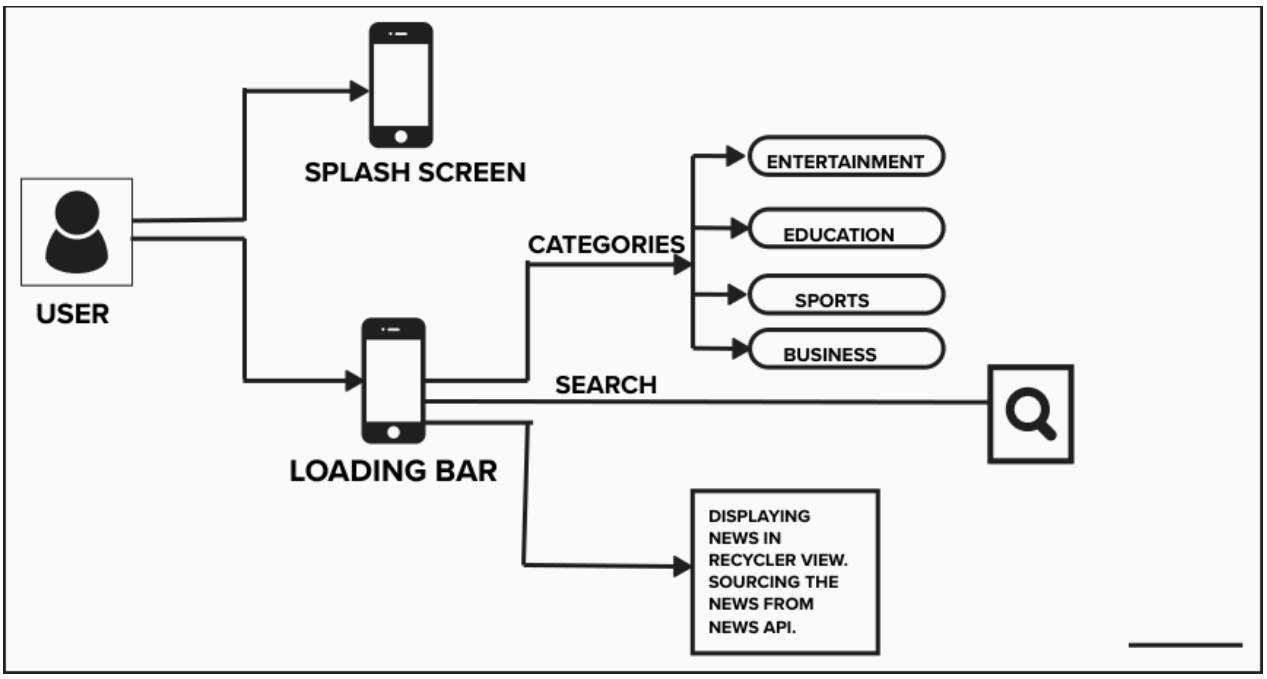
An Android Application for Keeping Up with the Latest Headlines :

In our fast-paced digital age, staying informed about the latest news and headlines has never been more important. With the advent of smartphones and the widespread availability of the internet, people have the world's news at their fingertips. An Android application designed to keep up with the latest headlines is a valuable tool for individuals seeking to stay updated on current events, trends, and breaking news stories. This application serves as a gateway to the vast and ever-changing landscape of news and information, making it easier for users to access, customize, and consume news content according to their interests and preferences.

In this digital era, news applications offer a convenient and user-friendly way to access news articles, videos, and other forms of media from various reputable sources. They provide users with the ability to personalize their news consumption, choosing the topics and sources that matter most to them. Whether it's global affairs, technology, sports, entertainment, or local news, such an Android application allows users to create a tailored news experience that aligns with their interests.

This introduction serves as a stepping stone to explore the world of Android applications for keeping up with the latest headlines. We will delve into the features, benefits, and importance of such applications, as well as the role they play in today's information landscape. Furthermore, we will discuss how these applications have become an indispensable tool for individuals seeking to stay informed, and we will highlight the impact they have on the way we consume news in the modern world.

Architecture



Learning Outcomes :

By end of this project:

- You'll be able to work on Android studio and build an app.
- You'll be able to integrate the database accordingly.
- You'll be able to integrate the API's accordingly.

Project Workflow:

- Users register into the application.
- After registration , user logins into the application.
- User enters into the main page

Tasks:

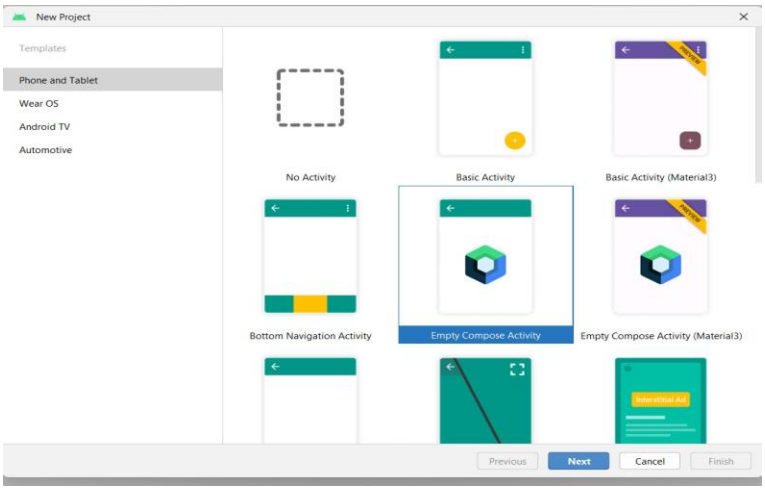
1.Required initial steps
2.Creating a new project.
3.Adding required dependencies.
4.Adding permissions
5.Creating the database classes.
6.Creating API Service and required classes for integrating API
7.Building application UI and connecting to database.
8.Modifying AndroidManifest.xml
9.Running the application.

Task 1:

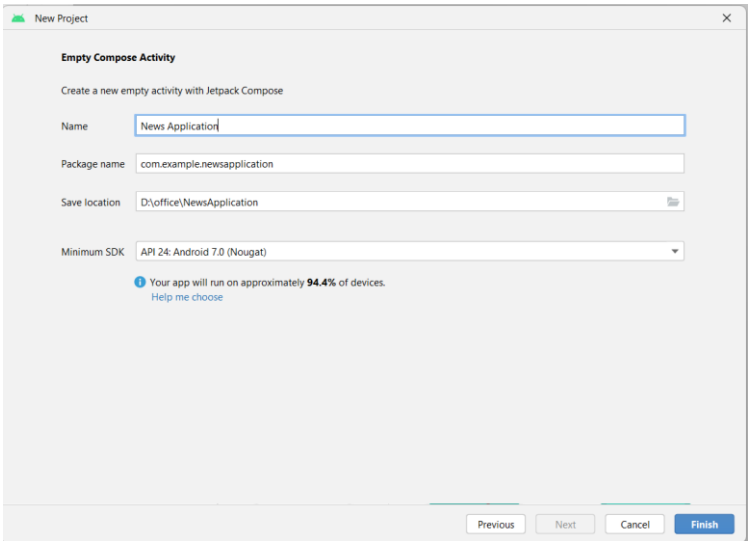
Required initial steps :

Task 2 :

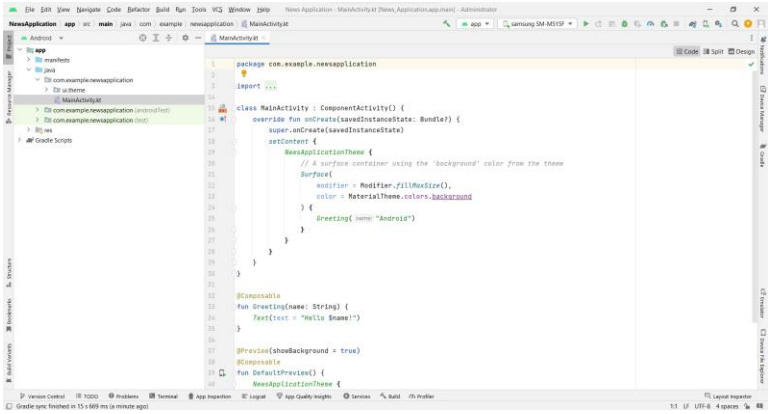
Creating a new project.
Step 1 : Android studio > File > New > New Project > Empty Compose Activity
Step 2 : Click on **Next** button.



Step 3 : Give name to the new project.
Step 4 : Give the Minimum SDK value
Step 5 : Click Finish



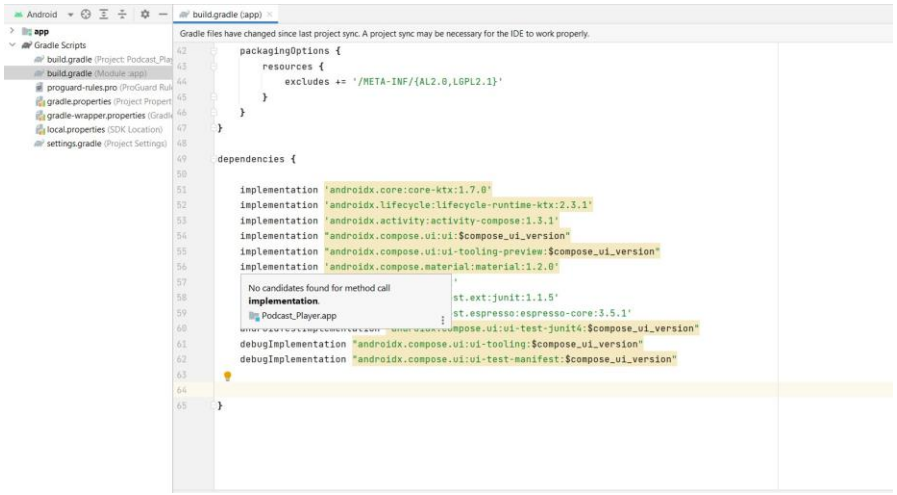
Main activity file



Task 3 :

Adding required dependencies.

Step 1 : Gradle scripts > build.gradle(Module :app)



Step 2 : Adding room dependencies.
Add the below code in dependencies

```
// Room Database
implementation 'androidx.room:room-common:2.5.0'
implementation 'androidx.room:room-ktx:2.5.0'
```

Step 3 : Adding Retrofit dependencies

```
// Retrofit
implementation 'com.squareup.retrofit2:retrofit:2.9.0'
implementation "com.squareup.okhttp3:okhttp:5.0.0-alpha.2"
implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
```

Step 4 : Adding Coil dependencies

```
implementation("io.coil-kt:coil-compose:1.4.0")
```

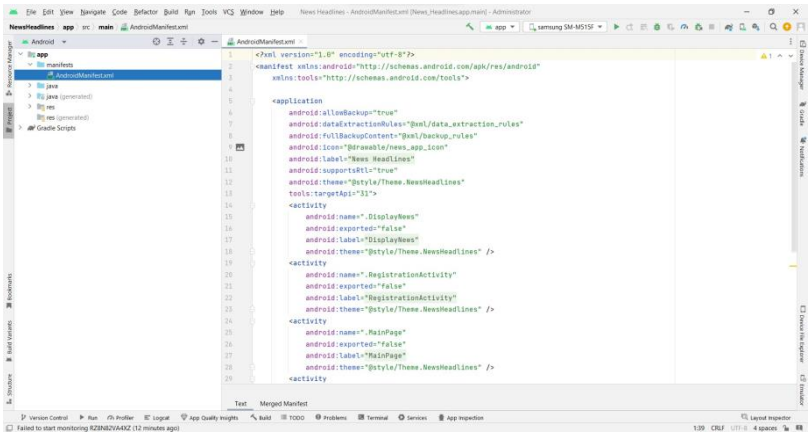
```
dependencies {

    implementation 'androidx.core:core-ktx:1.7.0'
    implementation 'androidx.lifecycle:lifecycle-runtime-ktx:2.3.1'
    implementation 'androidx.activity:activity-compose:1.3.1'
    implementation "androidx.compose.ui:ui:$compose_ui_version"
    implementation "androidx.compose.ui:ui-tooling-preview:$compose_ui_version"
    implementation 'androidx.compose.material:material:1.2.0'

    testImplementation 'junit:junit:4.13.2'
    androidTestImplementation 'androidx.test.ext:junit:1.1.5'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.5.1'
    androidTestImplementation "androidx.compose.ui:ui-test-junit4:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-tooling:$compose_ui_version"
    debugImplementation "androidx.compose.ui:ui-test-manifest:$compose_ui_version"


    // Room Database
    implementation 'androidx.room:room-common:2.5.0'
    implementation 'androidx.room:room-ktx:2.5.0'

    // Retrofit
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation "com.squareup.okhttp3:okhttp:5.0.0-alpha.2"
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'

    implementation("io.coil-kt:coil-compose:1.4.0")
}
```

Step 5 : Click on Sync now

Task 4:

Adding permissions

Step 1: Open AndroidManifest.xml.



Step 2: Add permission to access wifi and internet

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

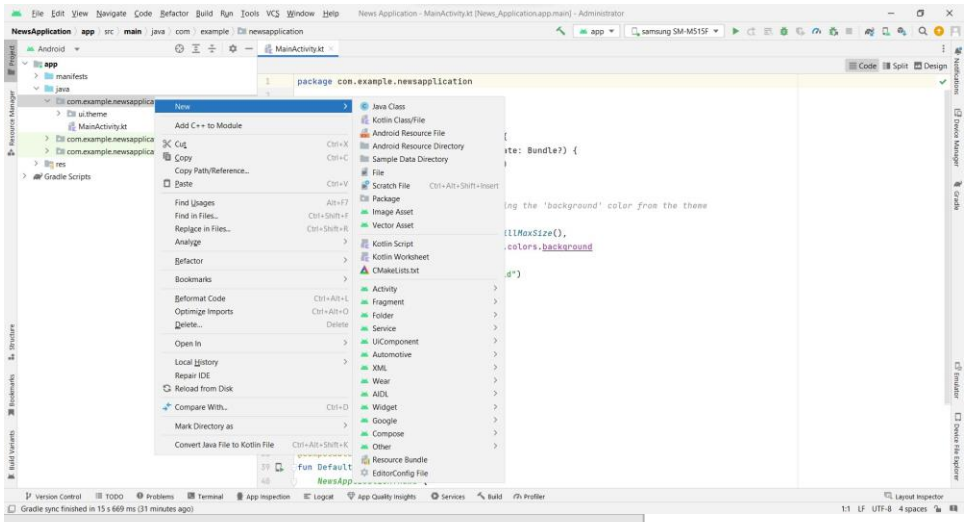
```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools">
<!--    permissions-->
    <uses-permission android:name="android.permission.INTERNET"/>
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE"/>

    <application
        android:allowBackup="true"
        android:dataExtractionRules="@xml/data_extraction_rules"
        android:fullBackupContent="@xml/backup_rules"
        android:icon="@drawable/news_app_icon"
        android:label="News Headlines"
        android:supportsRtl="true"
        android:theme="@style/Theme.NewsHeadlines"
```

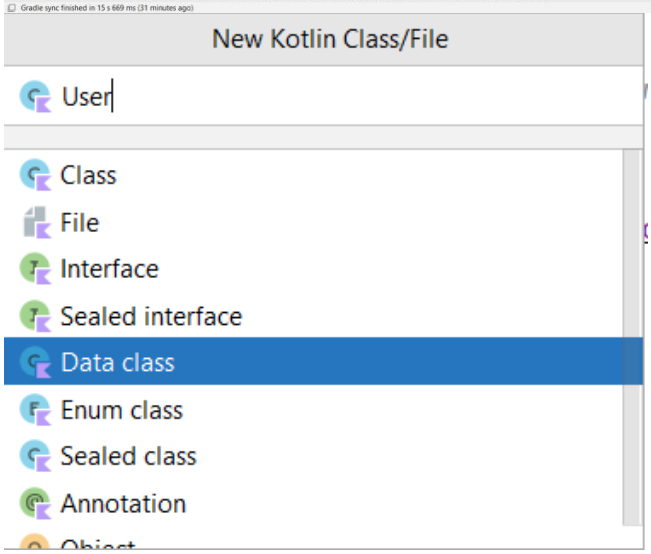
Task 5:

Creating the database classes.

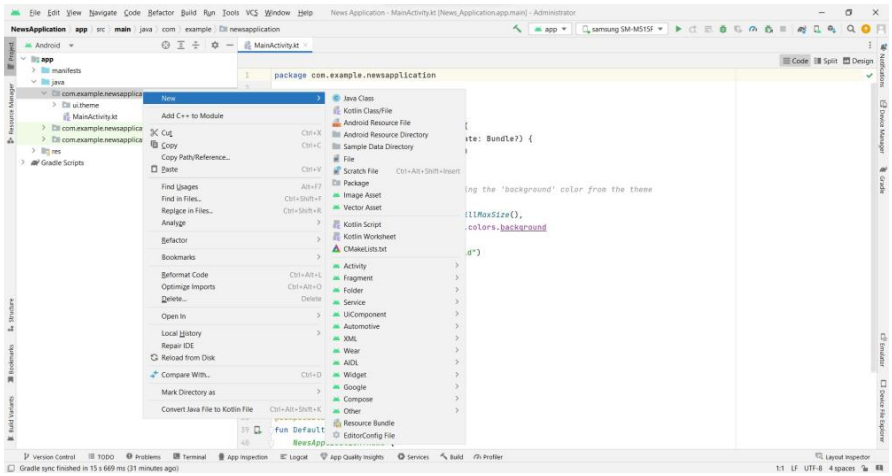
- To learn more about Database follow this link:

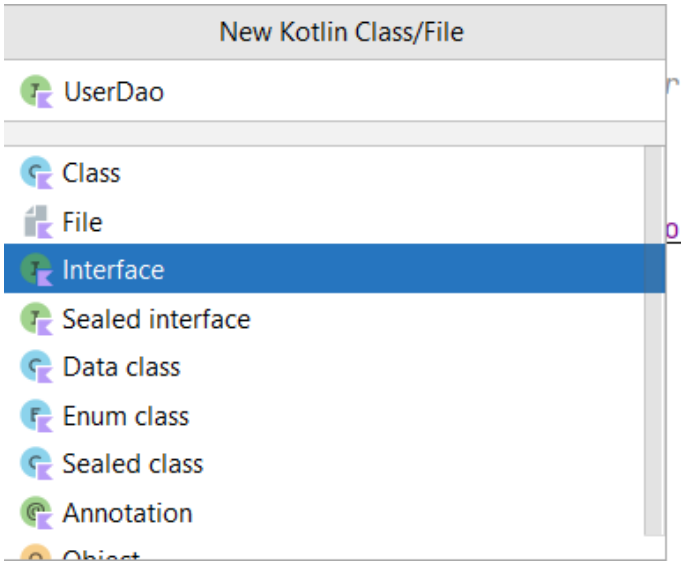
Step 1 : Create User data class



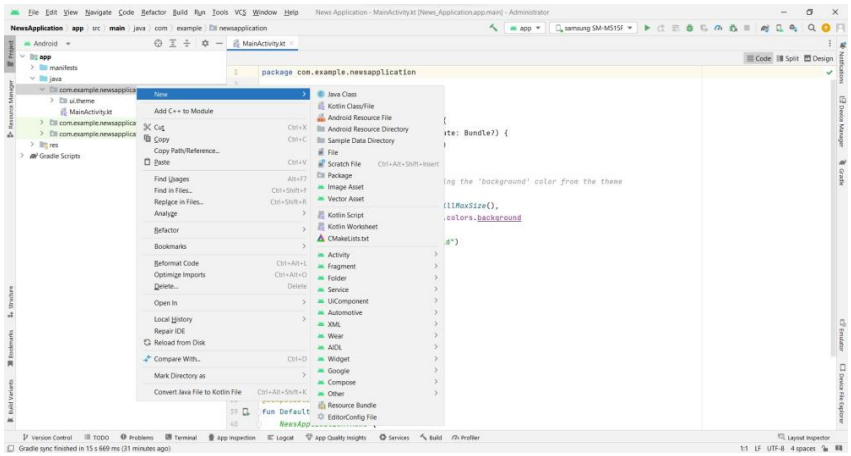


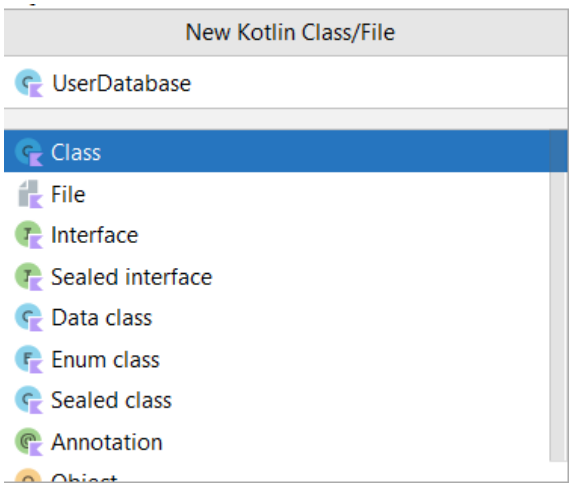
Step 2 : Create an UserDao interface



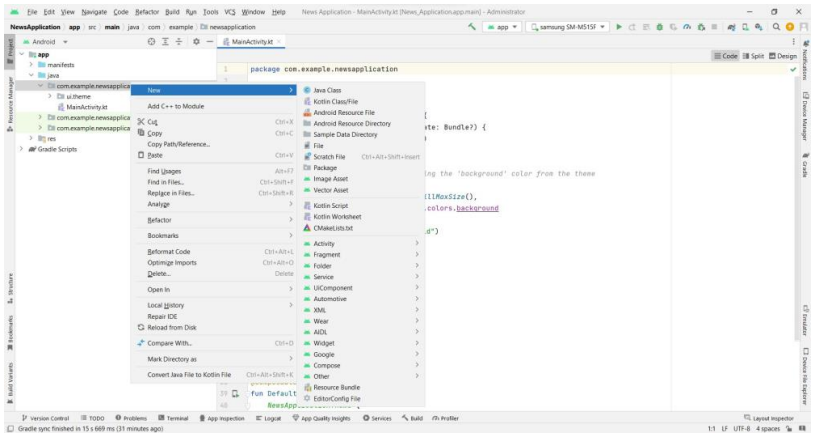


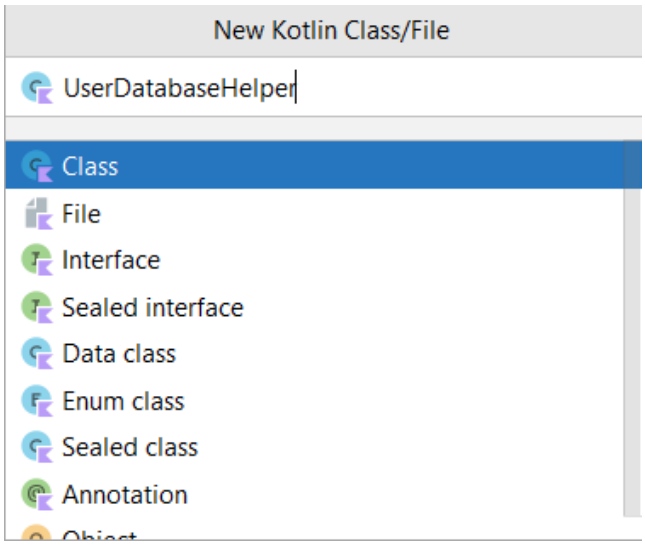
Step 3 : Create an UserDatabase class





Step 4 : Create an UserDatabaseHelper class

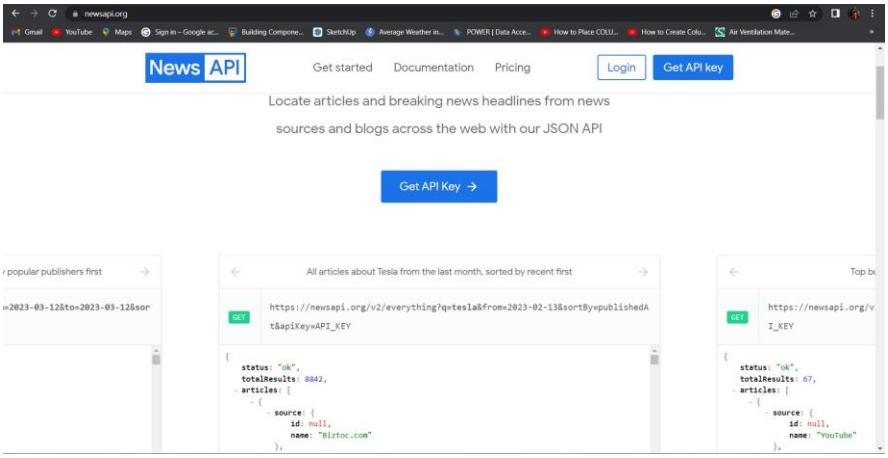




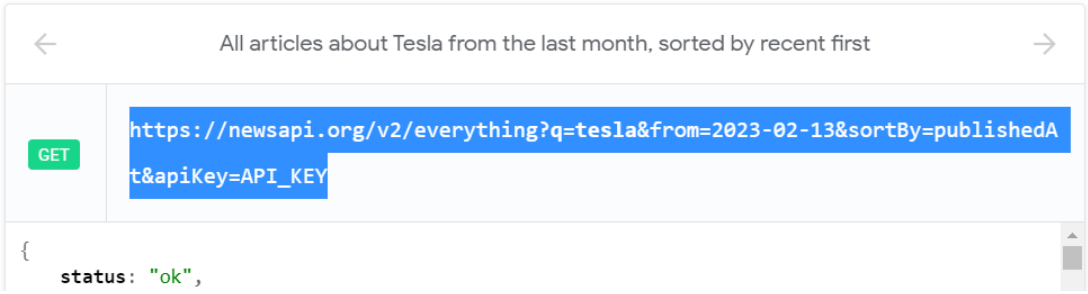
Task 6 :

Creating API Service and required classes for integrating API

Step 1: Create a API key for the required API

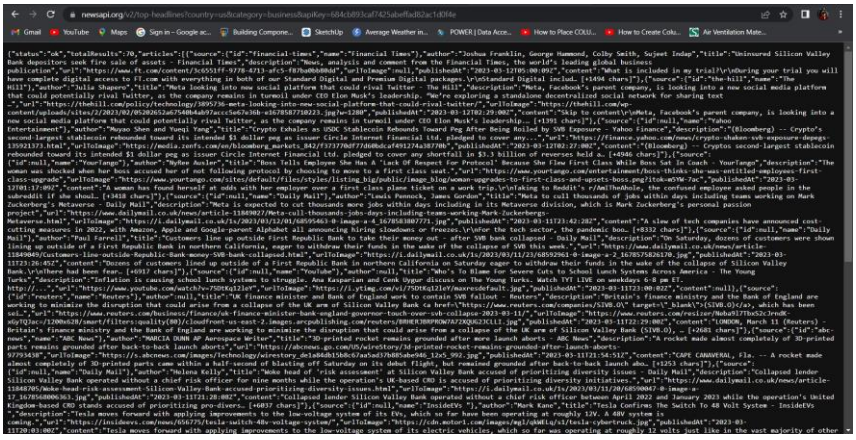


- Code which is needed to copy:



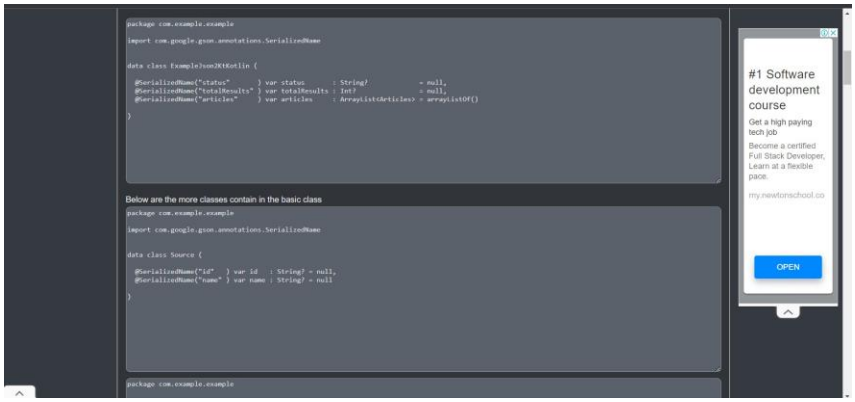
- Enter the API key at API_KEY to get the complete json file.

Json File



Step 2: Copy and paste the complete json file to json2kt.com

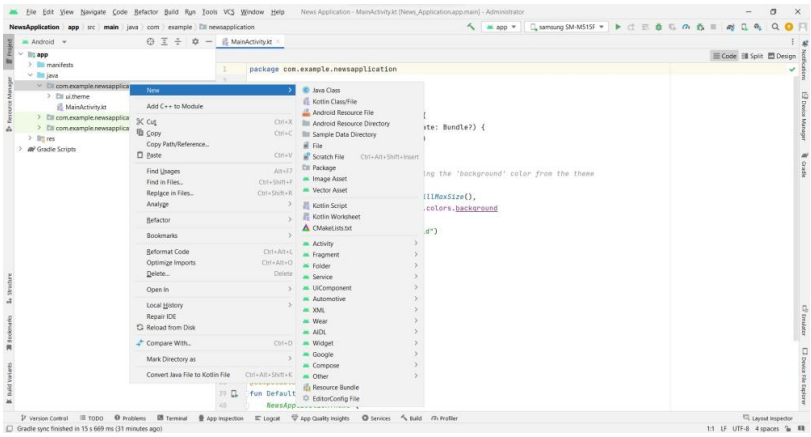
Click on Output Data class

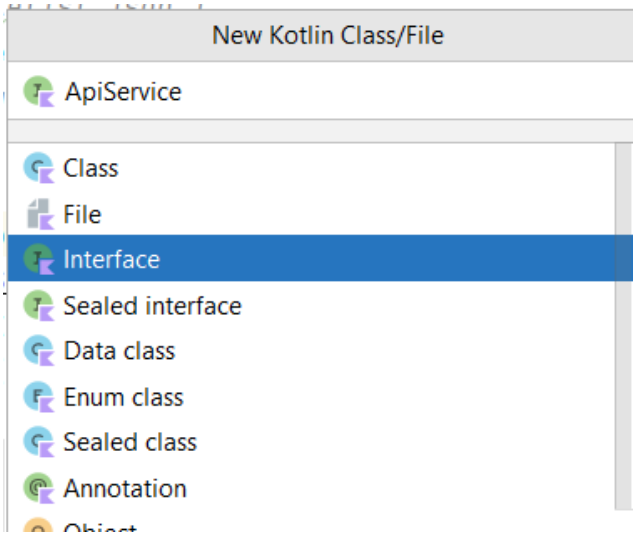


Now use this classes in android studio

Database for news integration into project

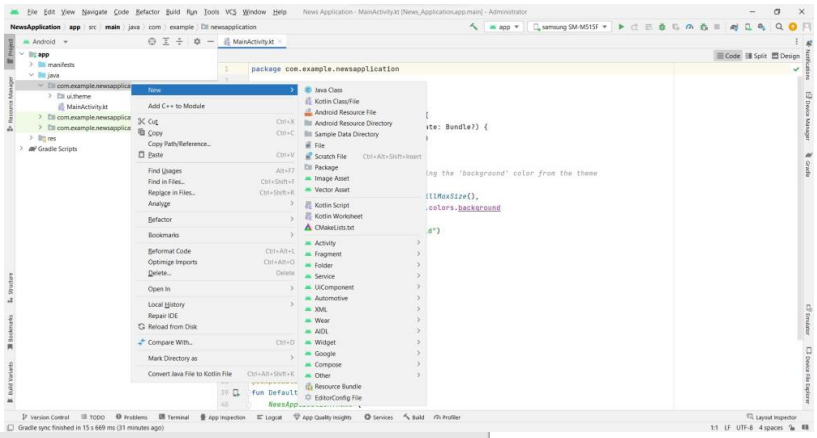
Step 3:
Create ApiService interface

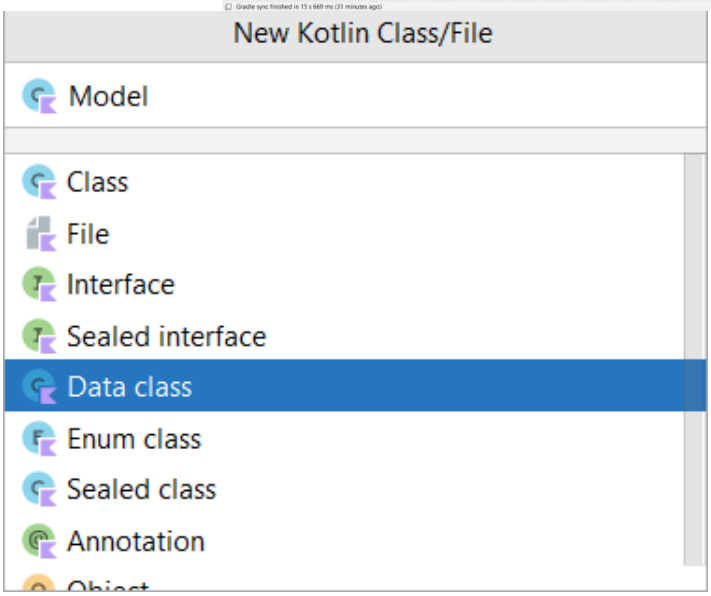





```
package com.example.newsheadlines

import ...

interface ApiService {

    @GET("top-headlines?country=us&category=business&apiKey=684cb893caf7425abeffad82ac1d0f4e")
    suspend fun getMovies() :News

    companion object {
        var apiService: ApiService? = null
        fun getInstance() : ApiService {
            if (apiService == null) {
                apiService = Retrofit.Builder()
                    .baseUrl("https://newsapi.org/v2/")

                    .addConverterFactory(GsonConverterFactory.create())
                    .build().create(ApiService::class.java)
            }
            return apiService!!
        }
    }

}
```

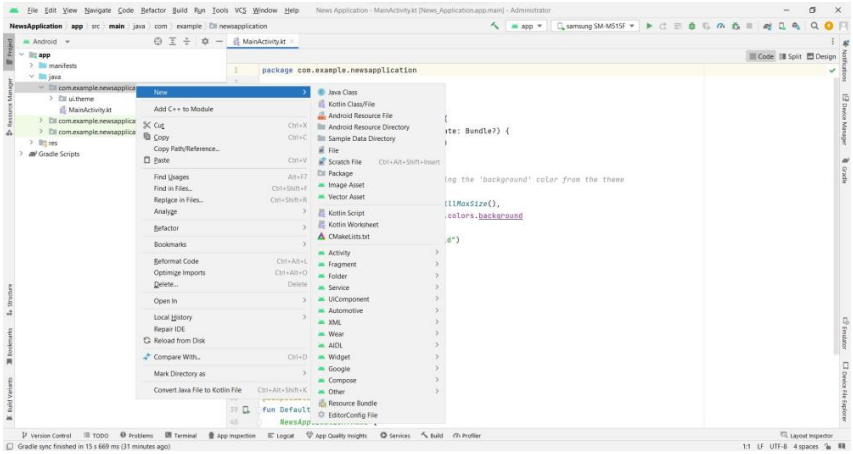
Step 4:
Create Model data class

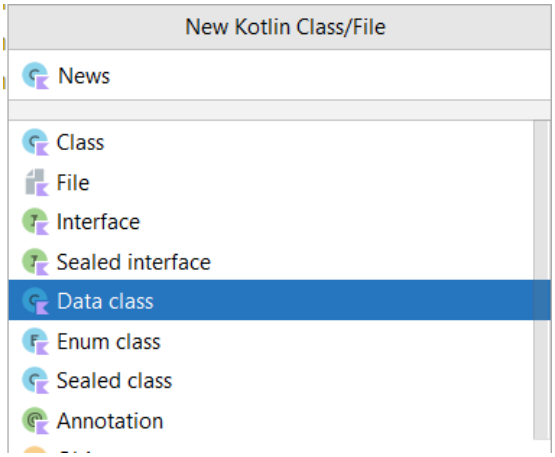





```
package com.example.newsheadlines


data class Movie(val name: String,
                 val imageUrl: String,
                 val desc: String,
                 val category: String)
```

Step 5:
Create News data class

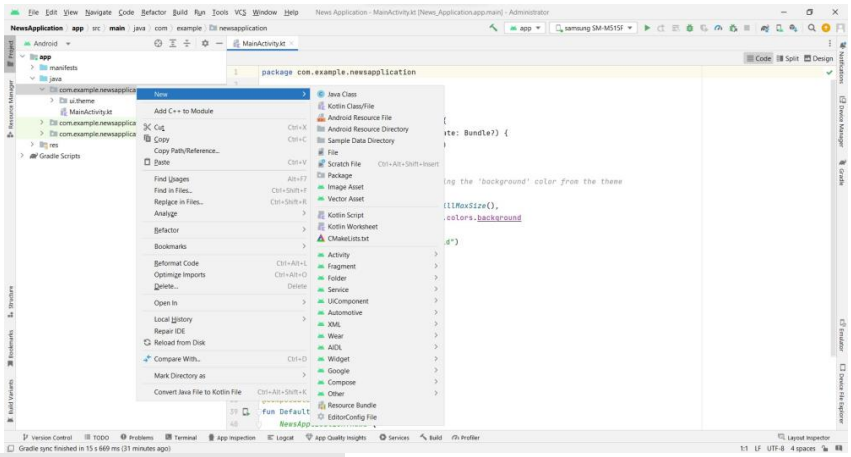


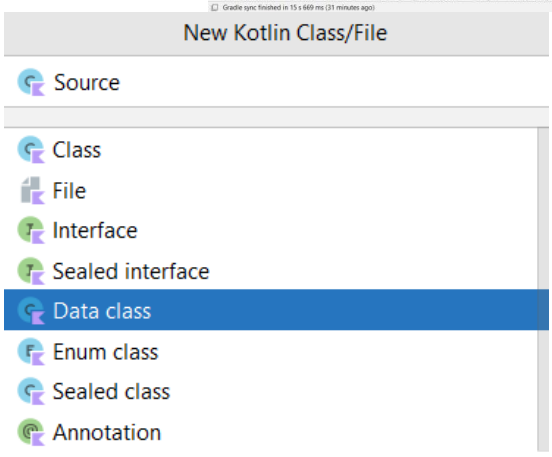


```
package com.example.newsheadlines

import ...


data class News (
    @SerializedName("status") var status:String?= null,
    @SerializedName("totalResults") var totalResults : Int?                 = null,
    @SerializedName("articles") var articles     : ArrayList<Articles> = arrayListOf()
)
```

Step 6:

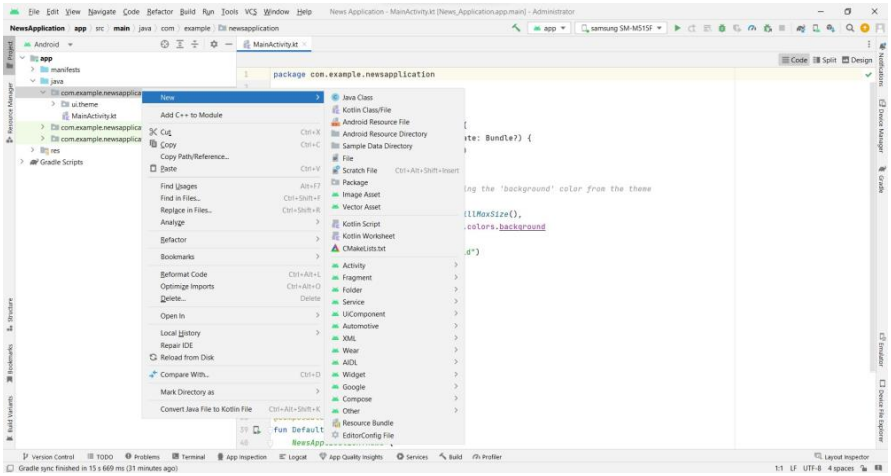
Create Source data class

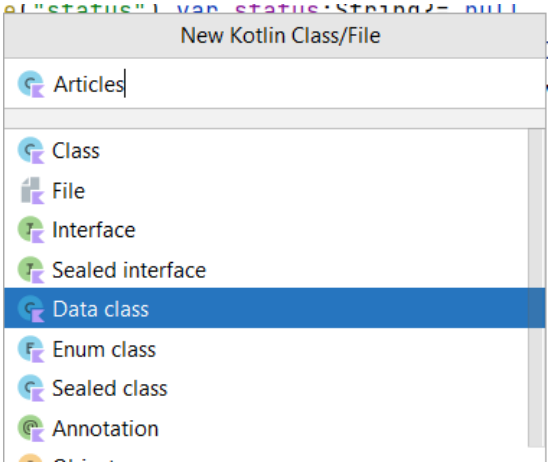




```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Source (

  @SerializedName("id"   ) var id   : String? = null,
  @SerializedName("name" ) var name : String? = null

)
```

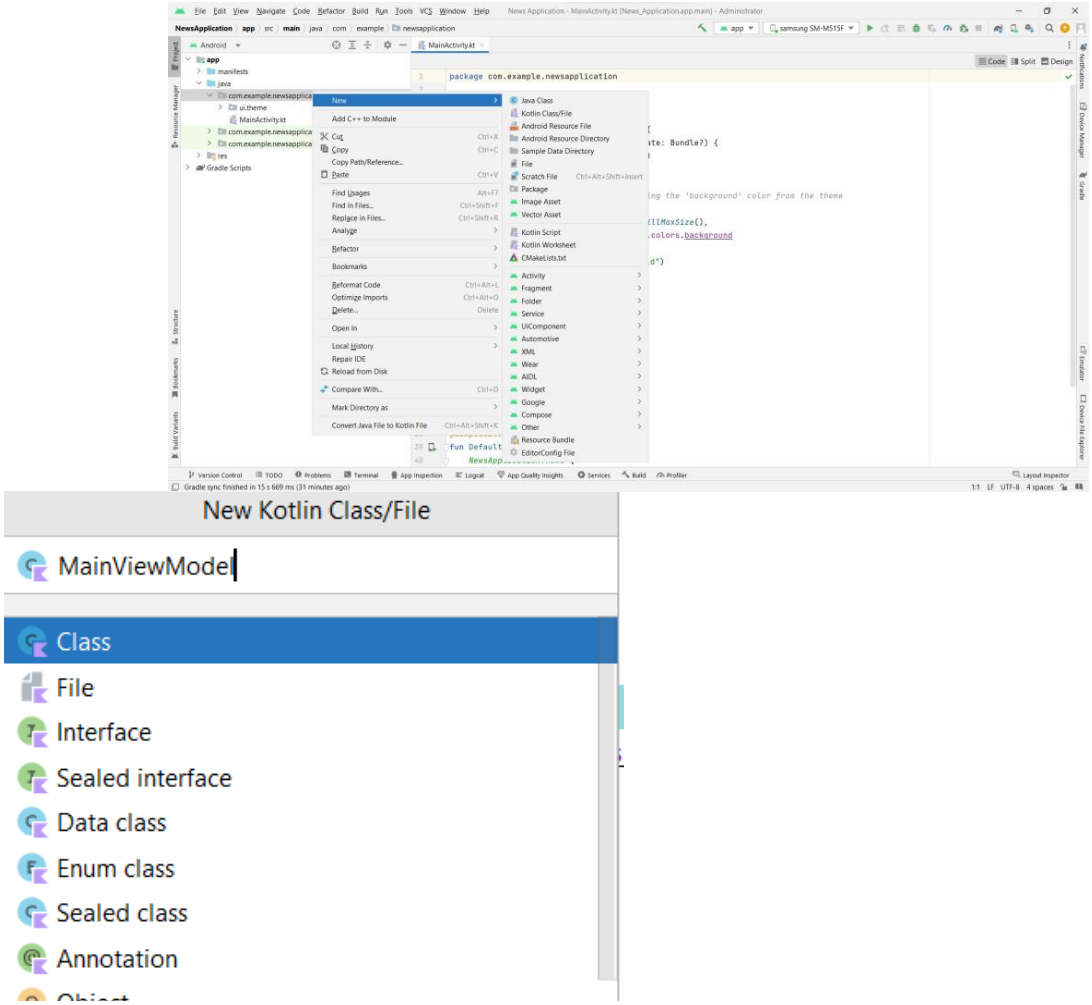
Step 7:
Create Articles data class

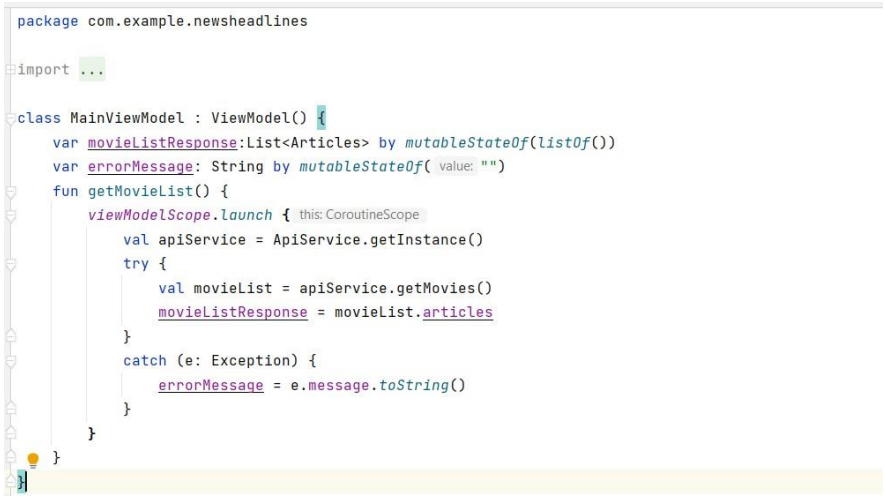




```
package com.example.example

import com.google.gson.annotations.SerializedName


data class Articles (

  @SerializedName("title"       ) var title       : String? = null,
  @SerializedName("description" ) var description : String? = null,
  @SerializedName("urlToImage"  ) var urlToImage  : String? = null,

)
```

Step 8:
Create MainViewModel class



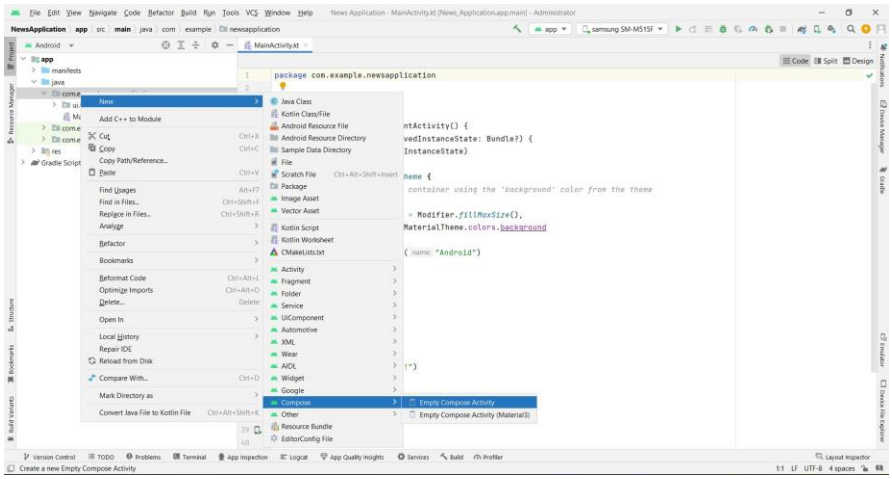


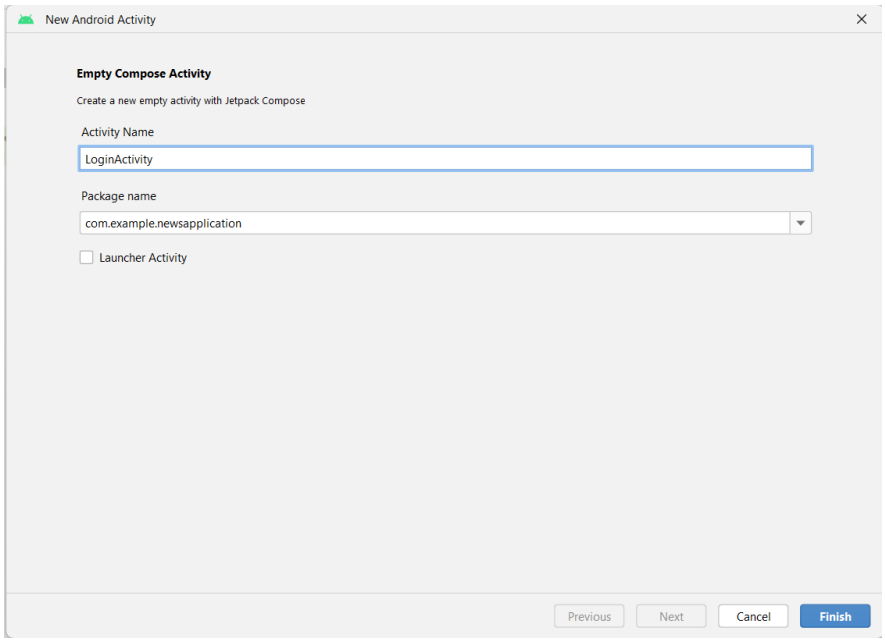
```
package com.example.newsheadlines

import ...

class MainViewModel : ViewModel() {
    var movieListResponse:List<Articles> by mutableStateOf(listOf())
    var errorMessage: String by mutableStateOf( value: "")
    fun getMovieList() {
        viewModelScope.launch { this: CoroutineScope
            val apiService = ApiService.getInstance()
            try {
                val movieList = apiService.getMovies()
                movieListResponse = movieList.articles
            }
            catch (e: Exception) {
                errorMessage = e.message.toString()
            }
        }
    }
}
```

Task 7:

Building application UI and connecting to database.

Step 1: Creating LoginActivity.kt with database

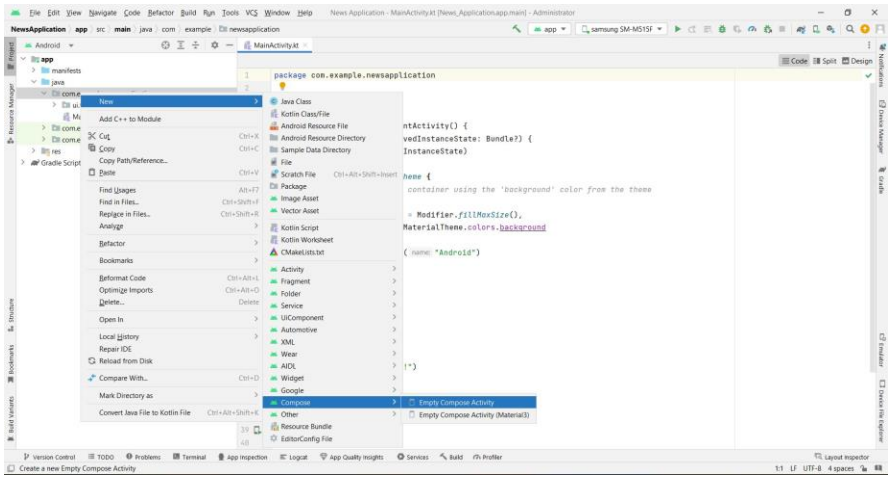


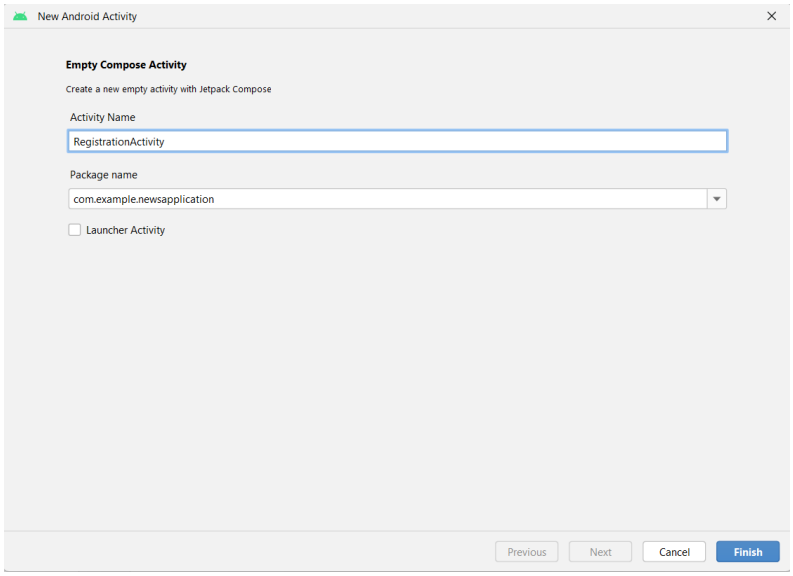


Database connection in LoginActivity.kt

```kotlin
package com.example.newsheadlines

import ...

class LoginActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper( context: this)
        setContent {

            LoginScreen( context: this, databaseHelper)
        }
    }
}
@Composable
fun LoginScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    var username by remember { mutableStateOf( value: "") }
    var password by remember { mutableStateOf( value: "") }
    var error by remember { mutableStateOf( value: "") }
```

Step 2 : Creating RegistrationActivity.kt with database





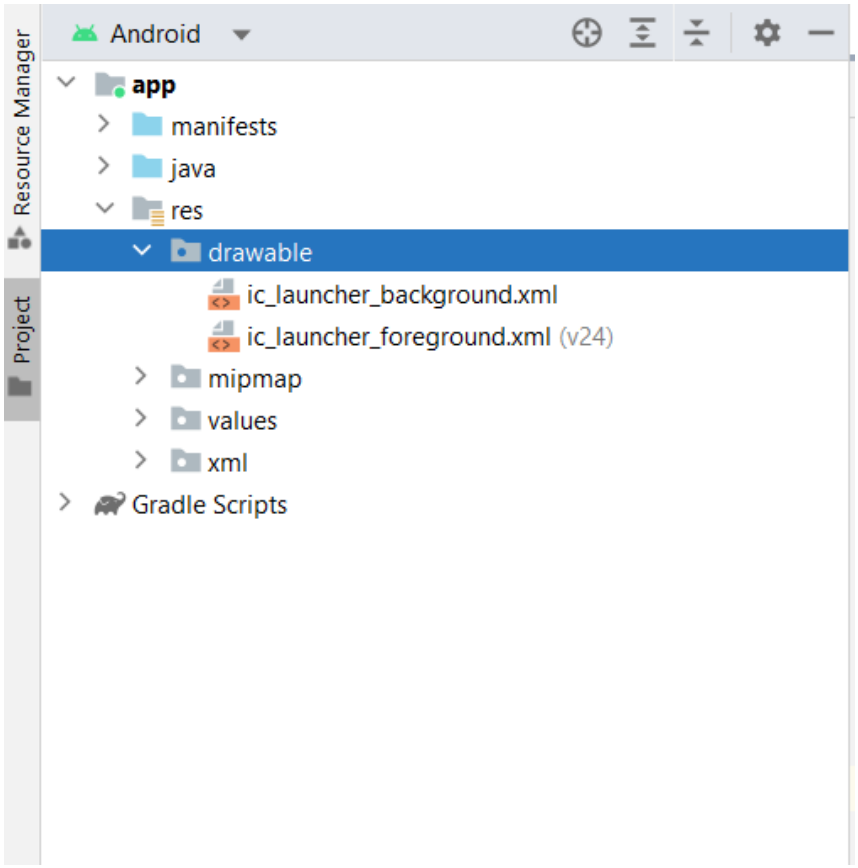
Database connection in RegistrationActivity.kt

```
package com.example.newsheadlines

import ...

class RegistrationActivity : ComponentActivity() {
    private lateinit var databaseHelper: UserDatabaseHelper
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        databaseHelper = UserDatabaseHelper( context: this)
        setContent {


                RegistrationScreen( context: this,databaseHelper)
            }
        }
    }



@Composable
fun RegistrationScreen(context: Context, databaseHelper: UserDatabaseHelper) {
    var username by remember { mutableStateOf( value: "") }
    var password by remember { mutableStateOf( value: "") }
    var email by remember { mutableStateOf( value: "") }
    var error by remember { mutableStateOf( value: "") }

    Column(
```

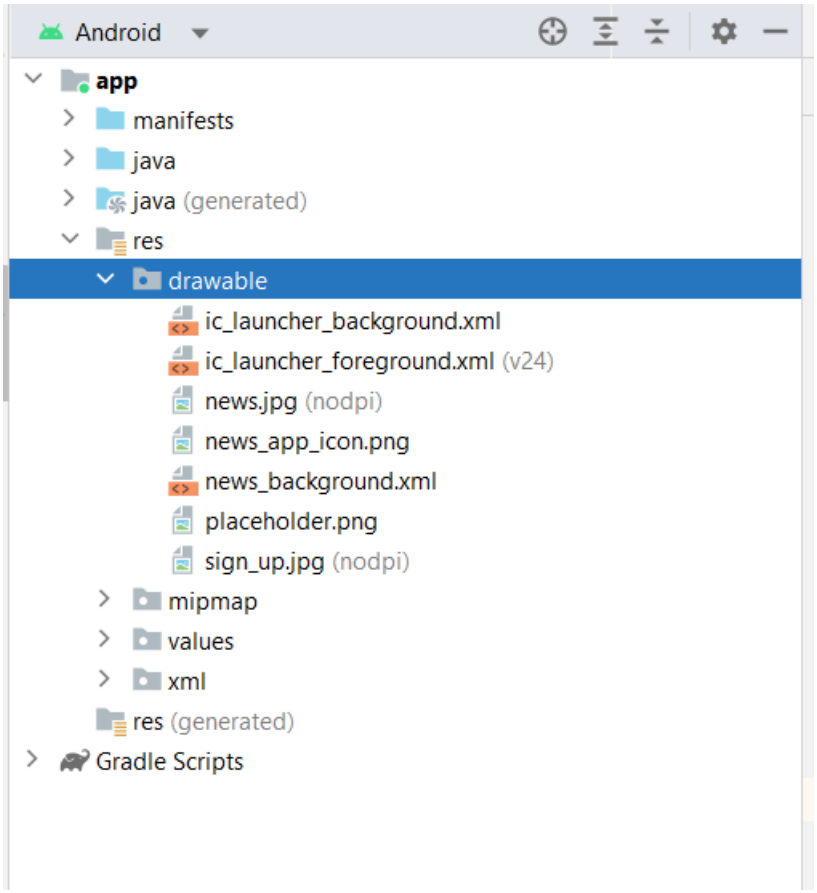
Step 3 : Creating MainActivity.kt file

In MainActivity.kt file the main application is developed

- Before creating UI we need to add some images in drawables which are in res



Required drawables



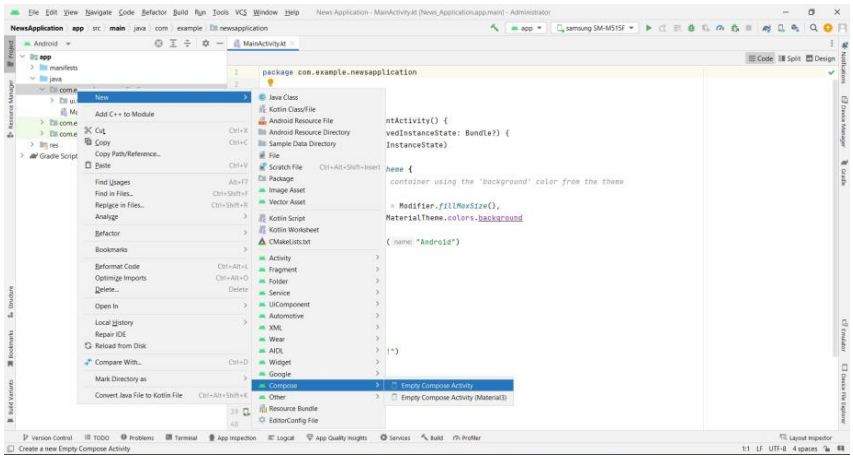
- After add all this we need to create the UI in MainActivity.kt file

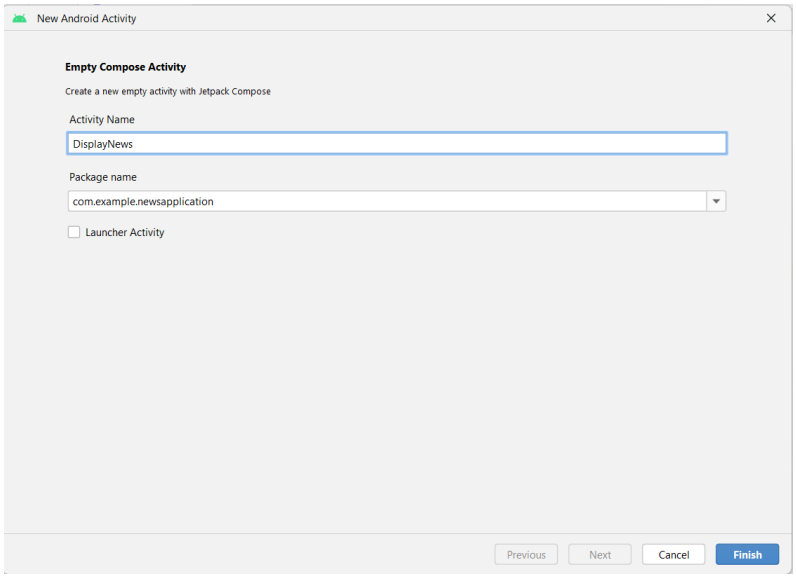
Mainpage.kt file

Linking MainActivity with DisplayNews.kt

```
            modifier = Modifier
                .fillMaxHeight()
                .weight(0.3f)
        )


        Column(
            verticalArrangement = Arrangement.Center,
            modifier = Modifier
                .padding(4.dp)
                .fillMaxHeight()
                .weight(0.8f)
                .background(Color.Gray)
                .padding(20.dp)
                .selectable( selected: true,  enabled: true,  role: null,
                    onClick = {
                        Log.i( tag: "test123abc",  msg: "MovieItem: $index/n${movie.description}")
                        context.startActivity(
                            Intent(context, DisplayNews::class.java)
                                .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                                .putExtra( name: "desk", movie.description.toString())
                                .putExtra( name: "urlToImage", movie.urlToImage)
                                .putExtra( name: "title", movie.title)
                        )
                    })
        ) { this: ColumnScope
```

Step 4 : Creating DisplayNews.kt file

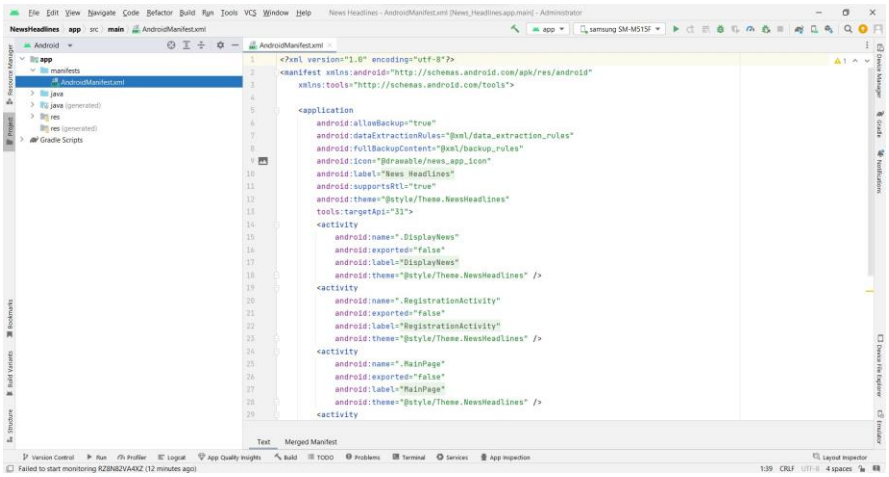





```
class DisplayNews : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            NewsHeadlinesTheme {
                // A surface container using the 'background' color from the theme
                Surface(
                    modifier = Modifier.fillMaxSize(),
                    color = MaterialTheme.colors.background
                ) {

                    var desk = getIntent().getStringExtra( name: "desk")
                    var title = getIntent().getStringExtra( name: "title")
                    var uriImage = getIntent().getStringExtra( name: "urlToImage")
                    Log.i( tag: "test123abc",  msg: "MovieItem: $desk")

                    Column(Modifier.background(Color.Gray).padding(20.dp), horizontalAlignment = Alignment.CenterHorizontally
                        Text(text = ""+title, fontSize = 32.sp)
                        HtmlText(html = desk.toString())
                        /*  AsyncImage(
                              model = "https://example.com/image.jpg",
                              contentDescription = "Translated description of what the image contains"
                          )*/
                        Image(
                            painter = rememberImagePainter(uriImage),
                            contentDescription = "My content description",
                        )
                    }
                    //  Greeting(desk.toString())
```

Task 8:

Modifying AndroidManifest.xml



When we run the app we will get the MainActivity.kt file as our first screen , but we want LoginActivity.kt , So we need to change in AndroidManifest.xml.

Changed AndroidManifest.xml.

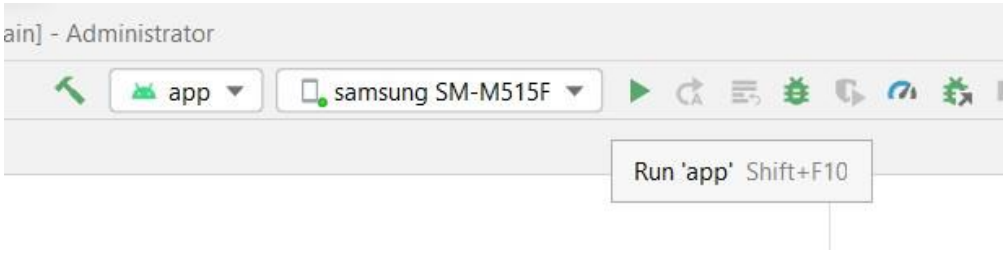
```
AndroidManifest.xml
        android:theme="@style/Theme.NewsHeadlines"
        tools:targetApi="31">
        <activity
            android:name=".DisplayNews"
            android:exported="false"
            android:label="DisplayNews"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".RegistrationActivity"
            android:exported="false"
            android:label="RegistrationActivity"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".MainPage"
            android:exported="false"
            android:label="@string/title_activity_main_page"
            android:theme="@style/Theme.NewsHeadlines" />
        <activity
            android:name=".LoginActivity"
            android:exported="true"
            android:label="News Headlines"
            android:theme="@style/Theme.NewsHeadlines">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

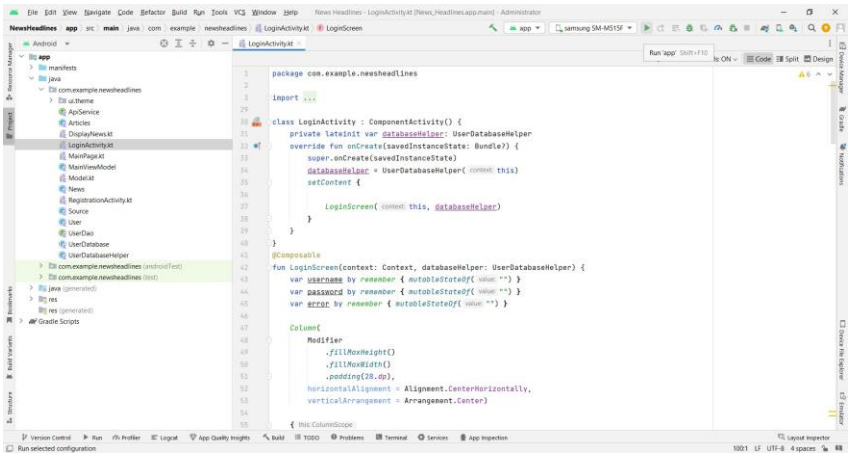
Task 9:

Running the application.

Step 1: Run apps on a hardware device

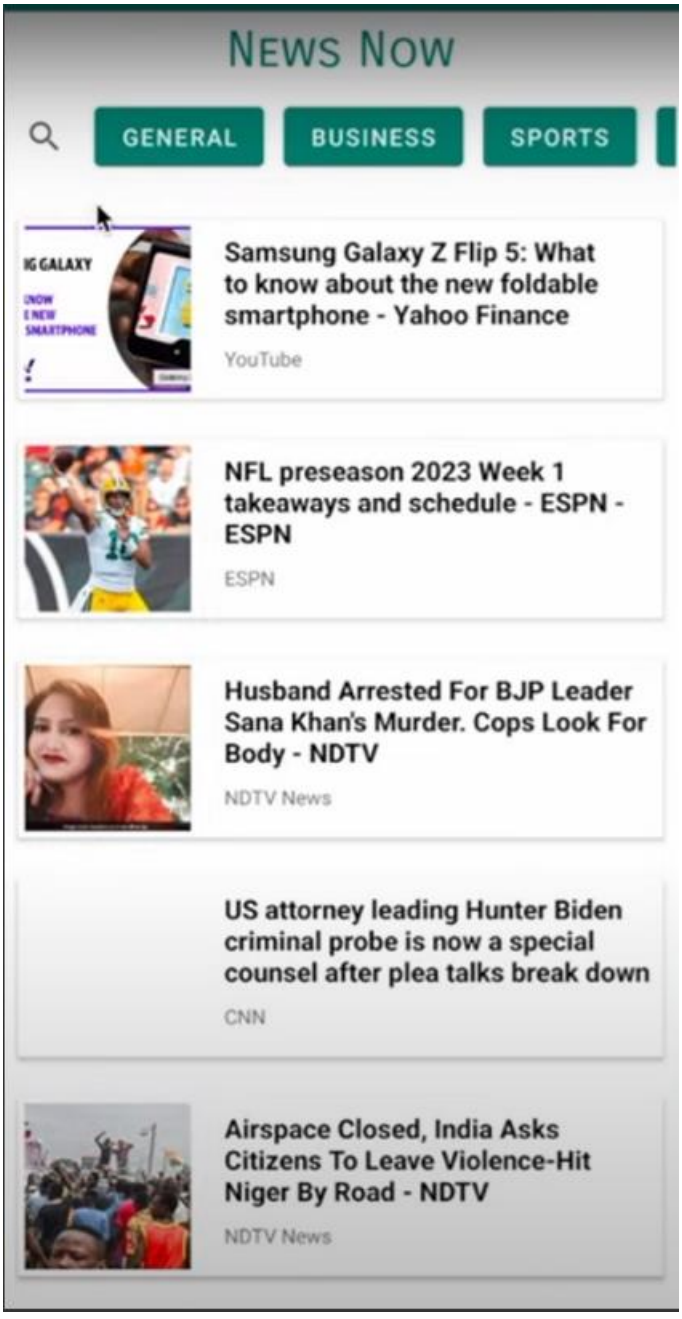
Step 2: Run the application in Mobile





Final Output of the Application : https://github.com/smartinternz02/SI-GuidedProject-587502-1696857144

Register Page :




News Now