar
        android:id="@+id/progressBar"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/constraintLayout"
        tools:visibility="visible" />
  </androidx.constraintlayout.widget.ConstraintLayout>
</layout>
```

view_holder_search_articles.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"
    tools:ignore="MissingDefaultResource">

    <data>

    </data>

    <!--//Part-7f(step-15)-->
    <androidx.constraintlayout.widget.ConstraintLayout
        android:layout_width="match_parent"
        android:layout_height="wrap_content">

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/tvHeadlines"
            style="@style/TextAppearance.MaterialComponents.Headline5"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingHorizontal="8dp"
            android:paddingVertical="12dp"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintTop_toTopOf="parent"
            tools:text="Headline 5" />

        <androidx.appcompat.widget.AppCompatImageView
            android:id="@+id/ivArticle"
            android:layout_width="match_parent"
            android:layout_height="300dp"
            android:scaleType="centerCrop"
            app:layout_constraintEnd_toEndOf="parent"
            app:layout_constraintStart_toStartOf="parent"
            app:layout_constraintTop_toBottomOf="@+id/tvHeadlines" />

        <androidx.appcompat.widget.AppCompatTextView
            android:id="@+id/tvContent"
            style="@style/TextAppearance.MaterialComponents.Body1"
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingHorizontal="8dp"
            android:paddingVertical="12dp"
            app:layout_constraintEnd_toEndOf="parent"
```

```
        app:layout_constraintTop_toBottomOf="@+id/ivArticle"
        tools:text="Headline 5" />

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

Dependencies.kt:

```
//Part-1(step-2).
object Versions {
    const val core = "1.9.0"
    const val appcompat = "1.5.1"
    const val androidMaterial = "1.6.1"
    const val constraintLayout = "2.1.4"

    const val testImplJunit = "4.13.2"
    const val androidTestImplJunit = "1.1.3"
    const val androidTestEspresso = "3.4.0"

    const val retrofit = "2.9.0"
    const val gsonConvertor = "2.9.0"
    const val okHttp = "4.9.0"
    const val scalerConvertor = "2.1.0"

    const val kotlinCoroutines = "1.6.1"
    const val coroutineLifecycleScope = "2.5.1"
```

```
    const val glide = "4.12.0"

    const val viewModelDeligate = "1.6.0"

    const val dagger = "2.44"
    const val hiltCompiler = "1.0.0"

    const val roomVersion = "2.4.3"

    const val swipeRefresh = "1.1.0"

    const val lottieAnimations = "3.4.2"
}

object Deps {
    const val core = "androidx.core:core-ktx:${Versions.core}"
    const val appCompat = "androidx.appcompat:appcompat:${Versions.appcompat}"
    const val androidMaterial =
"com.google.android.material:material:${Versions.androidMaterial}"
    const val constraintLayout =
        "androidx.constraintlayout:constraintlayout:${Versions.constraintLayout}"
}

object TestImplementation {
    const val junit = "junit:junit:${Versions.testImplJunit}"
}

object AndroidTestImplementation {
    const val junit = "androidx.test.ext:junit:${Versions.androidTestImplJunit}"
    const val espresso = "androidx.test.espresso:espresso-core:${Versions.androidTestEspresso}"
}

object Retrofit {
    const val retrofit = "com.squareup.retrofit2:retrofit:${Versions.retrofit}"
    const val gsonConvertor = "com.squareup.retrofit2:converter-
gson:${Versions.gsonConvertor}"
    const val okHttp = "com.squareup.okhttp3:okhttp:${Versions.okHttp}"
    const val scalersConvertors =
        "com.squareup.retrofit2:converter-scalars:${Versions.scalerConvertor}"
}

object Coroutines {
    const val coroutineCore =
        "org.jetbrains.kotlinx:kotlinx-coroutines-core:${Versions.kotlinCoroutines}"
```

```
    const val coroutineAndroid =
        "org.jetbrains.kotlinx:kotlinx-coroutines-android:${Versions.kotlinCoroutines}"
}

object CoroutinesLifecycleScope {
    const val lifecycleViewModel =
        "androidx.lifecycle:lifecycle-viewmodel-ktx:${Versions.coroutineLifecycleScope}"
    const val lifeCycleRuntime =
        "androidx.lifecycle:lifecycle-runtime-ktx:${Versions.coroutineLifecycleScope}"
}

object Glide {
    const val glide = "com.github.bumptech.glide:glide:${Versions.glide}"
    const val annotationProcessor = "com.github.bumptech.glide:compiler:${Versions.glide}"
}

object ViewModelDelegate {
    const val viewModelDeligate = "androidx.activity:activity-ktx:${Versions.viewModelDeligate}"
}

object DaggerHilt {
    const val hilt = "com.google.dagger:hilt-android:${Versions.dagger}"
    const val hiltAndroidCompiler = "com.google.dagger:hilt-android-
compiler:${Versions.dagger}"
    const val hiltCompiler = "androidx.hilt:hilt-compiler:${Versions.hiltCompiler}"
}

object Room {
    const val roomCompiler = "androidx.room:room-compiler:${Versions.roomVersion}"
    const val room = "androidx.room:room-ktx:${Versions.roomVersion}"
}

object CircularProgressBar {
    const val swipeRefresh =
        "androidx.swiperefreshlayout:swiperefreshlayout:${Versions.swipeRefresh}"
}

object LottieAnimations {
    const val lottieAnimations = "com.airbnb.android:lottie:${Versions.lottieAnimations}"
}
```

build.gradle(project):

```
plugins {
   //Part-3(step-3).
   id 'com.google.dagger.hilt.android' version '2.44' apply false
}
```

build.gradle(:app):

```
plugins {
   //Part-3(step-1) for dagger hilt.
   id 'kotlin-kapt'
   id 'com.google.dagger.hilt.android'
}
android {
   //Part-2(step-2).
   dataBinding{
      enabled=true
   }
}

//Part-1(step-3f).
dependencies {
   //Part-4(step-1) for navigation.
   implementation project(":common:common_utils")

   implementation project(":news:news_data")
   implementation project(":news:news_domain")
   implementation project(":news:news_presentation")

   implementation project(":search:search_data")
   implementation project(":search:search_domain")
   implementation project(":search:search_presentation")

   implementation Deps.core
   implementation Deps.appCompat
   implementation Deps.androidMaterial
   implementation Deps.constraintLayout
   testImplementation TestImplementation.junit
   androidTestImplementation AndroidTestImplementation.junit
   androidTestImplementation AndroidTestImplementation.espresso
```

```
    //Part-3(step-2).
    implementation DaggerHilt.hilt
    kapt DaggerHilt.hiltAndroidCompiler
    kapt DaggerHilt.hiltCompiler

    implementation Room.room
    kapt Room.roomCompiler

    //Part-2(step-1) for splash screen.
    implementation LottieAnimations.lottieAnimations
}
```

build.gradle(:common:common_utils):

```
plugins {
    id 'kotlin-kapt'
    id 'com.google.dagger.hilt.android'
}

dependencies {
    implementation Deps.core
    implementation Deps.appCompat
    implementation Deps.androidMaterial
    implementation Deps.constraintLayout
    testImplementation TestImplementation.junit
    androidTestImplementation AndroidTestImplementation.junit
    androidTestImplementation AndroidTestImplementation.espresso

    implementation DaggerHilt.hilt
    kapt DaggerHilt.hiltAndroidCompiler
    kapt DaggerHilt.hiltCompiler

    implementation Retrofit.retrofit
    implementation Retrofit.gsonConvertor
    implementation Retrofit.okHttp
}
```

build.gradle(:news:news_data):

```
same as above plugins.
dependencies {
   same as above dependencies.
   implementation project(":common:common_utils")
   implementation project(":news:news_domain")
}
```

build.gradle(:news:news_domain):

```
same as above plugins.
dependencies {
   same as above dependencies.
   implementation project(":common:common_utils")
}
```

build.gradle(:news:news_presentation):

```
same as above plugins.
dependencies {
   same as above dependencies.
   //Part-4(step-2)
   implementation project(":common:common_utils")
   implementation project(":news:news_domain")
}
```

build.gradle(:search:search_data):

```
same as above plugins.
dependencies {
   same as above dependencies.
   implementation project (":search:search_domain")
   implementation project (":common:common_utils")
}
```

build.gradle(:search:search_domain):

```
same as above plugins.
dependencies {
   same as above dependencies.
   implementation project(":common:common_utils")
}
```

build.gradle(:search:search_presentation):

```
same as above plugins.
dependencies {
  same as above dependencies.
   implementation project(":common:common_utils")
   implementation project(":search:search_domain")
}
```

build.gradle.kts:

```
//Part-1(step-1) for add module & dependencies.
plugins{
   `kotlin-dsl`
}
repositories{
   mavenCentral()
}
```

Unit Testing

What & Why:

Testing smallest piece of code in isolation

Helps in catching bugs

Helps in writing Modular code – features can be added easily

Types of test:

Pure Kotlin/Java logic can be tested with the JVM (Desktop only - JUnit)

For Android tests, we need device(Instrumentation Test)

Android tests can be further divided into UI Tests (Espresso) and Non UI Test

Unit Test

→ JVM Test (Local Unit Test)

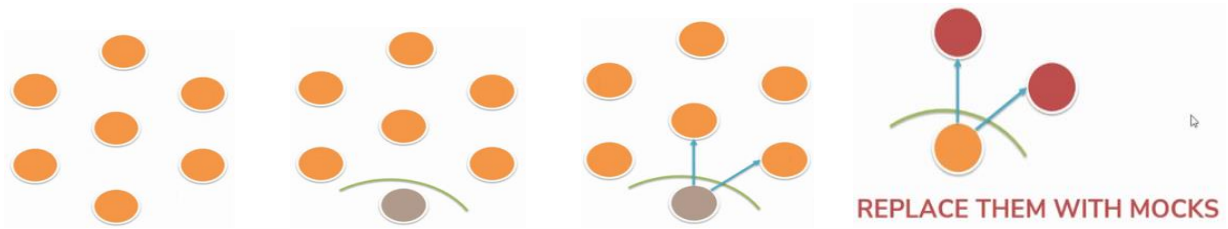
→ On Device Test (Instrumentation Test)

→ UI Tests (Interaction with Views)

→ Non UI Tests (Context, Asset Manager, etc.)

JUNIT: JVM test.
MOCKITO: Mock Object or Fake Objects.
ESPRESSO: UI Interactions.

Mocking:



Benefits of Mocking:

Deterministic

Execution Speed

Parallel Development

PROGRAM

ParameterizedTesting:

Program 1:

Utils.kt:

```
package com.ghani.parameterizedtesting.utils

class Utils {

   fun isPalindrome(input:String):Boolean{
      var i = 0
      var j = input.length -1
      var result = true

      while (i<j){
         if (input[i] != input[j]){
            result = false
            break
         }
         i++
         j--
      }

      return result
   }


   fun validatePassword(input: String)=when{
      input.isBlank() -> {
         "Password is required field"
      }
      input.length < 6 -> {
         "Length of the Password  should be greater than 6"
      }
      input.length > 15  -> {
         "Length of the Password  should be less than 15"
      }
      else -> {
         "Valid"
      }
   }
```

```
    fun reverseString(input:String?):String{
        if(input == null){
            throw IllegalArgumentException("Input String required")
        }
        var chars = input.toCharArray()
        var i = 0
        var j = chars.size-1
        while(i<j){
            val temp = chars[i]
            chars[i]  = chars[j]
            chars[j] = temp
            i++
            j--
        }
        return chars.joinToString("")
    }

}
```

UtilsTest.kt:

```
package com.ghani.parameterizedtesting.utils

import org.junit.After
import org.junit.Assert.*
import org.junit.Before
import org.junit.Test

class UtilsTest {

    /*
    //Part-1
    @Test
    fun isPalindrome() {
        //Arrange
        val utils = Utils()
        //Act
        val result = utils.isPalindrome("Hello")
        //Assert
        assertEquals(false,result)
    }
```

```
    @Test
    fun isPalindrome_inputString_level_expectedTrue() {
        //Arrange
        val utils= Utils()
        //Act
        val result = utils.isPalindrome("level")
        //Assert
        assertEquals(true,result)
    }
    */


    //Part-2
    lateinit var utils: Utils

    @Before  //it works before every test cases
    fun setUp(){
        utils = Utils()
    }

    @After  //it works after every test cases
    fun tearDown(){
        println("After every test case")
    }

    @Test
    fun isPalindrome() {
        //Act
        val result = utils.isPalindrome("Hello")
        //Assert
        assertEquals(false,result)
    }

    @Test
    fun isPalindrome_inputString_level_expectedTrue() {
        //Act
        val result = utils.isPalindrome("level")
        //Assert
        assertEquals(true,result)
    }

}
```

ParameterizedExample.kt:

```
package com.ghani.parameterizedtesting.utils

import org.junit.Assert.assertEquals
import org.junit.Test
import org.junit.runner.RunWith
import org.junit.runners.Parameterized

//Part-3
@RunWith(value = Parameterized::class) //Its inform JUnit,this is parameterized class
class ParameterizedExample(val input:String,val expectedValue:Boolean) {

    @Test
    fun test(){
        val utils = Utils()
        val result = utils.isPalindrome(input)
        assertEquals(expectedValue,result)
    }

    companion object{
        @JvmStatic
        @Parameterized.Parameters(name = "{index}: {0} is Palindrome - {1}") //Its inform
JUnit,this fun() return parameters
        fun data():List<Array<Any>>{
            return listOf(
                arrayOf("hello",true),
                arrayOf("level",true),
                arrayOf("a",true),
                arrayOf(" ",true)
            )
        }
    }
}
```

PasswordTest.kt:

```
package com.ghani.parameterizedtesting.utils
import org.junit.Assert.assertEquals
import org.junit.Test

class PasswordTest {

    @Test
    fun validate_Password_blankInput_expectedRequiredField(){
        val sut = Utils()
        val result = sut.validatePassword("   ")
        assertEquals("Password is required field",result)
    }

    @Test
    fun validate_Password_2CharInput_expectedValidationMsg(){
        val sut = Utils()
        val result = sut.validatePassword("ab")
        assertEquals("Length of the Password  should be greater than 6",result)
    }

    @Test
    fun validate_Password_CorrectInput_expectedValidPassword(){
        val sut = Utils()
        val result = sut.validatePassword("Pass123")
        assertEquals("Valid",result)
    }
}
```

StringTest.kt:

```
package com.ghani.parameterizedtesting.utils

import org.junit.Assert.assertEquals
import org.junit.Test

class StringTest {

    @Test
    fun testStringReversal_EmptyString_expectedEmptyString(){
        val sut = Utils()
        val result = sut.reverseString("")
        assertEquals("",result)
    }
```

```
    @Test
    fun testStringReversal_SingleChar_expectedSingleChar(){
        val sut = Utils()
        val result = sut.reverseString("a")
        assertEquals("a",result)
    }
    @Test
    fun testStringReversal_ValidInput_expectedSameString(){
        val sut = Utils()
        val result = sut.reverseString("CheezyCode")
        assertEquals("edoCyzeehC",result)
    }
    @Test (expected = IllegalArgumentException::class)
    fun testStringReversal_NullValue_expectedException(){
        val sut = Utils()
        val result = sut.reverseString(null)
    }
}
```

Program2:

Quote.kt:

```
package com.ghani.parameterizedtesting

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

QuoteManager.kt:

```
package com.ghani.parameterizedtesting
import android.content.Context
import com.google.gson.Gson

class QuoteManager {
    var quoteList = emptyArray<Quote>()
    var currentQuoteIndex = 0

    fun populateQuoteFromAssets(context: Context, fileName:String){
        val inputStream = context.assets.open(fileName)
        val size:Int = inputStream.available()
        val buffer = ByteArray(size)
        inputStream.read(buffer)
        inputStream.close()
        val json = String(buffer,Charsets.UTF_16)
        val gson = Gson()
        quoteList =  gson.fromJson(json,Array<Quote>::class.java)
    }
```

```
    fun populateQuotes(quotes: Array<Quote>){
        quoteList = quotes
    }

    fun getCurrentQoute():Quote{
        return quoteList[currentQuoteIndex]
    }

    fun getNextQuote():Quote{
        if (currentQuoteIndex==quoteList.size-1) return quoteList[currentQuoteIndex]
        return quoteList[++currentQuoteIndex]
    }

    fun getPreviousQuote():Quote{
        if (currentQuoteIndex==0) return quoteList[currentQuoteIndex]
        return quoteList[--currentQuoteIndex]
    }

}
```

test/QuoteManagerTest.kt:

```
package com.ghani.parameterizedtesting.utils

import android.content.Context
import android.content.res.AssetManager
import com.ghani.parameterizedtesting.QuoteManager
import org.junit.Assert
import org.junit.Before
import org.junit.Test
import org.mockito.ArgumentMatchers.anyString
import org.mockito.Mock
import org.mockito.Mockito
import org.mockito.Mockito.doReturn
import org.mockito.MockitoAnnotations

class QuoteManagerTest {

    @Mock
    lateinit var context: Context

    @Mock
    lateinit var assetManager:AssetManager
```

```
    @Before
    fun setUp(){
        MockitoAnnotations.openMocks(this)
    }

    @Test
    fun test(){    //Test fails & require debugging
        val testStream = QuoteManagerTest::class.java.getResourceAsStream("/quotes.json")
        doReturn(assetManager).`when`(context).assets
        Mockito.`when`(context.assets.open(anyString())).thenReturn(testStream)

        val sut = QuoteManager()
        sut.populateQuoteFromAssets(context,"")
        val quote = sut.getCurrentQoute()
        Assert.assertEquals("Genius is one percent inspiration and ninety-nine percent
perspiration.",quote.text)
    }
}
```

androidTest/QuoteManagerTest.kt:

```
package com.ghani.parameterizedtesting

import android.app.Application
import android.content.Context
import androidx.test.core.app.ApplicationProvider
import com.google.gson.JsonSyntaxException
import org.junit.Assert.*
import org.junit.After
import org.junit.Before
import org.junit.Test
import java.io.FileNotFoundException
import java.nio.file.FileSystemNotFoundException

class QuoteManagerTest {

    @Test(expected = FileNotFoundException::class) //Assert
    fun populateQuoteFromAssets() {
        val quoteManager = QuoteManager() //Arrange
        val context = ApplicationProvider.getApplicationContext<Context>() //Arrange
        quoteManager.populateQuoteFromAssets(context," ") //Act
    }
```

```
    @Test(expected = JsonSyntaxException::class)
    fun testPopulateQuoteFromAssets() {
        val quoteManager = QuoteManager()
        val context = ApplicationProvider.getApplicationContext<Context>()
        quoteManager.populateQuoteFromAssets(context,"malformed.json")
    }

    @Test //Face Problem??
    fun testPopulateQuoteFromAssets_ValidJSON_expected_Count() {
        val quoteManager = QuoteManager()
        val context = ApplicationProvider.getApplicationContext<Context>()
        quoteManager.populateQuoteFromAssets(context,"quotes.json")
        assertEquals(9,quoteManager.quoteList.size)
    }

    @Test
    fun testPreviousQuote_expected_CorrectQuote() {
        val quoteManager = QuoteManager()
        quoteManager.populateQuotes(arrayOf(
            Quote("This is first quote","1"),
            Quote("This is second quote","2"),
            Quote("This is third quote","3")
        ))
        val quote = quoteManager.getPreviousQuote()
        assertEquals("1",quote.author)
    }

    @Test
    fun testNextQuote_expected_CorrectQuote() {
        val quoteManager = QuoteManager()
        quoteManager.populateQuotes(arrayOf(
            Quote("This is first quote","1"),
            Quote("This is second quote","2"),
            Quote("This is third quote","3")
        ))
        val quote = quoteManager.getNextQuote()
        assertEquals("2",quote.author)
    }

}
```

QuotifyAppUsingViewModed:

MainActivityTest.kt:

```
package com.ghani.quotifyappusingviewmodel

import android.content.Intent
import android.support.test.espresso.Espresso.onView
import android.support.test.espresso.action.ViewActions.click
import android.support.test.espresso.assertion.ViewAssertions.matches
import android.support.test.espresso.intent.Intents
import android.support.test.espresso.intent.matcher.IntentMatchers.hasAction
import android.support.test.espresso.matcher.ViewMatchers.*
import androidx.test.ext.junit.rules.activityScenarioRule
import org.junit.Assert.*
import org.junit.Rule
import org.junit.Test

class MainActivityTest{

   @get:Rule  //It will must be execute at first.
   val activityScenarioRule = activityScenarioRule<MainActivity>()

   @Test
   fun testNextButton_expectedCorrectQuote(){
      onView(withId(R.id.btnNext)).perform(click())
      onView(withId(R.id.btnNext)).perform(click())
      onView(withId(R.id.btnNext)).perform(click())
      onView(withId(R.id.quoteText)).check(matches(withText("Difficulties increase the nearer
we get to the goal.")))
   }

   @Test
   fun testShareButton_expectedIntentChooser(){
      Intents.init()
      //val expected = allOff(hasAction(Intent.ACTION_SEND))
      val expected = allOff()
      onView(withId(R.id.floatingActionButton)).perform(click())
      Intended(expected)
      Intents.release()
   }

}
```

EspressoTesting:

MainActivityTest.kt:

```
package com.ghani.espressotesting

import androidx.test.espresso.Espresso.onView
import androidx.test.espresso.action.ViewActions.*
import androidx.test.espresso.assertion.ViewAssertions.matches
import androidx.test.espresso.matcher.ViewMatchers.withId
import androidx.test.espresso.matcher.ViewMatchers.withText
import androidx.test.ext.junit.rules.activityScenarioRule
import org.junit.Rule
import org.junit.Test

class MainActivityTest{

   @get:Rule
   val activityScenarioRule = activityScenarioRule<MainActivity>()

   @Test
   fun testSubmitButton_expectedCorrectValues(){
      onView(withId(R.id.txt_title)).perform(typeText("Hello"))
      onView(withId(R.id.txt_description)).perform(typeText("CheezyCode",
closeSoftKeyboard()))
      onView(withId(R.id.btn_submit)).perform(click())
      onView(withId(R.id.txt_message)).check(matches(withText("Title - Hello | Desc -
CheezyCode")))
   }
}
```

RoomDBTesting:

Quote.kt:

```
package com.ghani.roomdbtesting

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

@Entity(tableName = "quote")
data class Quote(
   @PrimaryKey(autoGenerate = true)
   val id:Int,
   val text:String,
   val author:String
)
```

QuotesDAO.kt:

```
package com.ghani.roomdbtesting

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

@Dao
interface QuotesDAO {
   @Insert
   suspend fun insertQuote(quote:Quote) //suspend fun execute in background

   @Update
   suspend fun updateQuote(quote:Quote)

   @Query("DELETE FROM quote")
   suspend fun delete()

   @Query("SELECT * FROM quote")
   fun getQuotes():LiveData<List<Quote>> //LiveData execute in background

   @Query("SELECT * FROM quote where id = :quoteId")
   suspend fun getQuoteById(quoteId:Int):Quote
}
```

QuoteDatabase.kt:

```
package com.ghani.roomdbtesting

import androidx.room.*

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

   abstract fun quoteDao():QuotesDAO

}
```

QuotesDaoTest.kt:

```
package com.ghani.roomdbtesting
import androidx.arch.core.executor.testing.InstantTaskExecutorRule
import androidx.room.Room
import androidx.test.core.app.ApplicationProvider
import kotlinx.coroutines.runBlocking
import org.junit.After
import org.junit.Assert
import org.junit.Before
import org.junit.Rule
import org.junit.Test

class QuotesDaoTest {

   //This rule execute all architecture components code synchronously in main thread, no
background thread running.
   @get:Rule
   val instantExecutorRule = InstantTaskExecutorRule()

   lateinit var quoteDatabase: QuoteDatabase
   lateinit var quotesDAO: QuotesDAO

   @Before
   fun setUp() {
      quoteDatabase =
Room.inMemoryDatabaseBuilder(ApplicationProvider.getApplicationContext(),
         QuoteDatabase::class.java)
         .allowMainThreadQueries()
         .build()

      quotesDAO = quoteDatabase.quoteDao()
   }

   @Test
   fun insertQuote_expectedSingleQuote()= runBlocking{
      val quote = Quote(0,"This is test quote","CheezyCode")
      quotesDAO.insertQuote(quote)

      //.getOrAwaitValue() block the thread, until livedata not found.
      val result = quotesDAO.getQuotes().getOrAwaitValue()

     Assert.assertEquals(1,result.size)
     Assert.assertEquals("This is test quote",result[0].text)
   }
```

```
    @Test
    fun deleteQuote_expectedNoResults()= runBlocking{
        val quote = Quote(0,"This is test quote","CheezyCode")
        quotesDAO.insertQuote(quote)
        quotesDAO.delete()

        val result = quotesDAO.getQuotes().getOrAwaitValue()

        Assert.assertEquals(0,result.size)
    }

    @After
    fun tearDown(){
        quoteDatabase.close()
    }
}
```

LiveDataTestUtil.kt:

```
package com.ghani.roomdbtesting
import androidx.lifecycle.LiveData
import androidx.lifecycle.Observer
import java.util.concurrent.CountDownLatch
import java.util.concurrent.TimeUnit
import java.util.concurrent.TimeoutException

class LiveDataTestUtil {

    fun <T> LiveData<T>.getOrAwaitValue( //getOrAwaitValue() is a extension method of livedata
class.
        time: Long = 2,
        timeUnit: TimeUnit = TimeUnit.SECONDS,
        afterObserve: () -> Unit = {}
    ): T {
        var data: T? = null
        val latch = CountDownLatch(1) //Its value (1) & its block the thread.
        val observer = object : Observer<T> {
            override fun onChanged(o: T?) { //onChanged() call,when livedata find.
                data = o
                latch.countDown() //When find livedata ,latch value become(0) & its unblock the
thread.
                this@getOrAwaitValue.removeObserver(this)
            }
        }
```

```
        this.observeForever(observer) //Set observer for forever.

        afterObserve.invoke()

        // Don't wait indefinitely if the LiveData is not set.
        // Here, its wait 2 minutes.
        if (!latch.await(time, timeUnit)) {
            this.removeObserver(observer)
            throw TimeoutException("LiveData value was never set.")
        }

        @Suppress("UNCHECKED_CAST")
        return data as T
    }
}
```

MockUnitTesting:

User.kt:

```
package com.ghani.mockunittesting.mock

data class User(
    val id:Int,
    val name:String,
    val email:String,
    val password:String
)

enum class LOGIN_STATUS{
    INVALID_USER,
    INVALID_PASSWORD,
    UNKNOWN_ERROR,
    SUCCESS
}
```

UserRepository.kt:

```
package com.ghani.mockunittesting.mock

class UserRepository {

    val users = listOf<User>(
        User(1,"Rony","rony@gmail.com","slfsr342323"),
        User(2,"Jony","jonyl@gmail.com","nvxcvxc23534535"),
        User(3,"Tony","tony@gmail.com","jkdfgd7897353"))
```

```
  fun loginUser(email:String,password:String):LOGIN_STATUS{
    val users = users.filter { user -> user.email == email }
    return if (users.size == 1){
      if (users[0].password == password){
        LOGIN_STATUS.SUCCESS
      }
      else{
        LOGIN_STATUS.INVALID_PASSWORD
      }
    }
    else{
      LOGIN_STATUS.INVALID_USER
    }
  }
}
```

UserService.kt:

```
package com.ghani.mockunittesting.mock

import com.ghani.mockunittesting.mock.LOGIN_STATUS.*

class UserService (private val userRepository: UserRepository) {

  fun loginUser(email:String,password:String):String{
    val status = userRepository.loginUser(email, password)
    return when(status){
      INVALID_USER -> "User does not exit"
      INVALID_PASSWORD -> "Password is invalid"
      UNKNOWN_ERROR -> "Unknown error occurred"
      SUCCESS -> "Logged in successfully"
    }
  }
}
```

UserServiceTest.kt:

```
package com.ghani.mockunittesting

import com.ghani.mockunittesting.mock.LOGIN_STATUS
import com.ghani.mockunittesting.mock.UserRepository
import com.ghani.mockunittesting.mock.UserService
import org.junit.Assert
import org.junit.Before
```

```
import org.junit.Test
import org.mockito.ArgumentMatchers.anyString
import org.mockito.Mock
import org.mockito.Mockito
import org.mockito.MockitoAnnotations

class UserServiceTest {

   @Mock
   lateinit var userRepository: UserRepository

   @Before
   fun setUp(){
      MockitoAnnotations.openMocks(this)
Mockito.`when`(userRepository.loginUser(anyString(),anyString())).thenReturn(LOGIN_STATUS.
SUCCESS)
   }

   @Test
   fun testUserService(){
      val sut = UserService(userRepository)
      val status = sut.loginUser("abc@gmail.com","asghff090")
      Assert.assertEquals("Logged in successfully",status)
   }
}
```

CoroutineTesting:

MainCoroutineRule.kt:

```
package com.ghani.coroutinetesting
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.test.StandardTestDispatcher
import kotlinx.coroutines.test.resetMain
import kotlinx.coroutines.test.setMain
import org.junit.rules.TestWatcher
import org.junit.runner.Description

class MainCoroutineRule:TestWatcher() {

   val testDispatcher = StandardTestDispatcher()

   override fun starting(description: Description) {
      super.starting(description)
      Dispatchers.setMain(testDispatcher)
   }
```

```
    override fun finished(description: Description) {
        super.finished(description)
        Dispatchers.resetMain()
    }
}
```

Util.kt:

```
package com.ghani.coroutinetesting

import kotlinx.coroutines.*

//In case of testing, pass testDispatcher in val dispatcher.
//In case of application, pass Dispatchers.Main/Dispatchers.IO/others in val dispatcher,
class Util (val dispatcher: CoroutineDispatcher) {

    suspend fun getUserName():String{
        delay(10000)
        return "CheezeCode"
    }

    suspend fun getUser():String{
        CoroutineScope(Dispatchers.Main).launch{
            delay(2000)
        }
        return "User - CheezeCode"
    }

    suspend fun getAddress():String{
        withContext(dispatcher){
            delay(2000)
        }
        return "Address"
    }

    var globalArg = false
    suspend fun getAddressDetail(){
        CoroutineScope(dispatcher).launch{
            globalArg = true
        }
    }

}
```

UtilTest.kt:

```
package com.ghani.coroutinetesting

import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.test.StandardTestDispatcher
import kotlinx.coroutines.test.resetMain
import kotlinx.coroutines.test.runTest
import kotlinx.coroutines.test.setMain
import org.junit.*

class UtilTest {

    /*
    @Test
    fun testGetUserName() {
        val sut = Util()

        //runTest{} best for testing suspend fun() or coroutineScope over runBlocking{}.Because it
avoid delay().
        runTest{
            sut.getUserName()
        }
    }
    */

    /*
    //It test CoroutineScope(Dispatchers.Main) in one thread,
    //But unable to test withContext(Dispatchers.IO).
    private val testDispatcher = StandardTestDispatcher()

    @Before
    fun setUp(){
        Dispatchers.setMain(testDispatcher)
    }

    @Test
    fun testGetUser() {
        val sut = Util()

        runTest{
            sut.getUser()
        }
    }
```

```
@After
fun tearDown(){
   Dispatchers.resetMain()
}
*/

/*
//It test all the dispatchers in one thread(include withContext(Dispatchers.IO)).
private val testDispatcher = StandardTestDispatcher()

@Before
fun setUp(){
   Dispatchers.setMain(testDispatcher)
}

@Test
fun testGetAddress(){
   val sut = Util(testDispatcher)

   runTest{
      sut.getAddress()
   }
}

@After
fun tearDown(){
   Dispatchers.resetMain()
}
*/

/*
@get:Rule
val mainCoroutineRule = MainCoroutineRule()

@Test
fun testGetAddress(){
   val sut = Util(mainCoroutineRule.testDispatcher)

   runTest{
      sut.getAddress()
   }
}
*/
```

```
    @get:Rule
    val mainCoroutineRule = MainCoroutineRule()

    @Test
    fun testGetAddressDetail(){
        val sut = Util(mainCoroutineRule.testDispatcher)

        runTest{
            sut.getAddressDetail()
            //advanceUntilIdle() execute all the coroutines that store in testDispatcher's scheduler
queue.
            mainCoroutineRule.testDispatcher.scheduler.advanceUntilIdle()
            Assert.assertEquals(true,sut.globalArg)
        }
    }

}
```

ViewModelTesting:
MainViewModelTest.kt

```
package com.ghani.viewmodeltesting.viewmodels

import androidx.arch.core.executor.testing.InstantTaskExecutorRule
import com.ghani.viewmodeltesting.repository.ProductRepository
import com.ghani.viewmodeltesting.utils.NetworkResult
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.test.StandardTestDispatcher
import kotlinx.coroutines.test.resetMain
import kotlinx.coroutines.test.runTest
import kotlinx.coroutines.test.setMain
import org.junit.*
import org.mockito.Mock
import org.mockito.Mockito
import org.mockito.MockitoAnnotations

class MainViewModelTest {

    private val testDispatcher = StandardTestDispatcher()

    @get:Rule
    val instantTaskExecutorRule = InstantTaskExecutorRule()
```

```
    @Mock
    lateinit var repository: ProductRepository

    @Before
    fun setUp() {
        MockitoAnnotations.openMocks(this)
        Dispatchers.setMain(testDispatcher)
    }

    @Test
    fun test_GetProducts() = runTest{
        Mockito.`when`(repository.getProducts()).thenReturn(NetworkResult.Success(emptyList()))
        val sut = MainViewModel(repository)
        sut.getProducts()
        testDispatcher.scheduler.advanceUntilIdle()
        val result = sut.products.getOrAwaitValue()
        Assert.assertEquals(0,result.data!!.size())
    }

    @Test
    fun test_GetProducts_expectedError() = runTest{
        Mockito.`when`(repository.getProducts()).thenReturn(NetworkResult.Error("Something
went wrong"))
        val sut = MainViewModel(repository)
        sut.getProducts()
        testDispatcher.scheduler.advanceUntilIdle()
        val result = sut.products.getOrAwaitValue()
        Assert.assertEquals(true, result is NetworkResult.Error<*>)
        Assert.assertEquals("Something went wrong",result.message)
    }

    @After
    fun tearDown() {
        Dispatchers.resetMain()
    }
}
```

MainViewModel.kt:

```
package com.ghani.viewmodeltesting.viewmodels
import androidx.lifecycle.LiveData
import androidx.lifecycle.MutableLiveData
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.ghani.viewmodeltesting.models.ProductListItem
import com.ghani.viewmodeltesting.repository.ProductRepository
import com.ghani.viewmodeltesting.utils.NetworkResult
import kotlinx.coroutines.launch

class MainViewModel(private val repository: ProductRepository):ViewModel() {
    private val _products = MutableLiveData<NetworkResult<List<ProductListItem>>>()
    val products: LiveData<NetworkResult<List<ProductListItem>>>
    get() = _products

    fun getProducts(){
        viewModelScope.launch {
            val result = repository.getProducts()
            _products.postValue(result)
        }
    }
}
```

ProductRepository.kt:

```
package com.ghani.viewmodeltesting.repository
import com.ghani.viewmodeltesting.api.ProductsAPI
import com.ghani.viewmodeltesting.models.ProductListItem
import com.ghani.viewmodeltesting.utils.NetworkResult

class ProductRepository (private val productsAPI: ProductsAPI){
    suspend fun getProducts(): NetworkResult<List<ProductListItem>> {
        val response = productsAPI.getProducts()
        return if (response.isSuccessful){
            val responseBody = response.body()
            if (responseBody != null){
                NetworkResult.Success(responseBody)
            }else{
                NetworkResult.Error("Something went wrong")
            }
        }else{
            NetworkResult.Error("Something went wrong")
        }
    }
}
```

NetworkResult.kt:

```
package com.ghani.viewmodeltesting.utils

sealed class NetworkResult<T>(val data:T? = null, val message:String? = null) {

   class Success<T>(data: T):NetworkResult<T>(data)
   class Error<T>(message:String?,data:T? = null):NetworkResult<T>(data,message)
   class Loading<T>:NetworkResult<T>()

}
```

LiveDataTestUtil.kt:

```
package com.ghani.viewmodeltesting

import androidx.lifecycle.LiveData
import androidx.lifecycle.Observer
import java.util.concurrent.CountDownLatch
import java.util.concurrent.TimeUnit
import java.util.concurrent.TimeoutException

class LiveDataTestUtil {

   fun <T> LiveData<T>.getOrAwaitValue( //getOrAwaitValue() is a extension method of livedata
class.
      time: Long = 2,
      timeUnit: TimeUnit = TimeUnit.SECONDS,
      afterObserve: () -> Unit = {}
   ): T {
      var data: T? = null
      val latch = CountDownLatch(1) //Its value (1) & its block the thread.
      val observer = object : Observer<T> {
         override fun onChanged(o: T?) { //onChanged() call,when livedata find.
            data = o
            latch.countDown() //When find livedata ,latch value become(0) & its unblock the
thread.
            this@getOrAwaitValue.removeObserver(this)
         }
      }
      this.observeForever(observer) //Set observer for forever.

      afterObserve.invoke()
```

```
        // Don't wait indefinitely if the LiveData is not set.
        // Here, its wait 2 minutes.
        if (!latch.await(time, timeUnit)) {
            this.removeObserver(observer)
            throw TimeoutException("LiveData value was never set.")
        }

        @Suppress("UNCHECKED_CAST")
        return data as T
    }
}
```

RepositoryTesting:

ProductRepositoryTest.kt:

```
package com.ghani.viewmodeltesting.repository

import com.ghani.viewmodeltesting.api.ProductsAPI
import com.ghani.viewmodeltesting.models.ProductListItem
import com.ghani.viewmodeltesting.utils.NetworkResult
import kotlinx.coroutines.test.runTest
import okhttp3.ResponseBody.Companion.toResponseBody
import org.junit.Assert.*
import org.junit.Before
import org.junit.Test
import org.mockito.Mock
import org.mockito.Mockito
import org.mockito.MockitoAnnotations
import retrofit2.Response

class ProductRepositoryTest {

    @Mock
    lateinit var productsAPI: ProductsAPI

    @Before
    fun setUp() {
        MockitoAnnotations.openMocks(this)
    }
```

```
    //We use runTest{},because getProducts() is suspend fun.
    @Test
    fun test_GetProducts_EmptyList() = runTest{
        Mockito.`when`(productsAPI.getProducts()).thenReturn(Response.success(emptyList()))
        val sut = ProductRepository(productsAPI)
        val result = sut.getProducts()
        assertEquals(true,result is NetworkResult.Success)
        assertEquals(0,result.data!!.size)
    }

    @Test
    fun test_GetProducts_ProductsList() = runTest{
        val productList = listOf<ProductListItem>(
            ProductListItem("","",1,"",40.3,"Product-1"),
            ProductListItem("","",2,"",20.2,"Product-2")
        )
        Mockito.`when`(productsAPI.getProducts()).thenReturn(Response.success(productList))
        val sut = ProductRepository(productsAPI)
        val result = sut.getProducts()
        assertEquals(true,result is NetworkResult.Success)
        assertEquals(2,result.data!!.size)
        assertEquals("Product-1",result.data!![0].title)
    }

    @Test
    fun test_GetProducts_expectedError() = runTest{

Mockito.`when`(productsAPI.getProducts()).thenReturn(Response.error(401,"Unauthorized".to
ResponseBody()))
        val sut = ProductRepository(productsAPI)
        val result = sut.getProducts()
        assertEquals(true,result is NetworkResult.Error)
        assertEquals("Something went wrong",result.message)
    }

}
```

ProductRepository.kt:

```
package com.ghani.viewmodeltesting.repository

import com.ghani.viewmodeltesting.api.ProductsAPI
import com.ghani.viewmodeltesting.models.ProductListItem
import com.ghani.viewmodeltesting.utils.NetworkResult

class ProductRepository (private val productsAPI: ProductsAPI){

    suspend fun getProducts(): NetworkResult<List<ProductListItem>> {
        val response = productsAPI.getProducts()
        return if (response.isSuccessful){
            val responseBody = response.body()
            if (responseBody != null){
                NetworkResult.Success(responseBody)
            }else{
                NetworkResult.Error("Something went wrong")
            }
        }else{
            NetworkResult.Error("Something went wrong")
        }
    }
}
```

ProductsAPI.kt:

```
package com.ghani.viewmodeltesting.api

import com.ghani.viewmodeltesting.models.ProductListItem
import retrofit2.Response
import retrofit2.http.GET


interface ProductsAPI {

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

}
```

ProductListItem.kt:

```
package com.ghani.viewmodeltesting.models

data class ProductListItem(
   val category: String,
   val description: String,
   val id: Int,
   val image: String,
   val price: Double,
   val title: String
)
```

NetworkResult.kt:

```
package com.ghani.viewmodeltesting.utils

sealed class NetworkResult<T>(val data:T? = null, val message:String? = null) {

   class Success<T>(data: T):NetworkResult<T>(data)
   class Error<T>(message:String?,data:T? = null):NetworkResult<T>(data,message)
   class Loading<T>:NetworkResult<T>()

}
```

RetrofitCallTesting:

Program 1:

ProductsAPITest.kt:

```
package com.ghani.viewmodeltesting

import com.ghani.viewmodeltesting.api.ProductsAPI
import kotlinx.coroutines.test.runTest
import okhttp3.mockwebserver.MockResponse
import okhttp3.mockwebserver.MockWebServer
import org.junit.After
import org.junit.Assert
import org.junit.Before
import org.junit.Test
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
```

```
class ProductsAPITest {
    //Declare mockWebServer
    lateinit var mockWebServer: MockWebServer
    lateinit var productsAPI: ProductsAPI

    @Before
    fun setUp(){
        //Initialize mockWebServer
        mockWebServer = MockWebServer()

        //productsAPI use mockWebServer
        productsAPI = Retrofit.Builder()
            .baseUrl(mockWebServer.url("/"))
            .addConverterFactory(GsonConverterFactory.create())
            .build().create(ProductsAPI::class.java)
    }

    @Test
    fun testGetProducts() = runTest{
        val mockResponse = MockResponse() //Declare & Initialize mockResponse
        mockResponse.setBody("[]") //Set mockResponse by empty array
        mockWebServer.enqueue(mockResponse) //mockWebServer -> (receive request & send
        mockResponse)

        val response = productsAPI.getProducts() //Request send in mockWebserver
        mockWebServer.takeRequest() //mockWebserver take request

        Assert.assertEquals(true,response.body()!!.isEmpty())
    }

    @Test
    fun testGetProducts_returnProducts() = runTest{
        val mockResponse = MockResponse()
        val content = Helper.readFileResource("/response.json")
        mockResponse.setResponseCode(200)
        mockResponse.setBody(content)
        mockWebServer.enqueue(mockResponse)

        val response = productsAPI.getProducts()
        mockWebServer.takeRequest()

        Assert.assertEquals(false,response.body()!!.isEmpty())
        Assert.assertEquals(5,response.body()!!.size)
    }
```

```
    @Test
    fun testGetProducts_returnError() = runTest{
        val mockResponse = MockResponse()
        mockResponse.setResponseCode(404) //404 means wrong.
        mockResponse.setBody("Something went wrong")
        mockWebServer.enqueue(mockResponse)

        val response = productsAPI.getProducts()
        mockWebServer.takeRequest()

        //It depends on ResponseCode.
        Assert.assertEquals(false,response.isSuccessful)
        Assert.assertEquals(404,response.code())
    }

    @After
    fun tearDown(){
        mockWebServer.shutdown()
    }
}
```

ProductsAPI.kt:

```
package com.ghani.viewmodeltesting.api

import com.ghani.viewmodeltesting.models.ProductListItem
import retrofit2.Response
import retrofit2.http.GET


interface ProductsAPI {

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

}
```

Helper.kt:

```
package com.ghani.viewmodeltesting

import java.io.InputStreamReader

object Helper {
   fun readFileResource(fileName:String):String{
      val inputStream = Helper::class.java.getResourceAsStream(fileName)
      val builder = StringBuilder()
      val reader = InputStreamReader(inputStream,"UTF-8")
      reader.readLines().forEach {
         builder.append(it)
      }
      return builder.toString()
   }
}
```

response.json:

```
[
 {
  "text": "Genius is one percent inspiration and ninety-nine percent perspiration.",
  "author": "Thomas Edison"
 },
 {
  "text": "You can observe a lot just by watching.",
  "author": "Yogi Berra"
 },
 {
  "text": "A house divided against itself cannot stand.",
  "author": "Abraham Lincoln"
 },
 {
  "text": "Difficulties increase the nearer we get to the goal.",
  "author": "Johann Wolfgang von Goethe"
 },
 {
  "text": "Fate is in your hands and no one elses",
  "author": "Byron Pulsifer"
 }
]
```

Program 2:
ProductsAPITest2.kt:

```
package com.ghani.viewmodeltesting.repository

import com.ghani.viewmodeltesting.api.ProductsAPI
import com.ghani.viewmodeltesting.utils.NetworkResult
import kotlinx.coroutines.test.runTest
import okhttp3.mockwebserver.MockResponse
import okhttp3.mockwebserver.MockWebServer
import org.junit.After
import org.junit.Assert
import org.junit.Before
import org.junit.Test
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory

class ProductsAPITest2 {

    //Declare mockWebServer
    lateinit var mockWebServer: MockWebServer
    lateinit var productsAPI: ProductsAPI

    @Before
    fun setUp() {
        //Initialize mockWebServer
        mockWebServer = MockWebServer()

        //productsAPI use mockWebServer
        productsAPI = Retrofit.Builder()
            .baseUrl(mockWebServer.url("/"))
            .addConverterFactory(GsonConverterFactory.create())
            .build().create(ProductsAPI::class.java)
    }

    @Test
    fun testGetProducts_EmptyList() = runTest {
        val mockResponse = MockResponse()
        mockWebServer.enqueue(mockResponse.setResponseCode(404))
        val sut = ProductRepository(productsAPI)
        val result= sut.getProducts()
        val request = mockWebServer.takeRequest()
        //It depends on ResponseCode.
        Assert.assertEquals(false, result is NetworkResult.Success)
    }
```

```
    @After
    fun tearDown() {
        mockWebServer.shutdown()
    }
}
```

ProductsAPI.kt:

```
package com.ghani.viewmodeltesting.api

import com.ghani.viewmodeltesting.models.ProductListItem
import retrofit2.Response
import retrofit2.http.GET

interface ProductsAPI {

    @GET("/products")
    suspend fun getProducts(): Response<List<ProductListItem>>
}
```

ProductRepository.kt:

```
package com.ghani.viewmodeltesting.repository
import com.ghani.viewmodeltesting.api.ProductsAPI
import com.ghani.viewmodeltesting.models.ProductListItem
import com.ghani.viewmodeltesting.utils.NetworkResult

class ProductRepository (private val productsAPI: ProductsAPI){
    suspend fun getProducts(): NetworkResult<List<ProductListItem>> {
        val response = productsAPI.getProducts()
        return if (response.isSuccessful){
            val responseBody = response.body()
            if (responseBody != null){
                NetworkResult.Success(responseBody)
            }else{
                NetworkResult.Error("Something went wrong")
            }
        }else{
            NetworkResult.Error("Something went wrong")
        }
    }
}
```

NetworkResult.kt:

```
package com.ghani.viewmodeltesting.utils

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

HiltTesting:

CustomRunner.kt:

```
package com.ghani.dagger2_mvvm

import android.app.Application
import android.content.Context
import androidx.test.runner.AndroidJUnitRunner
import dagger.hilt.android.testing.HiltTestApplication

//Step-1
class CustomRunner: AndroidJUnitRunner() {
   override fun newApplication(
      cl: ClassLoader?,
      className: String?,
      context: Context?
   ): Application {
      return super.newApplication(cl, HiltTestApplication::class.java.name, context)
   }
}
```

TestDatabaseModule.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.android.qualifiers.ApplicationContext
import dagger.hilt.components.SingletonComponent
import dagger.hilt.testing.TestInstallIn
import javax.inject.Singleton
```

```
//Step-2
@TestInstallIn(components = [SingletonComponent::class], replaces = [DatabaseModule::class])
@Module
class TestDatabaseModule {

   @Singleton
   @Provides
   fun provideTestDB(@ApplicationContext context: Context): FakerDB {
      return Room.inMemoryDatabaseBuilder(
         context,
         FakerDB::class.java
      ).allowMainThreadQueries().build()
   }
}
```

FakerDAOTest.kt:

```
package com.ghani.dagger2_mvvm
import androidx.arch.core.executor.testing.InstantTaskExecutorRule
import com.ghani.dagger2_mvvm.db.FakerDAO
import com.ghani.dagger2_mvvm.db.FakerDB
import com.ghani.dagger2_mvvm.models.Product
import dagger.hilt.android.testing.HiltAndroidRule
import dagger.hilt.android.testing.HiltAndroidTest
import kotlinx.coroutines.test.runTest
import org.junit.After
import org.junit.Assert
import org.junit.Before
import org.junit.Rule
import org.junit.Test
import javax.inject.Inject

//Step-3
@HiltAndroidTest
class FakerDAOTest {

   //This rule execute all architecture components code synchronously in main thread, no
background thread running.
   @get:Rule
   val instantExecutorRule = InstantTaskExecutorRule()

   //Step-4
   @get:Rule
   val hiltAndroidRule = HiltAndroidRule(this)
```

```
    //Step-5
    @Inject
    lateinit var fakerDatabase: FakerDB
    private lateinit var fakerDAO: FakerDAO

    @Before
    fun setUp() {
        //Step-6
        hiltAndroidRule.inject()
        fakerDAO = fakerDatabase.getFakerDAO()
    }

    @Test
    fun insertProduct_returnSingleProduct()= runTest {
        val product = Product("", "", 1, "", 12.33,"Test Product")
        fakerDAO.addProducts(listOf(product))
        val result = fakerDAO.getProducts()
        Assert.assertEquals(1,result.size)
    }

    @After
    fun closeDatabase(){
        fakerDatabase.close()
    }
}
```

FlowsTesting:

Program 1:

FlowDemo.kt:

```
package com.ghani.roomdbtesting

import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.flow

class FlowDemo {
    fun getFlow() = flow{
        emit(1)
        delay(2000)
        emit(2)
        delay(2000)
    }
}
```

FlowDemoTest.kt:

```
package com.ghani.roomdbtesting

import kotlinx.coroutines.flow.toList
import kotlinx.coroutines.test.runTest
import org.junit.Assert

import org.junit.Test

class FlowDemoTest {

    @Test
    fun getFlowTest() = runTest{
        val sut = FlowDemo()
        val result = sut.getFlow().toList()
        Assert.assertEquals(listOf(1,2),result)
    }
}
```

Program 2:

QuotesDAO.kt:

```
package com.ghani.roomdbtesting

import androidx.room.*
import kotlinx.coroutines.flow.Flow

@Dao
interface QuotesDAO {
    @Insert
    suspend fun insertQuote(quote:Quote) //suspend fun execute in background

    @Update
    suspend fun updateQuote(quote:Quote)

    @Query("DELETE FROM quote")
    suspend fun delete()

    @Query("SELECT * FROM quote")
    fun getQuotes(): Flow<List<Quote>> //LiveData execute in background

    @Query("SELECT * FROM quote where id = :quoteId")
    suspend fun getQuoteById(quoteId:Int):Quote
}
```

QuotesDAOFlowTest.kt:

```
package com.ghani.roomdbtesting
import androidx.arch.core.executor.testing.InstantTaskExecutorRule
import androidx.room.Room
import androidx.test.core.app.ApplicationProvider
import app.cash.turbine.test
import kotlinx.coroutines.delay
import kotlinx.coroutines.flow.first
import kotlinx.coroutines.launch
import kotlinx.coroutines.runBlocking
import org.junit.After
import org.junit.Assert
import org.junit.Before
import org.junit.Rule
import org.junit.Test

class QuotesDAOFlowTest {
   //This rule execute all architecture components code synchronously in main thread,
   no background thread running.
   @get:Rule
   val instantExecutorRule = InstantTaskExecutorRule()

   lateinit var quoteDatabase: QuoteDatabase
   lateinit var quotesDAO: QuotesDAO

   @Before
   fun setUp() {
      quoteDatabase =
Room.inMemoryDatabaseBuilder(ApplicationProvider.getApplicationContext(),
         QuoteDatabase::class.java)
         .allowMainThreadQueries()
         .build()

      quotesDAO = quoteDatabase.quoteDao()
   }

   @Test
   fun insertQuote_expectedSingleQuote1()= runBlocking{
      val quote = Quote(0,"This is test quote","CheezyCode")
      quotesDAO.insertQuote(quote)
      val result = quotesDAO.getQuotes().first()

     Assert.assertEquals(1,result.size)
     Assert.assertEquals("This is test quote",result[0].text)
   }
```

```
    @Test
    fun insertQuote_expectedSingleQuote2()= runBlocking{
        val quote = Quote(0,"This is test quote","CheezyCode")
        quotesDAO.insertQuote(quote)
        //We use .test{} over .toList(),because .toList() wait infinite time.
        val result = quotesDAO.getQuotes().test{
            val quoteList = awaitItem()
            Assert.assertEquals(1,quoteList.size)
            cancel()
        }
    }
    @Test
    fun insertQuote_expectedSingleQuote3()= runBlocking{
        val quote1 = Quote(1,"This is test quote1","CheezyCode")
        val quote2 = Quote(2,"This is test quote2","CheezyCode")
        quotesDAO.insertQuote(quote1)
        quotesDAO.insertQuote(quote2)
        val result = quotesDAO.getQuotes().test{
            val quoteList = awaitItem()
            Assert.assertEquals(2,quoteList.size)
            cancel()
        }
    }
    @Test
    fun insertQuote_expectedSingleQuote4()= runBlocking{
        val quote1 = Quote(1,"This is test quote1","CheezyCode")
        val quote2 = Quote(2,"This is test quote2","CheezyCode")
        quotesDAO.insertQuote(quote1)
        launch{
            delay(500)
            quotesDAO.insertQuote(quote2)
        }
        val result = quotesDAO.getQuotes().test{
            val quoteList1 = awaitItem()
            Assert.assertEquals(1,quoteList1.size)
            val quoteList2 = awaitItem()
            Assert.assertEquals(2,quoteList2.size)
            cancel()
        }
    }
    @After
    fun tearDown(){
        quoteDatabase.close()
    }
}
```