Category: Android Application Development

Project Title

An Android Application For Keeping Up With The Latest Headlines

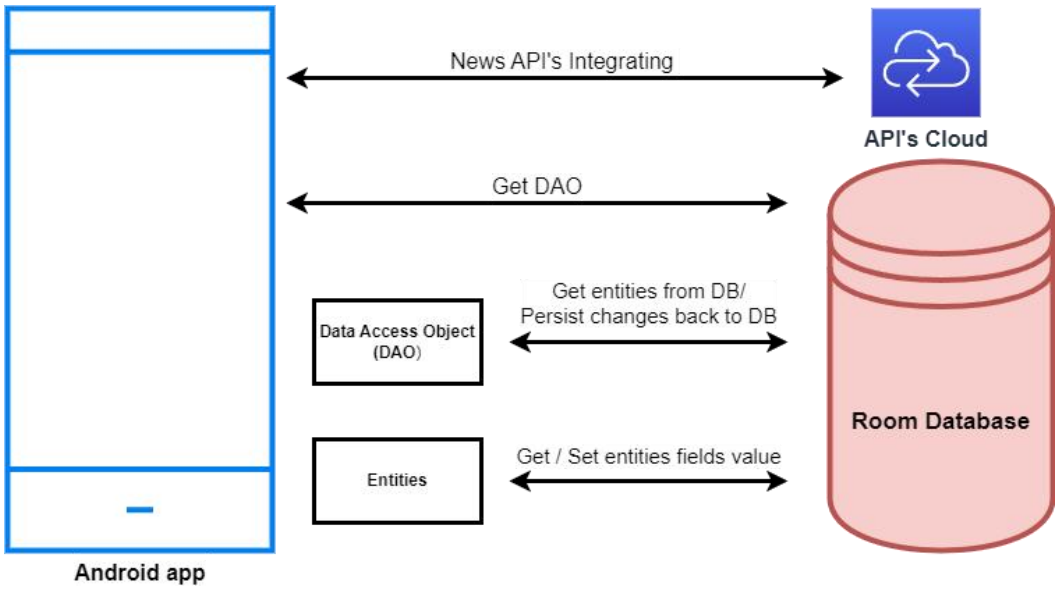
-Project Based Experiential Learning Program

Team Members
20MH1A0577(Dadi Lakshmi Neha)
20MH1A0595(Mangina Suvarna)
20MH1A05C0(Vangapandu Chandrika)

An Android Application For Keeping Up With The Latest Headlines

The app's main feature is displaying a list of news articles, each with a title, image, and brief description. Users can scroll through the list of articles and tap on an article to view more details. The app uses the Jetpack Compose UI toolkit to build the UI and it uses the coil library to load images. The app fetches data from a remote server using Retrofit library and demonstrates how to use the Jetpack Compose UI toolkit for Android development.

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

Note:

To complete the project you need to finish up the tasks listed below:

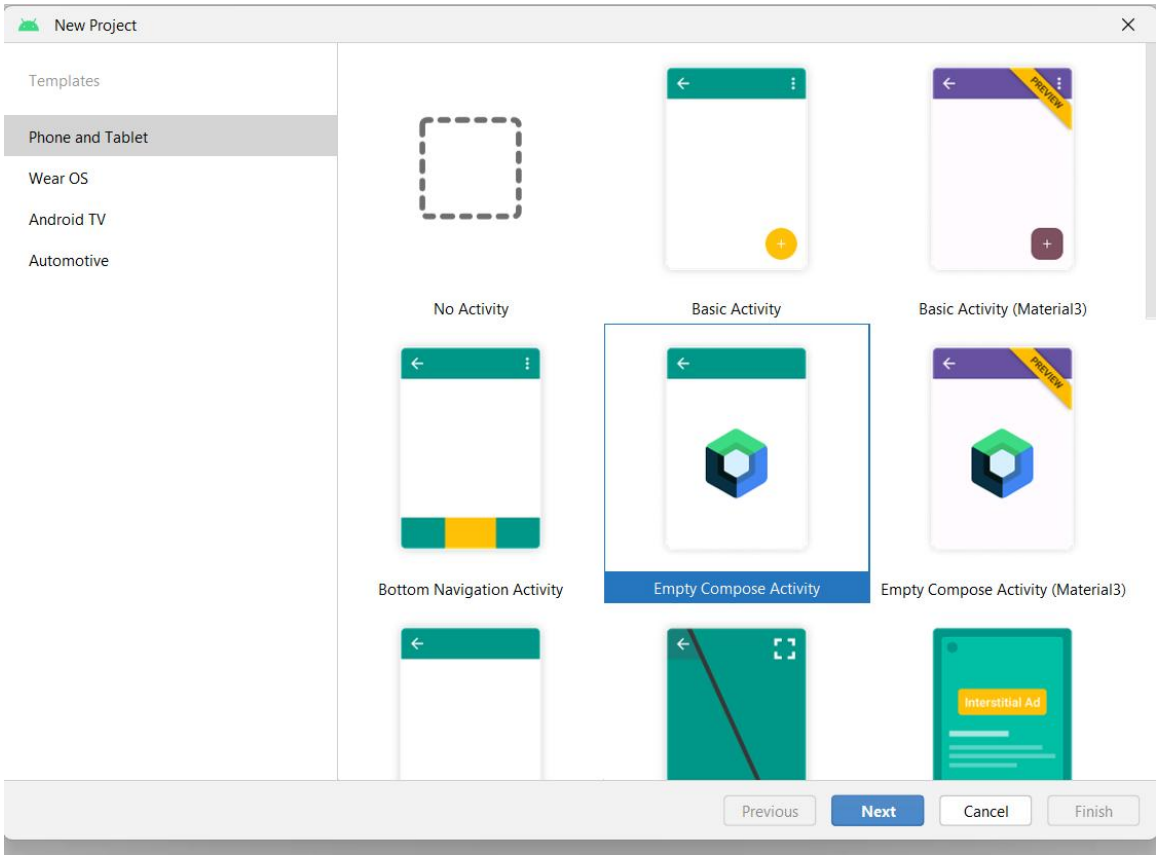
Tasks:

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.

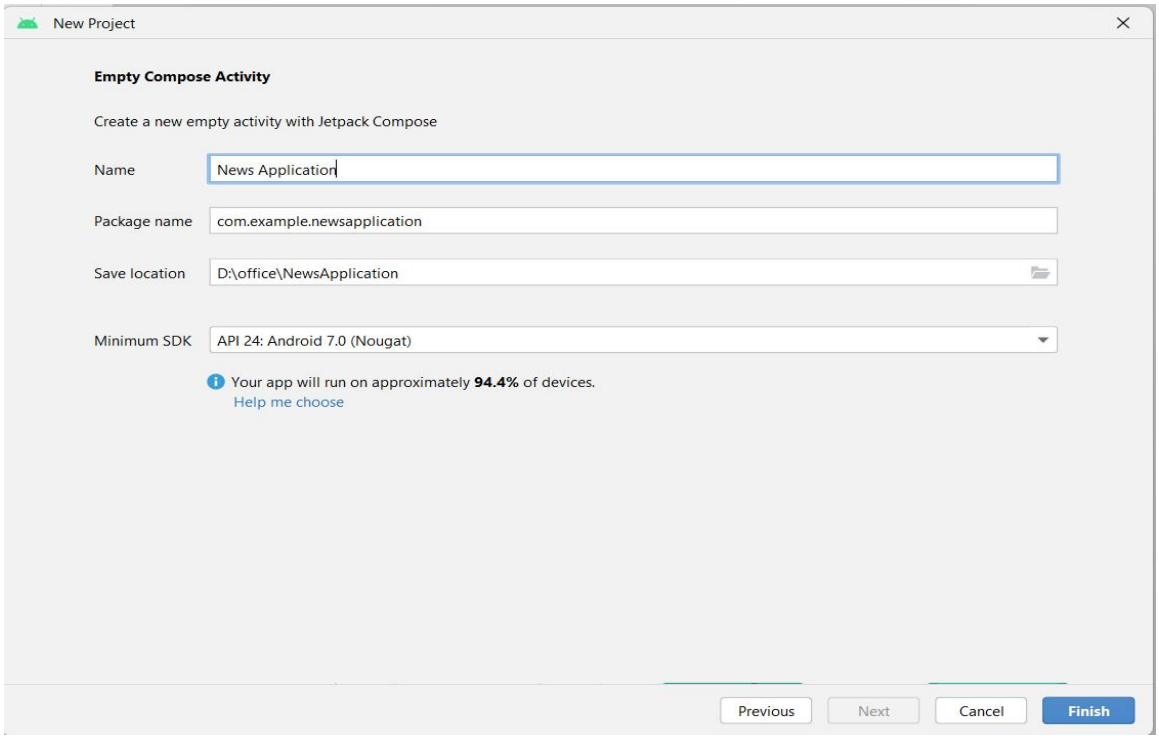
Creating A New Project

Steps need to be followed:

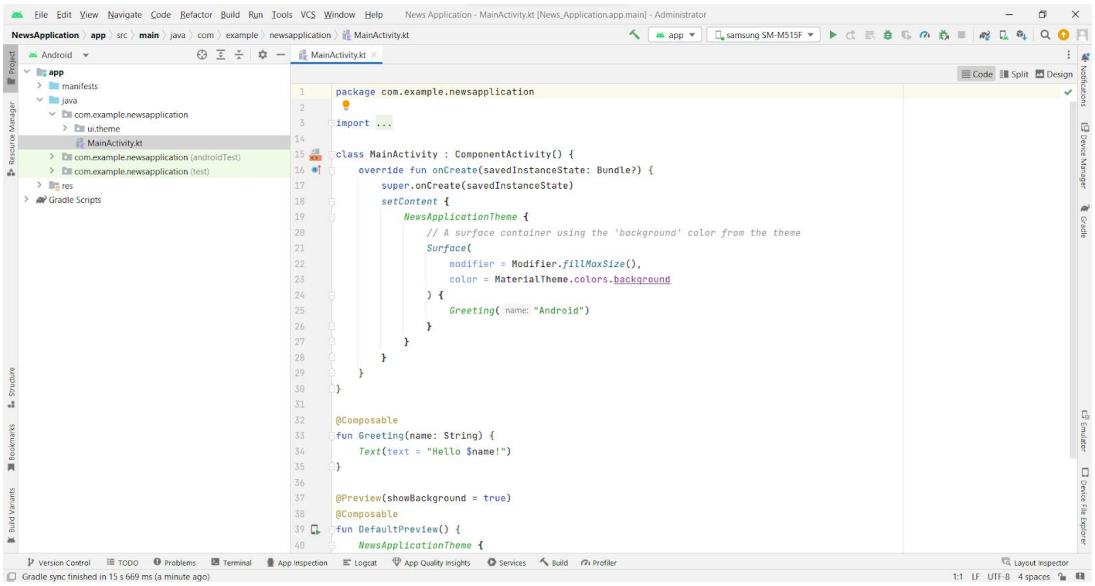
Step 1 : Android studio > File > New > New Project > Empty Compose Activity
Step 2 : Click on **Next** button.



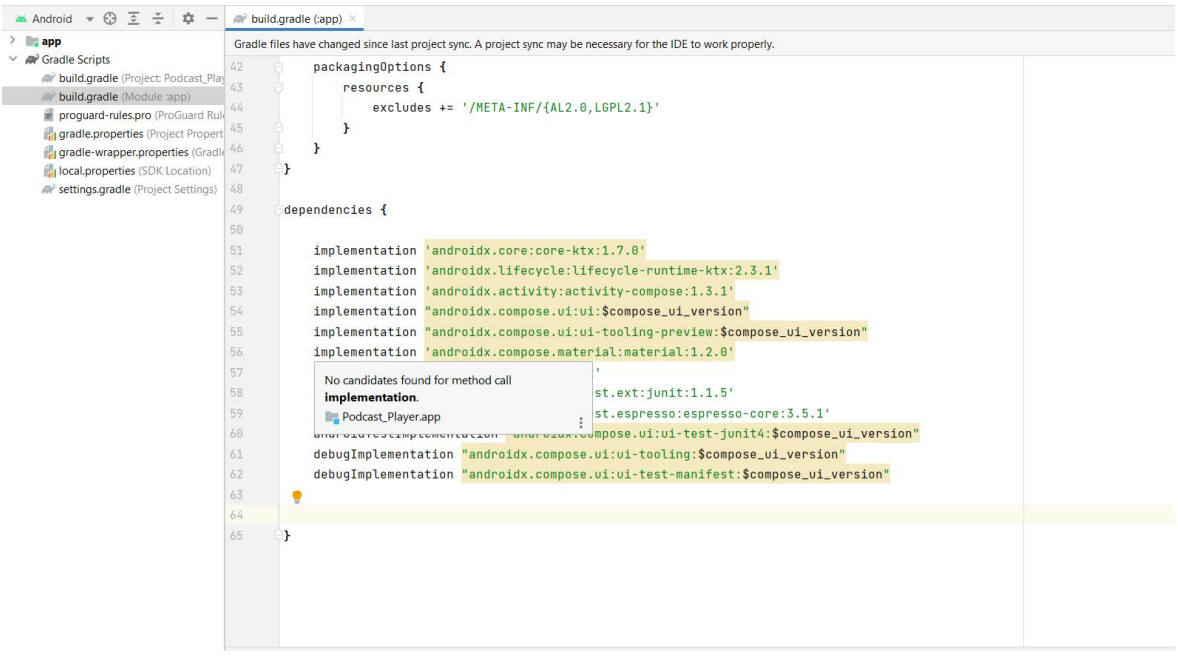
Step 3 : Give name to the new project.
Step 4 : Give the Minimum SDK value
Step 5 : Click Finish



Main activity file



Gradle Scripts > Build.Gradle(Module :App)



Adding Retrofit Dependencies

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'

Adding Coil Dependencies

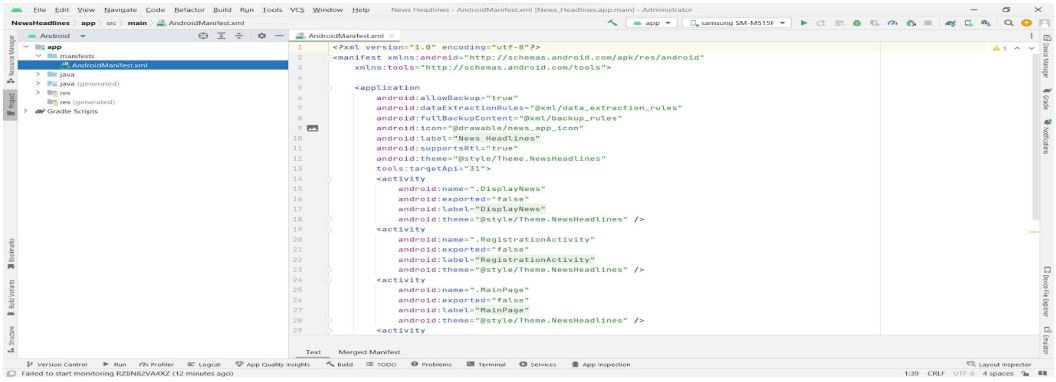
implementation("io.coil-kt:coil-compose:1.4.0")

```
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")
}
```

Click on Sync now.

Adding Permissions:

Open AndroidManifest.Xml



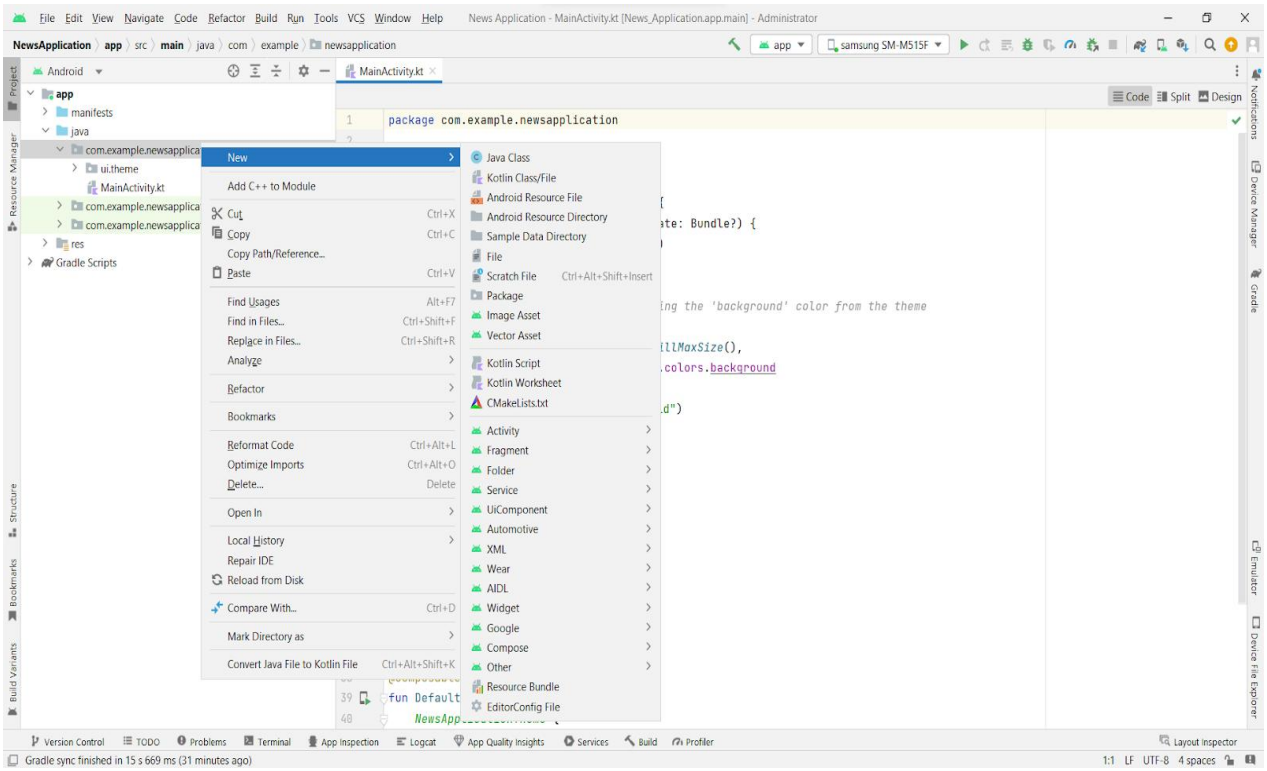
Add Permission To Access Wifi And Internet

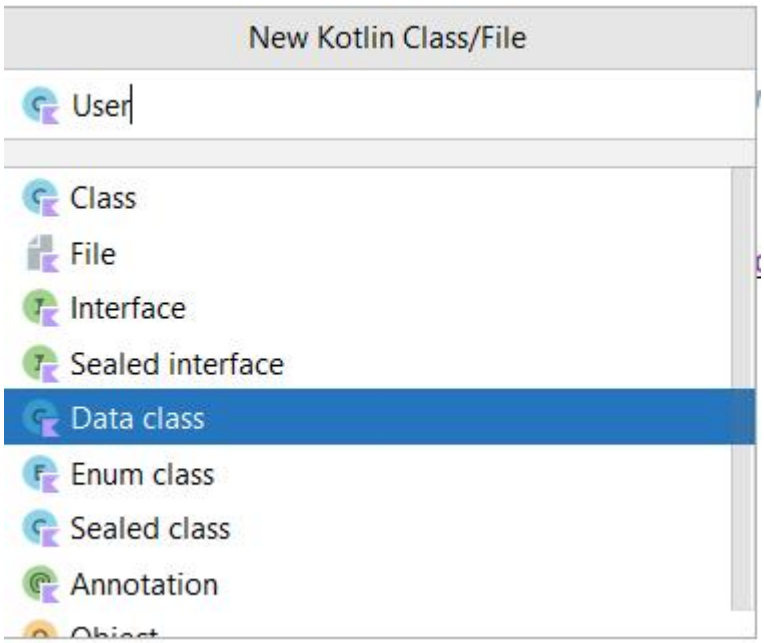
```
<!--    permissions-->
  <uses-permission android:name="android.permission.INTERNET"/>
  <uses-permission
android:name="android.permission.ACCESS_WIFI_STATE"/>
```

```
<?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"
```

Creating The Database Classes:

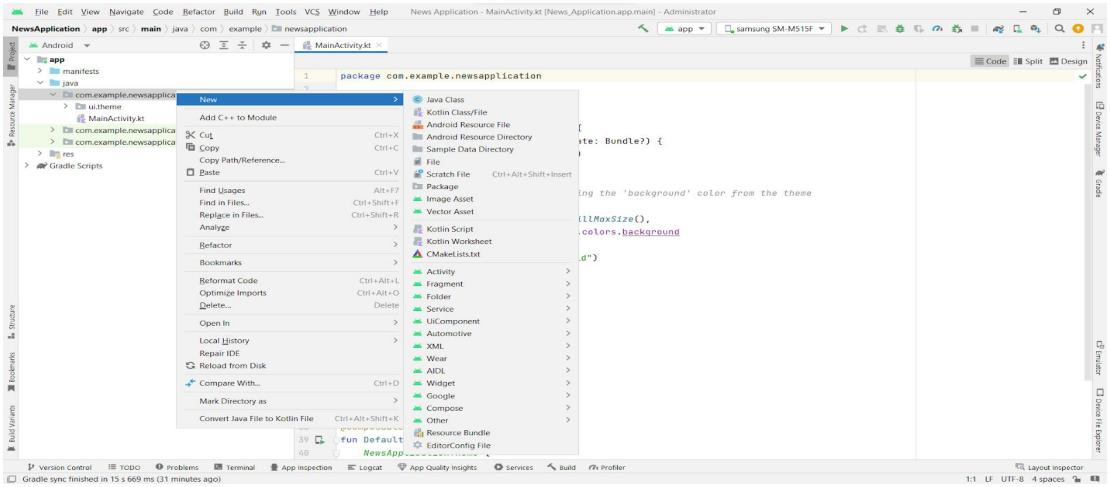
Create User Data Class

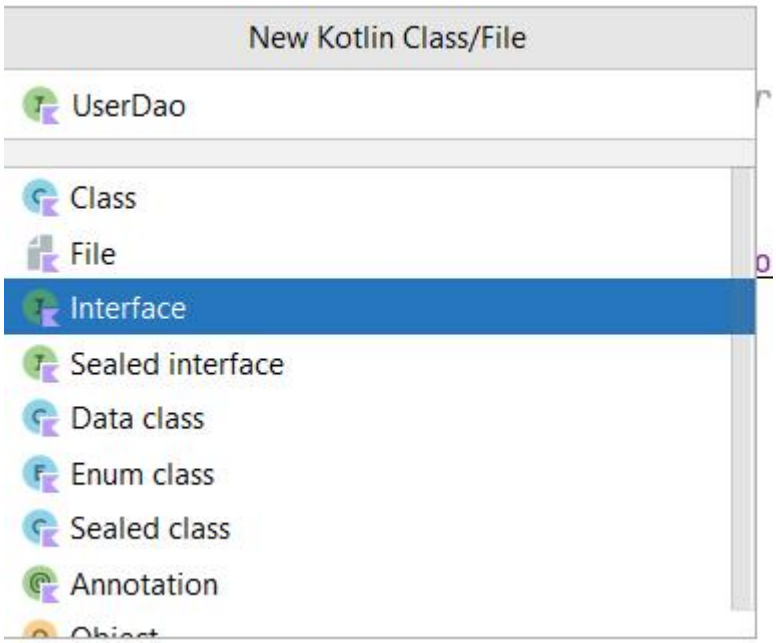





```kotlin
package com.example.androidnews
import androidx.room.ColumnInfo
import androidx.room.Entity
import androidx.room.PrimaryKey
@Entity(tableName = "user_table")
data class User(
    @PrimaryKey(autoGenerate = true) val id: Int?,
    @ColumnInfo(name = "first_name") val firstName: String?,
    @ColumnInfo(name = "last_name") val lastName: String?,
    @ColumnInfo(name = "email") val email: String?,
    @ColumnInfo(name = "password") val password: String?,
)
```

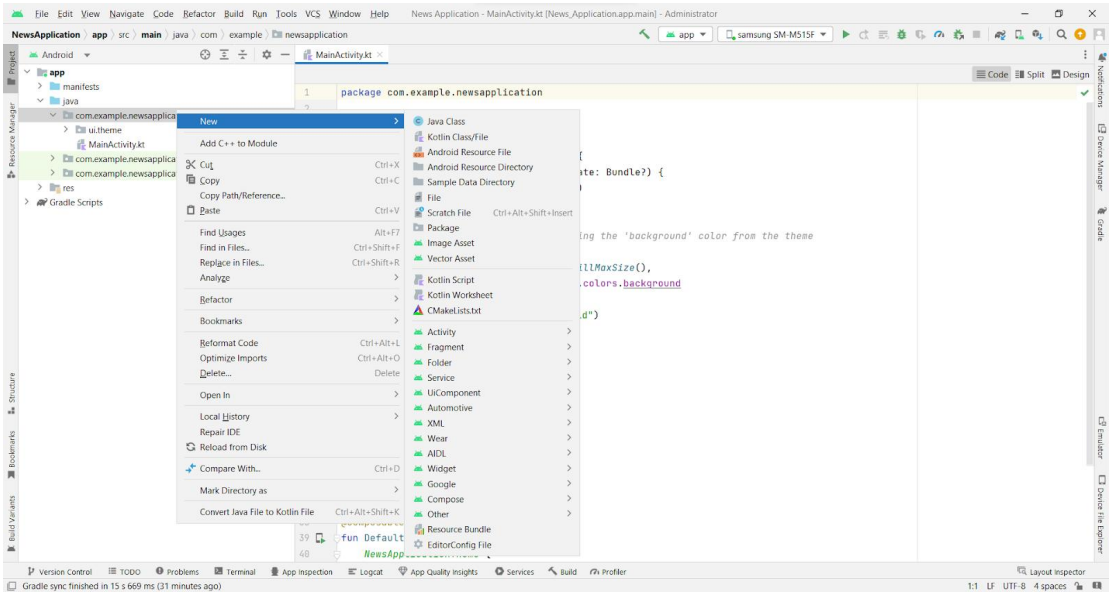
Create An UserDao Interface

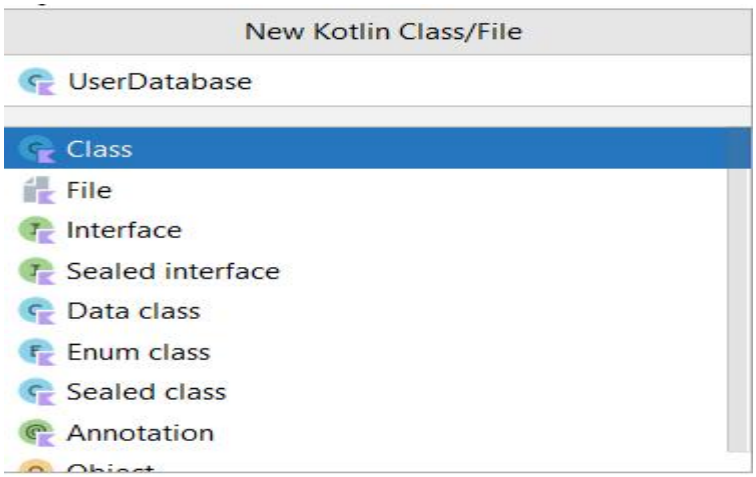





```kotlin
package com.example.androidnews
import androidx.room.*
@Dao
interface UserDao {
    @Query("SELECT * FROM user_table WHERE email = :email")
    suspend fun getUserByEmail(email: String): User?
    @Insert(onConflict = OnConflictStrategy.REPLACE)
    suspend fun insertUser(user: User)
    @Update
    suspend fun updateUser(user: User)
    @Delete
    suspend fun deleteUser(user: User)
}
```

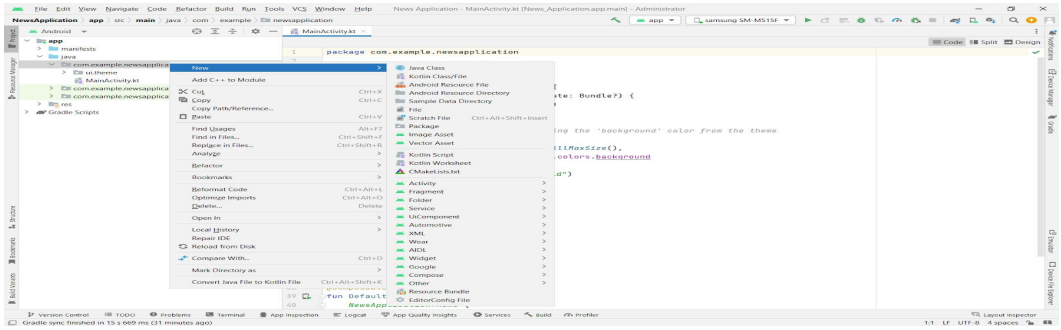
Create An UserDatabase Class

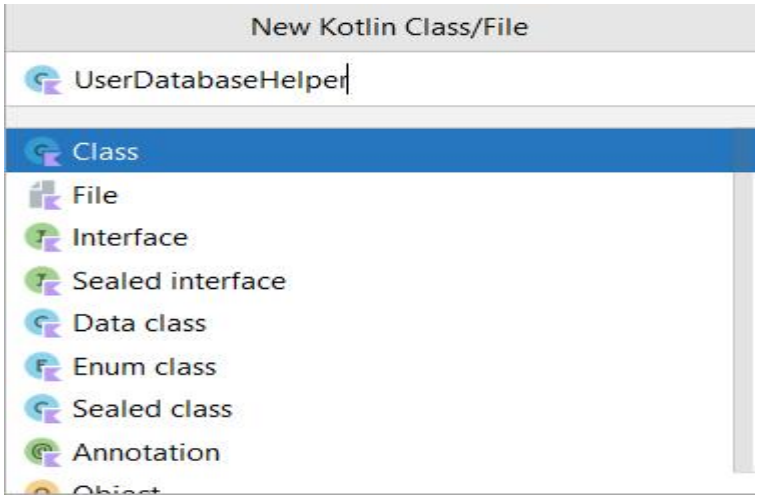





```
package com.example.androidnews
import android.content.Context
import androidx.room.Database
import androidx.room.Room
import androidx.room.RoomDatabase
@Database(entities = [User::class], version = 1)
abstract class UserDatabase : RoomDatabase() {
   abstract fun userDao(): UserDao
   companion object {
      @Volatile
      private var instance: UserDatabase? = null
      fun getDatabase(context: Context): UserDatabase {
         return instance ?: synchronized(this) {
            val newInstance = Room.databaseBuilder(
               context.applicationContext,
               UserDatabase::class.java,
               "user_database"
            ).build()
            instance = newInstance
            newInstance
         }
      }
   }
}
```

Create An UserDatabaseHelper Class



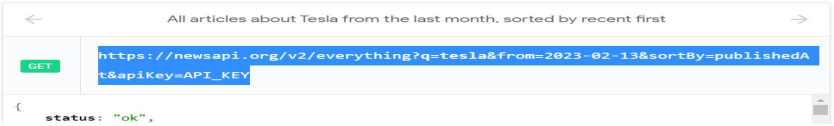


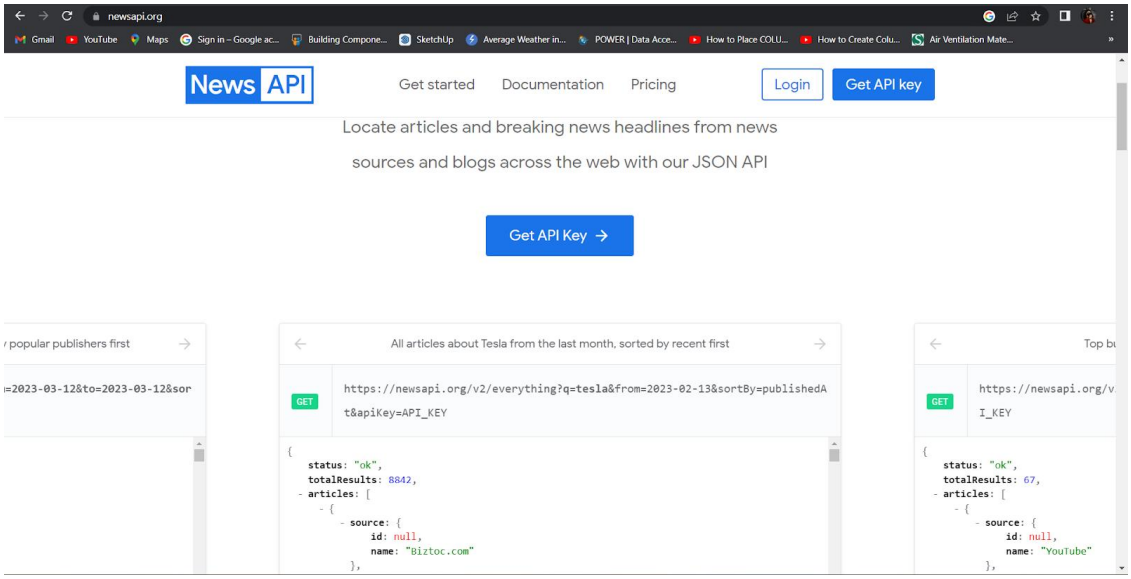

```
package com.example.androidnews
import android.annotation.SuppressLint
import android.content.ContentValues
import android.content.Context
import android.database.Cursor
import android.database.sqlite.SQLiteDatabase
import android.database.sqlite.SQLiteOpenHelper
class UserDatabaseHelper(context: Context):
    SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
    companion object {
        private const val DATABASE_VERSION = 1
        private const val DATABASE_NAME = "UserDatabase.db"

        private const val TABLE_NAME = "user_table"
        private const val COLUMN_ID = "id"
        private const val COLUMN_FIRST_NAME = "first_name"
        private const val COLUMN_LAST_NAME = "last_name"
        private const val COLUMN_EMAIL = "email"
        private const val COLUMN_PASSWORD = "password"
    }

    override fun onCreate(db: SQLiteDatabase?) {
        val createTable = "CREATE TABLE $TABLE_NAME (" +
                "$COLUMN_ID INTEGER PRIMARY KEY AUTOINCREMENT, " +
                "$COLUMN_FIRST_NAME TEXT, " +
                "$COLUMN_LAST_NAME TEXT, " +
                "$COLUMN_EMAIL TEXT, " +
                "$COLUMN_PASSWORD TEXT" +
                ")"
```

```
        db?.execSQL(createTable)
    }

    override fun onUpgrade(db: SQLiteDatabase?, oldVersion: Int, newVersion: Int) {
        db?.execSQL("DROP TABLE IF EXISTS $TABLE_NAME")
        onCreate(db)
    }

    fun insertUser(user: User) {
        val db = writableDatabase
        val values = ContentValues()
        values.put(COLUMN_FIRST_NAME, user.firstName)
        values.put(COLUMN_LAST_NAME, user.lastName)
        values.put(COLUMN_EMAIL, user.email)
        values.put(COLUMN_PASSWORD, user.password)
        db.insert(TABLE_NAME, null, values)
        db.close()
    }
    @SuppressLint("Range")
    fun getUserByUsername(username: String): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_FIRST_NAME = ?", arrayOf(username))
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
                id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
          firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
           lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
         email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
        password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)), ) }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getUserById(id: Int): User? {
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME WHERE
$COLUMN_ID = ?", arrayOf(id.toString()))
```

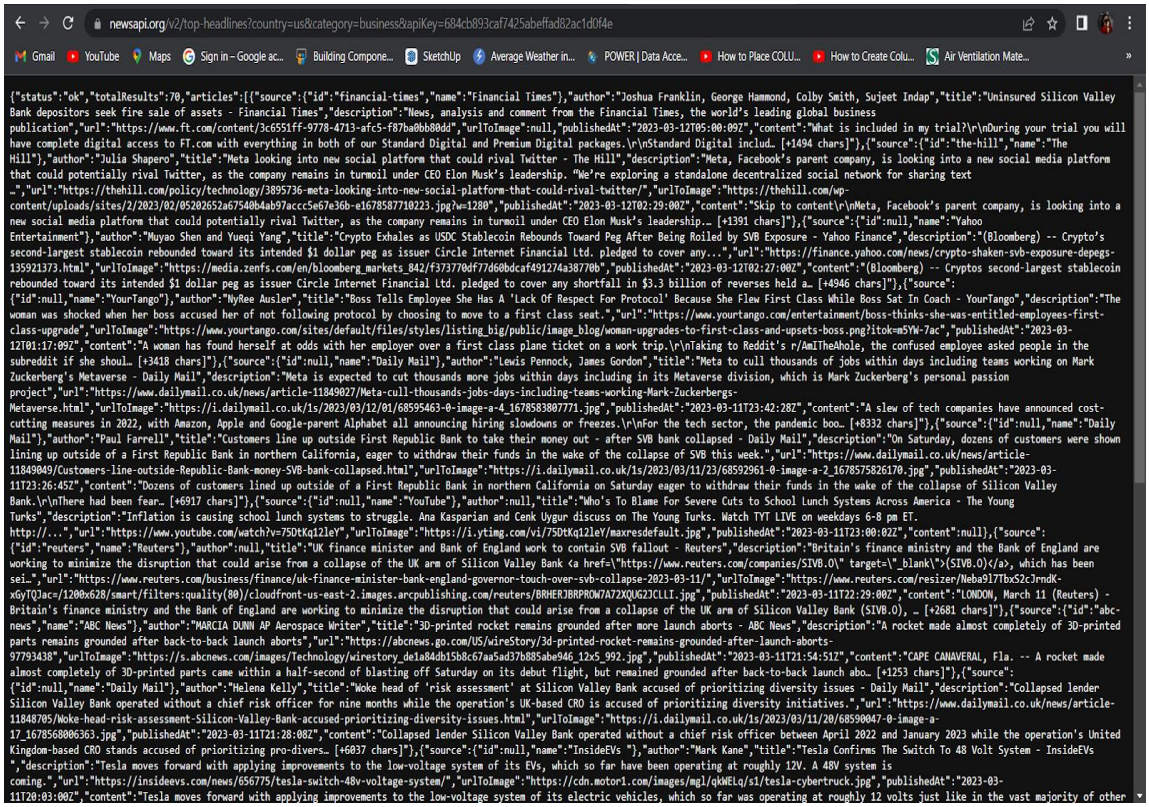
```
        var user: User? = null
        if (cursor.moveToFirst()) {
            user = User(
           id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
         firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
         lastName = cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
       email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
        password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)), ) }
        cursor.close()
        db.close()
        return user
    }
    @SuppressLint("Range")
    fun getAllUsers(): List<User> {
        val users = mutableListOf<User>()
        val db = readableDatabase
        val cursor: Cursor = db.rawQuery("SELECT * FROM $TABLE_NAME", null)
        if (cursor.moveToFirst()) {
            do {
                val user = User(
       id = cursor.getInt(cursor.getColumnIndex(COLUMN_ID)),
    firstName = cursor.getString(cursor.getColumnIndex(COLUMN_FIRST_NAME)),
 lastName =cursor.getString(cursor.getColumnIndex(COLUMN_LAST_NAME)),
  email = cursor.getString(cursor.getColumnIndex(COLUMN_EMAIL)),
  password = cursor.getString(cursor.getColumnIndex(COLUMN_PASSWORD)), )
                users.add(user)
            } while (cursor.moveToNext()) }
        cursor.close()
        db.close()
        return users
    }
}
```

Creating API Service And Required Classes For Integrating API

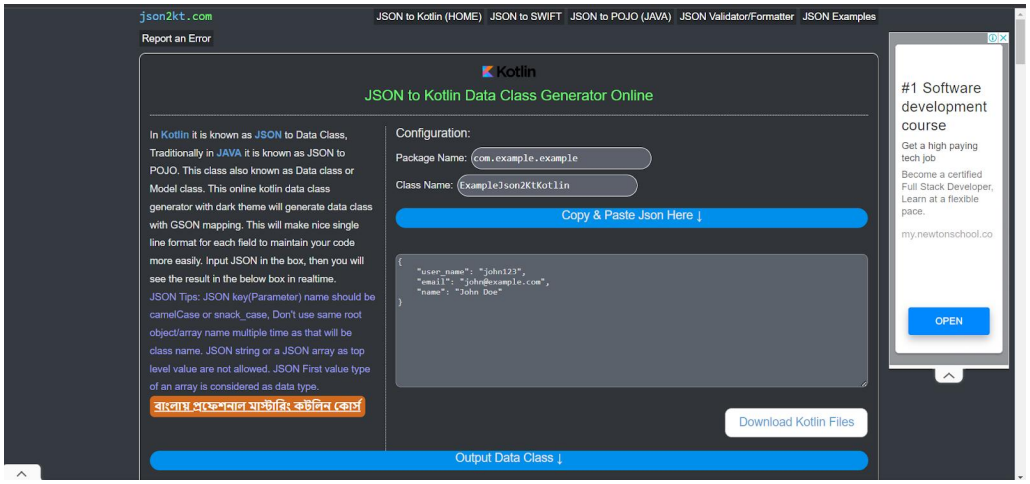




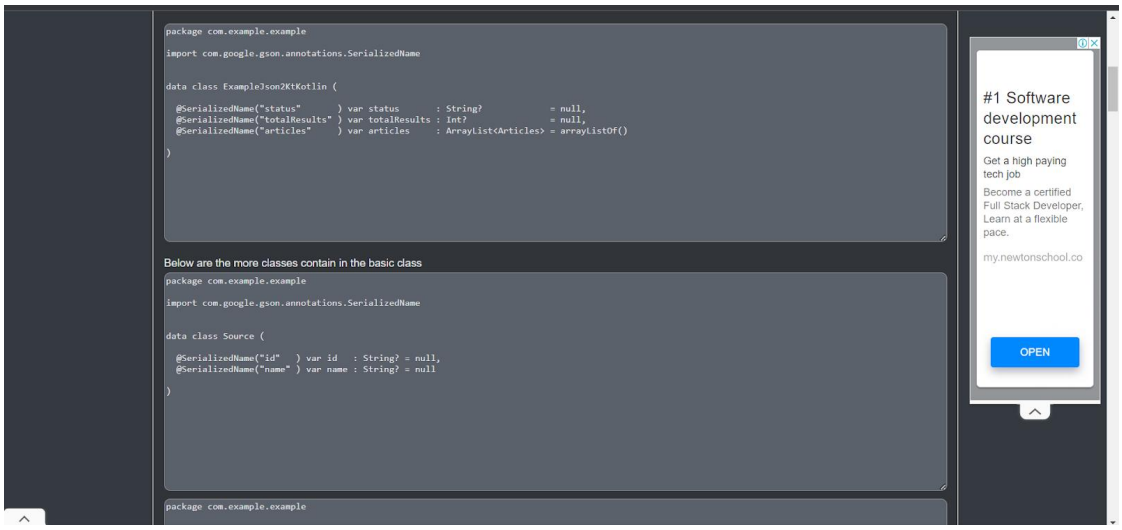
JSON



Copy And Paste The Complete Json File To Json2kt.Com

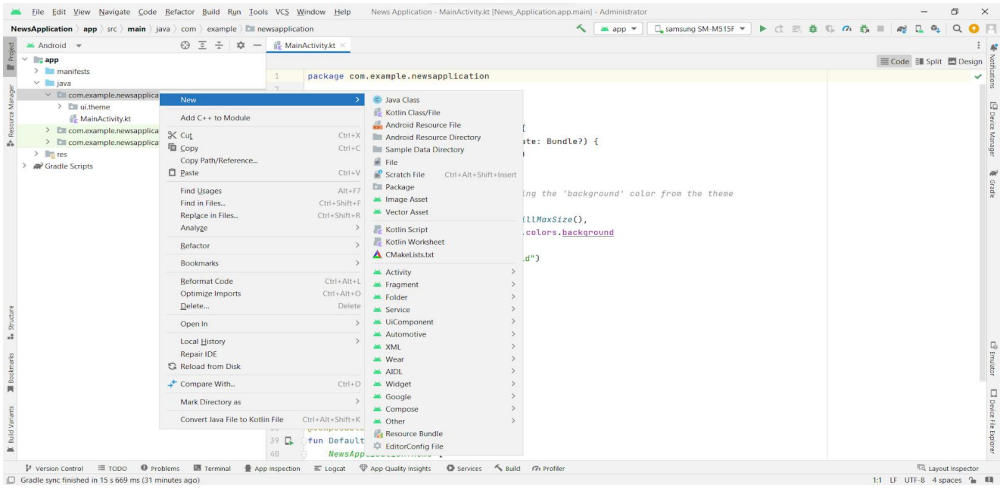


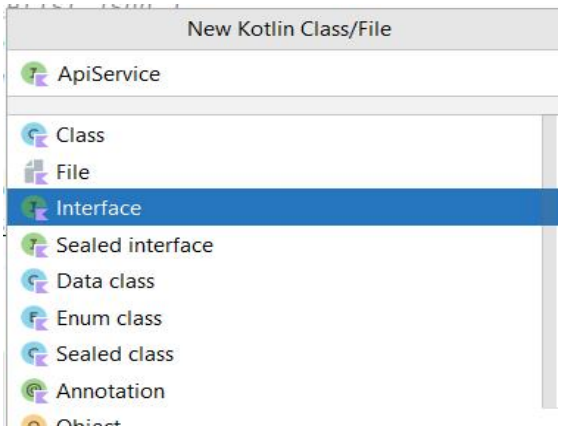
Click on Output Data class



Database For News Integration Into Project

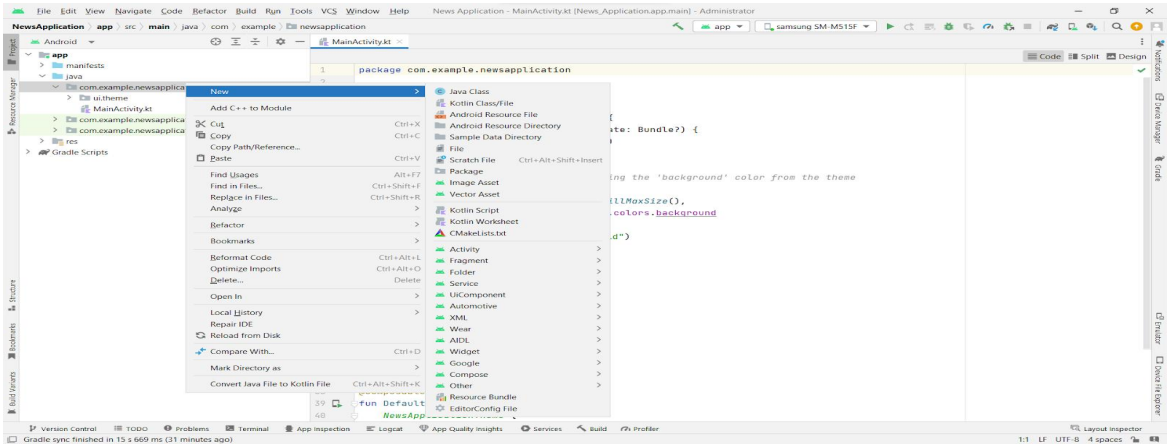
Create ApiService Interface

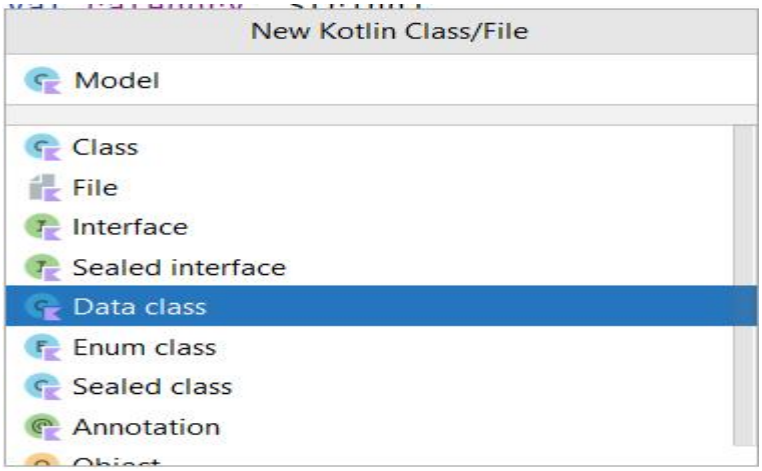





```
package com.example.androidnews
import retrofit2.Retrofit
import retrofit2.converter.gson.GsonConverterFactory
import retrofit2.http.GET
interface ApiService {
   @GET("top-
headlines?country=us&category=business&apiKey=684cb893caf7425abeffad82ac1
d0f4e")
   suspend fun getMovies() :News
   companion object {
      var apiService: ApiService? = null
      fun getInstance() : ApiService {
         if (apiService == null) {
            apiService = Retrofit.Builder()
               // .baseUrl("https://howtodoandroid.com/apis/")
               .baseUrl("https://newsapi.org/v2/")
               //.baseUrl("https://podcast-episodes.p.rapidapi.com/")
               .addConverterFactory(GsonConverterFactory.create())
               .build().create(ApiService::class.java)  }
         return apiService!!
      }}}
```

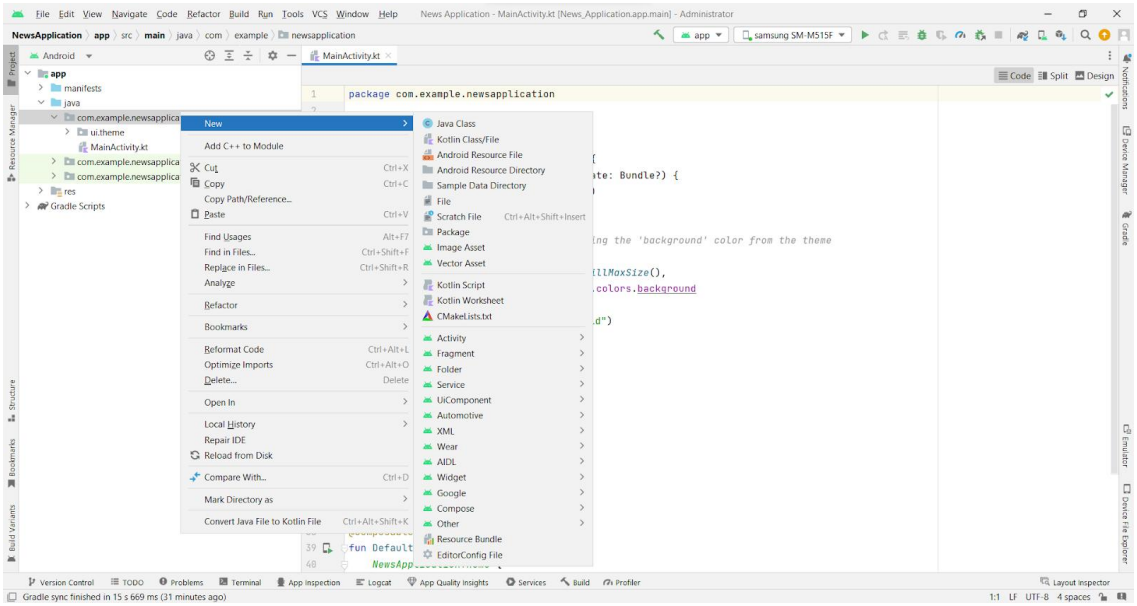
Create Model Data Class

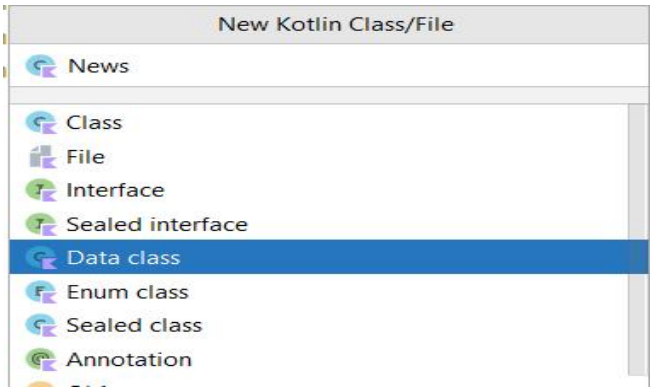





```
package com.example.androidnews
data class Model(val name: String,
val imageUrl: String,
val desc: String,
val category: String)
```

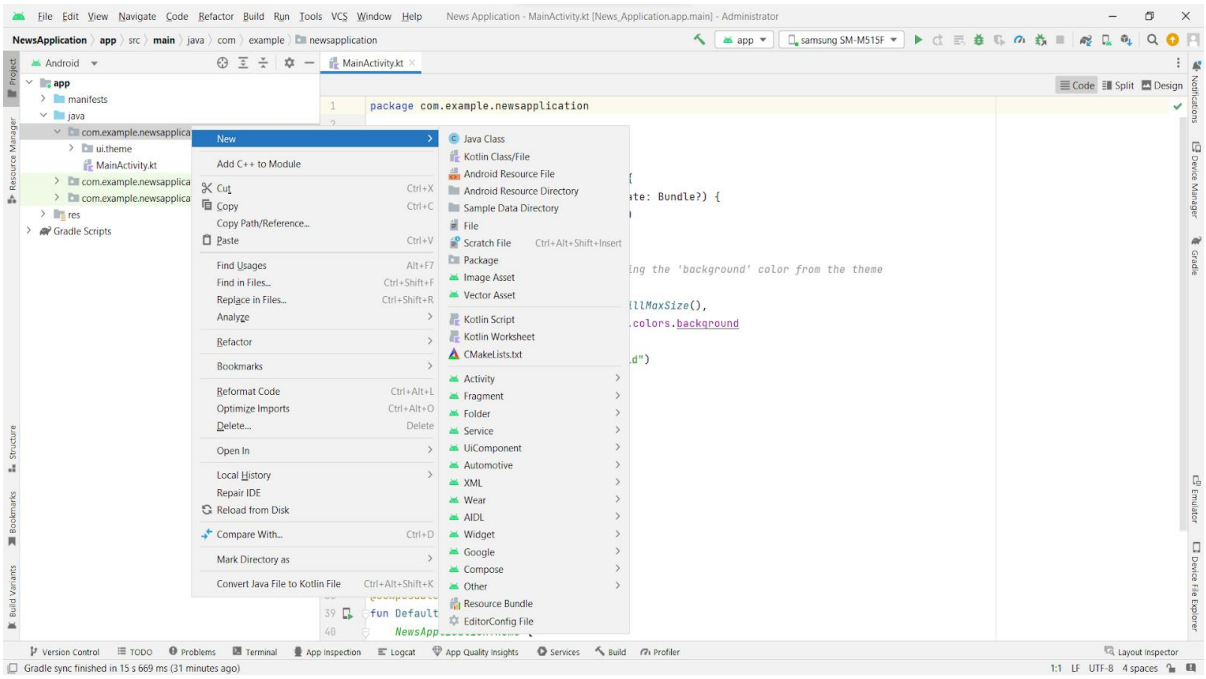
Create News Data Class

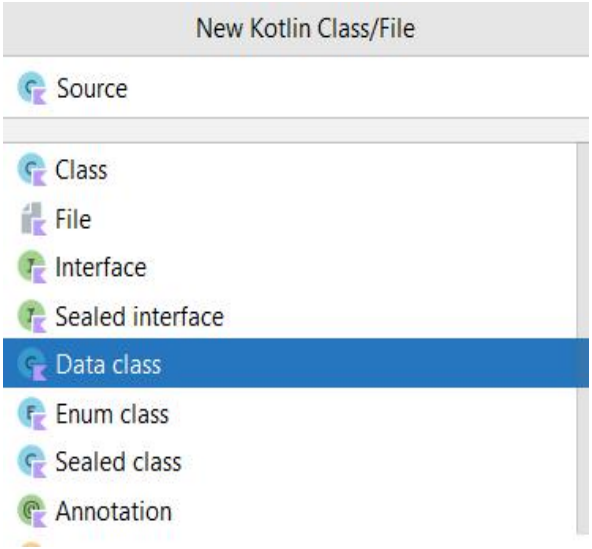




```
package com.example.androidnews
import com.example.androidnews.Articles
import com.google.gson.annotations.SerializedName
data class News(
    @SerializedName("status") var status:String?= null,
    @SerializedName("totalResults") var totalResults : Int?              = null,
    @SerializedName("articles") var articles     : ArrayList<Articles> =
arrayListOf()
)
```

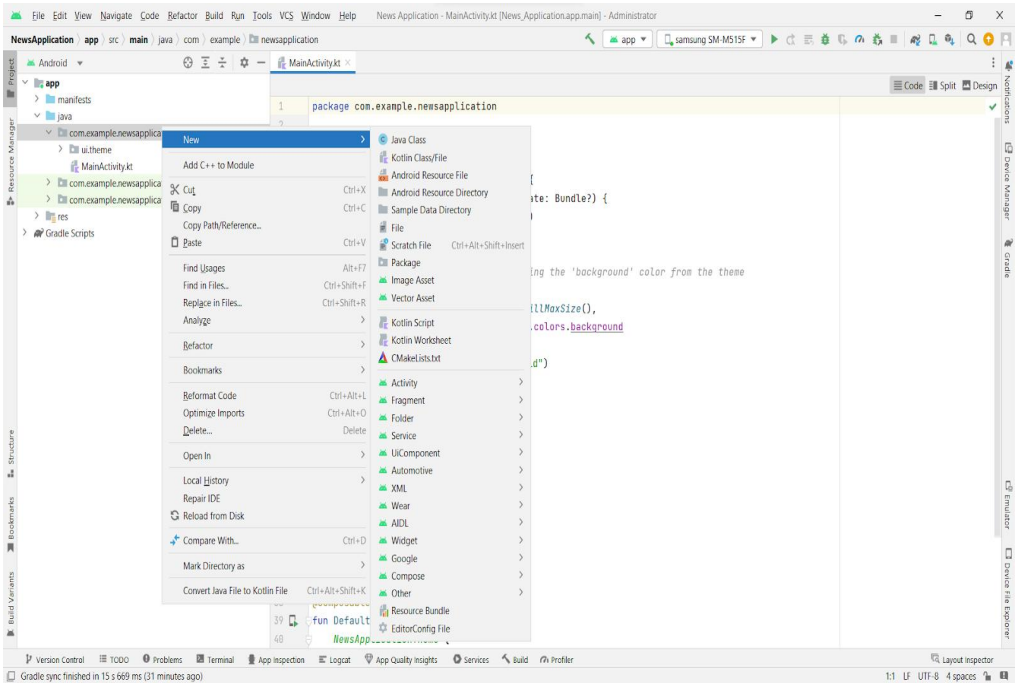
Create Source Data Class

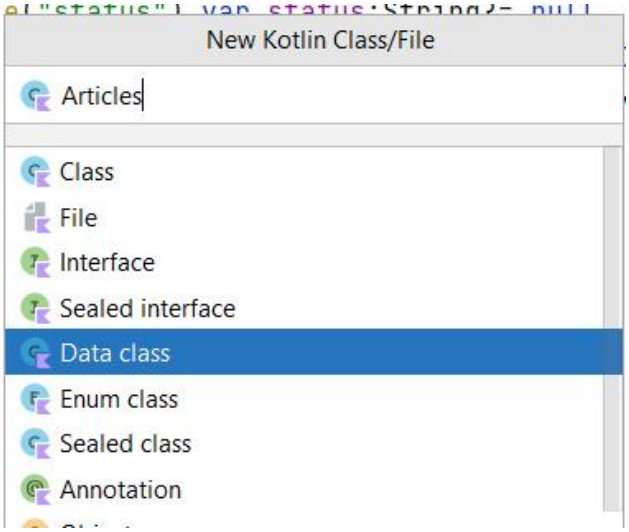




```
package com.example.androidnews
import com.google.gson.annotations.SerializedName
data class Source(
    @SerializedName("id"   ) var id   : String? = null,
    @SerializedName("name" ) var name : String? = null
)
```

Create Articles Data Class

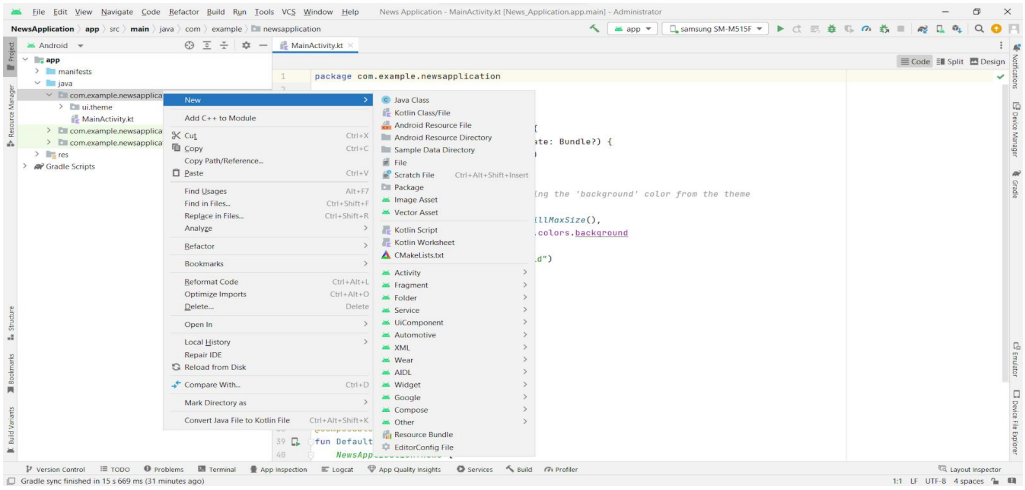


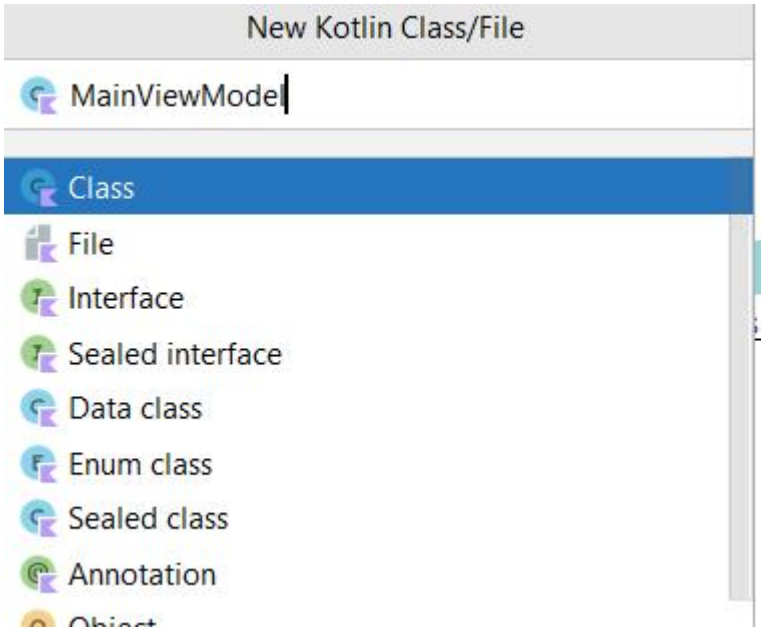



```
package com.example.androidnews
import com.google.gson.annotations.SerializedName
data class Articles(
    @SerializedName("title"       ) var title       : String? = null,
    @SerializedName("description" ) var description : String? = null,
```

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

Create MainViewModel Class

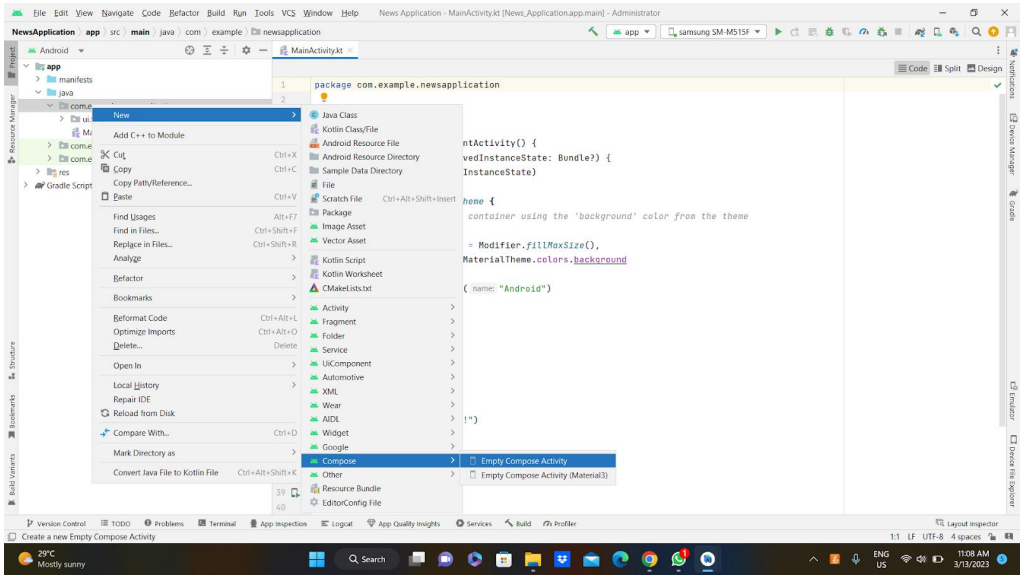


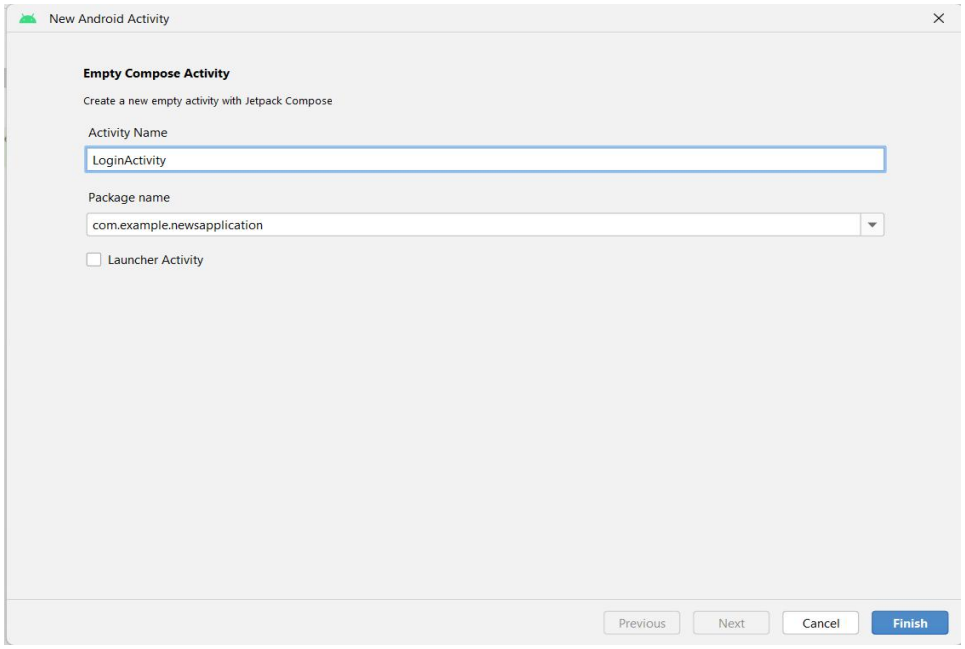



```
package com.example.androidnews
import android.util.Log
import androidx.compose.runtime.getValue
import androidx.compose.runtime.mutableStateOf
import androidx.compose.runtime.setValue
import androidx.lifecycle.ViewModel
import androidx.lifecycle.viewModelScope
import com.example.androidnews.Articles
import kotlinx.coroutines.launch
class MainViewModel  : ViewModel() {
   var movieListResponse:List<Articles> by mutableStateOf(listOf())
   var errorMessage: String by mutableStateOf("")
```

```
    fun getMovieList() {
        viewModelScope.launch {
            val apiService = ApiService.getInstance()
            try {
                val movieList = apiService.getMovies()
                movieListResponse = movieList.articles
            }
            catch (e: Exception) {
                errorMessage = e.message.toString()
            }
        }
    }
}
```

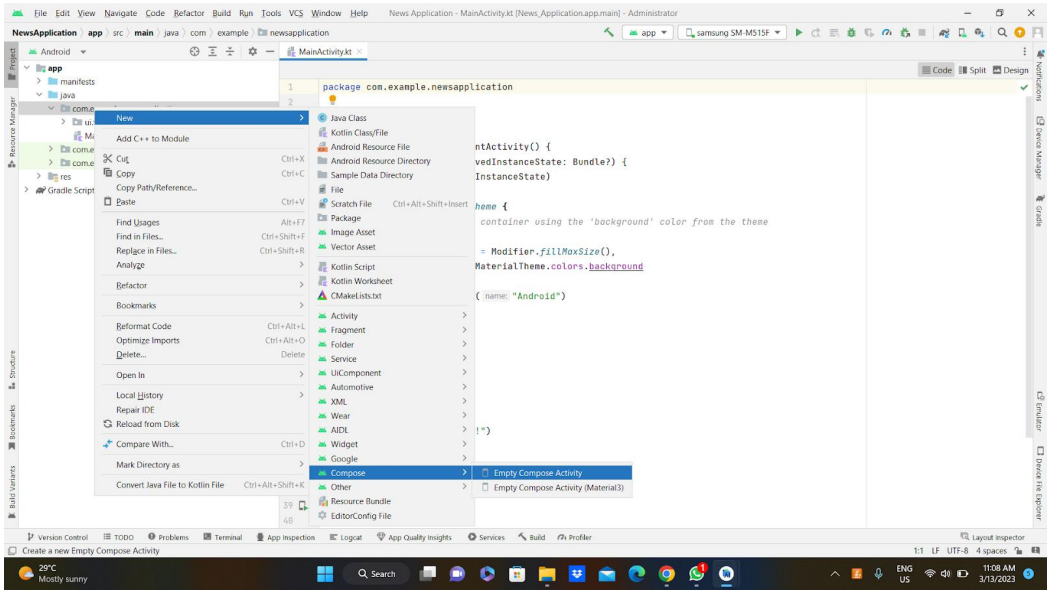
Creating LoginActivity.Kt With Database

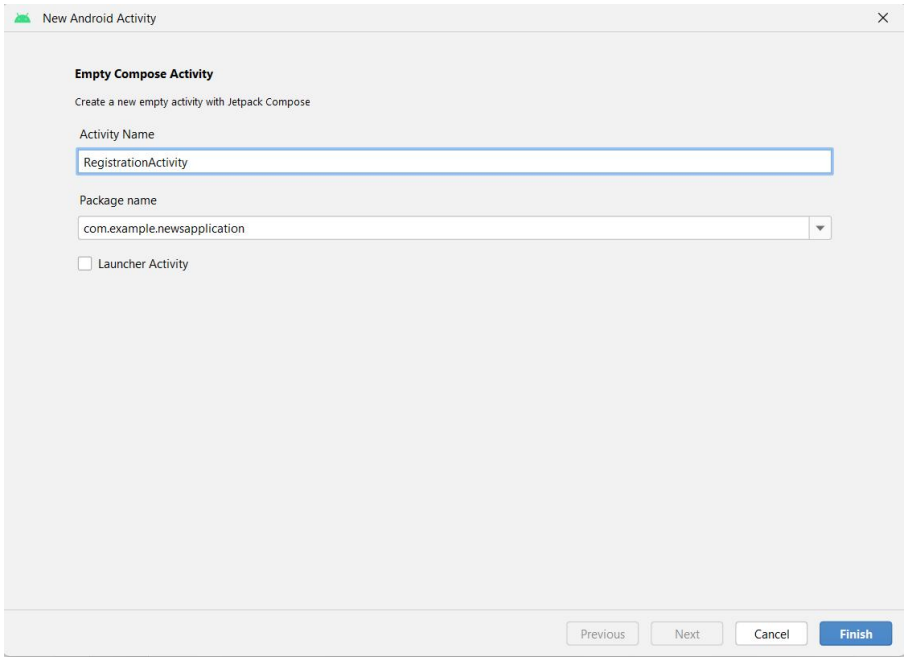




```
package com.example.newsheadlines

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: "") }
```

Creating RegistrationActivity.Kt With Database



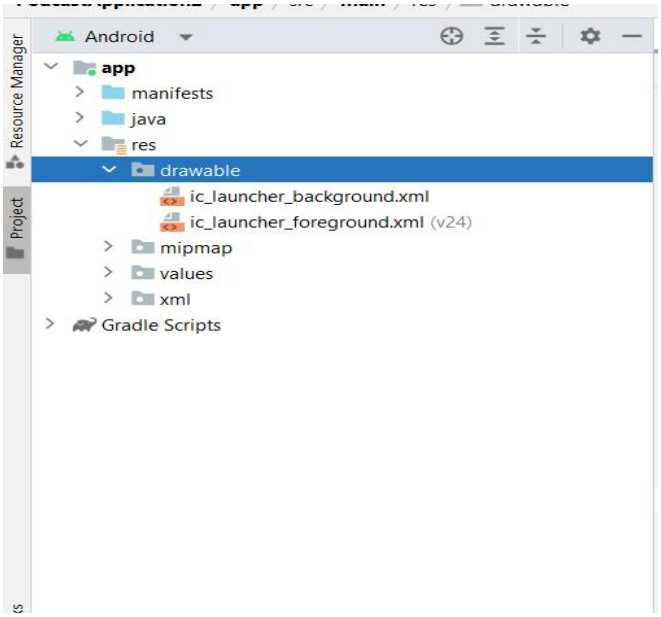


```
package com.example.newsheadlines

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(
```

Creating Mainpage.Kt File

In Mainpage.kt file the main application is developed

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




```
package com.example.androidnews
import android.os.Bundle
import androidx.activity.ComponentActivity
import androidx.activity.compose.setContent
import androidx.compose.foundation.layout.fillMaxSize
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.Composable
import android.content.Context
import android.content.Intent
import android.content.Intent.FLAG_ACTIVITY_NEW_TASK
```

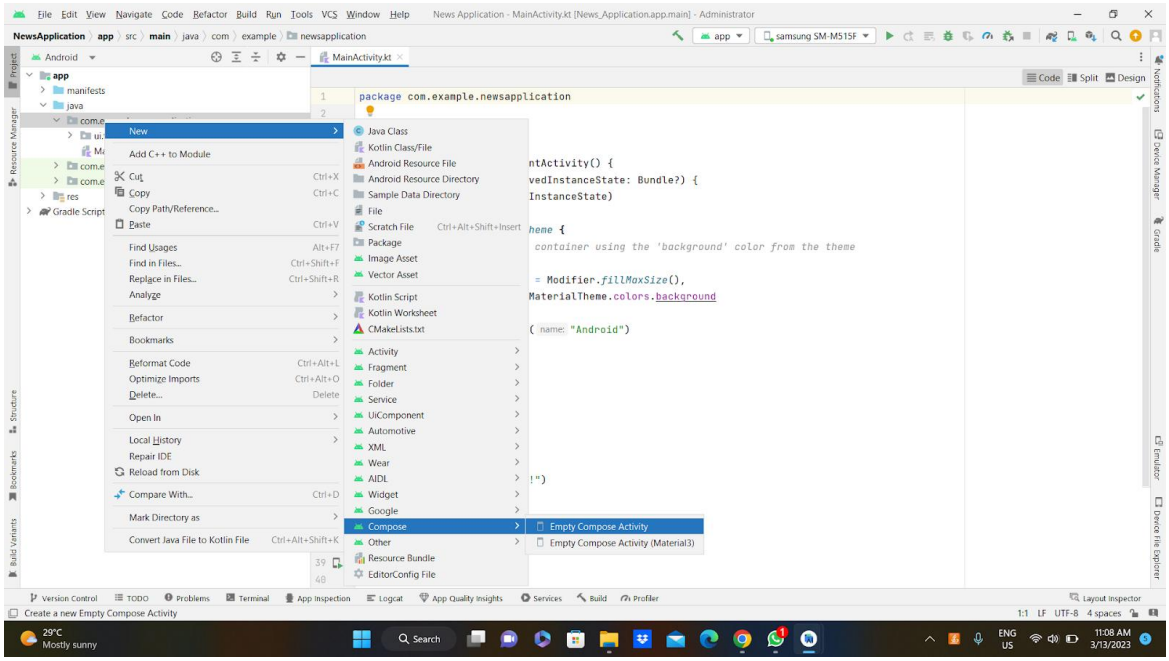
```
import android.util.Log
import android.widget.TextView
import androidx.activity.viewModels
import androidx.compose.material3.CardDefaults
import androidx.compose.material3.CardElevation
import androidx.compose.foundation.Image
import androidx.compose.foundation.background
import androidx.compose.foundation.clickable
import androidx.compose.foundation.layout.*
import androidx.compose.foundation.lazy.LazyColumn
import androidx.compose.foundation.lazy.itemsIndexed
import androidx.compose.foundation.selection.selectable
import androidx.compose.foundation.shape.RoundedCornerShape
import androidx.compose.material3.Card
import androidx.compose.material3.MaterialTheme
import androidx.compose.material3.Surface
import androidx.compose.material3.Text
import androidx.compose.runtime.*
import androidx.compose.ui.Modifier
import androidx.compose.ui.graphics.Color
import androidx.compose.ui.text.font.FontWeight
import androidx.compose.ui.text.style.TextAlign
import androidx.compose.ui.unit.dp
import androidx.compose.ui.unit.sp
import androidx.compose.ui.viewinterop.AndroidView
import androidx.core.text.HtmlCompat
import coil.compose.rememberImagePainter
import coil.size.Scale
import coil.transform.CircleCropTransformation
import androidx.compose.ui.tooling.preview.Preview
import com.example.androidnews.ui.theme.AndroidnewsTheme
class MainPage : ComponentActivity() {
   val mainViewModel by viewModels<MainViewModel>()
   override fun onCreate(savedInstanceState: Bundle?) {
      super.onCreate(savedInstanceState)
      setContent {
         AndroidnewsTheme {
            // A surface container using the 'background' color from the theme
            Surface(color = MaterialTheme.colorScheme.background) {
               Column() {
                  Text(text = "Latest NEWS", fontSize = 32.sp, modifier =
Modifier.fillMaxWidth(), textAlign = TextAlign.Center)
                  MovieList(applicationContext, movieList =
mainViewModel.movieListResponse)
                  mainViewModel.getMovieList()
               } } }}
}
```

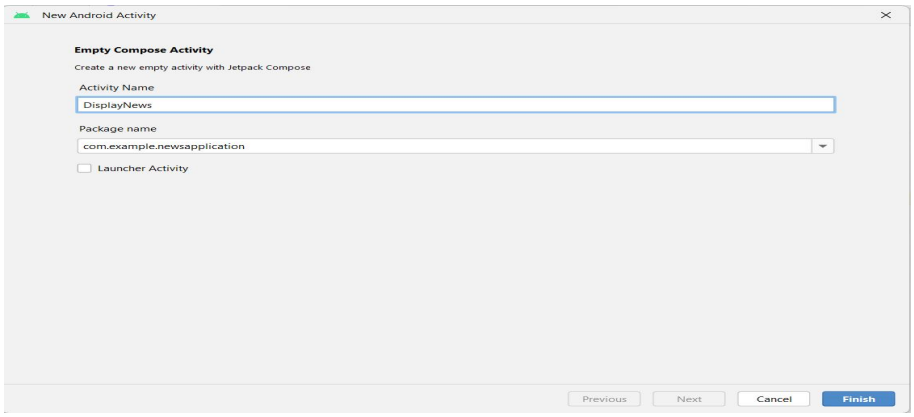
```
@Composable
fun MovieList(context: Context, movieList: List<Articles>) {
   var selectedIndex by remember { mutableStateOf(-1) }
   LazyColumn {
      itemsIndexed(items = movieList) {
            index, item ->
         MovieItem(context,movie = item, index, selectedIndex) { i ->
            selectedIndex = i
         } } }}
@Composable
fun MovieItem(context: Context) {
   val movie = Articles(
      "Coco",
      "",
      " articl"
   )
   MovieItem(context,movie = movie, 0, 0) { i ->
      Log.i("wertytest123abc", "MovieItem: "
            +i)}}
@Composable
fun MovieItem(context: Context, movie: Articles, index: Int, selectedIndex: Int,
          onClick: (Int) -> Unit)
{
   val backgroundColor = if (index == selectedIndex)
MaterialTheme.colorScheme.primary else MaterialTheme.colorScheme.background
   Card(
      modifier = Modifier
         .padding(8.dp, 4.dp)
         .fillMaxSize()
         .selectable(true, true, null,
            onClick = {
               Log.i("test123abc", "MovieItem: $index/n$selectedIndex")
            })
         .clickable { onClick(index) }
         .height(180.dp), shape = RoundedCornerShape(8.dp), elevation =
CardDefaults.cardElevation(defaultElevation = 4.dp)
   ) {
      Surface(color = Color.White) {

         Row(
            Modifier
               .padding(4.dp)
               .fillMaxSize()  )
         {
            Image(
               painter = rememberImagePainter(
```

```
                data = movie.urlToImage,
                builder = {
                    scale(Scale.FILL)
                    placeholder(R.drawable.placeholder)
                    transformations(CircleCropTransformation())
                }
            ),
            contentDescription = movie.description,
            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(true, true, null,
                    onClick = {
                        Log.i("test123abc", "MovieItem: $index/n${movie.description}")
                        context.startActivity(
                            Intent(context, DisplayNews::class.java)
                                .setFlags(Intent.FLAG_ACTIVITY_NEW_TASK)
                                .putExtra("desk", movie.description.toString())
                                .putExtra("urlToImage", movie.urlToImage)
                                .putExtra("title", movie.title)
                        )
                    })
        ) {
            Text(
                text = movie.title.toString(),
                style = MaterialTheme.typography.labelLarge,
                fontWeight = FontWeight.Bold
            )
            HtmlText(html = movie.description.toString())
        }
    }
  }
}
 @Composable
fun HtmlText(html: String, modifier: Modifier = Modifier) {
    AndroidView(
        modifier = modifier.fillMaxSize() .size(33.dp),
        factory = { context -> TextView(context) },
```

```
            update = { it.text = HtmlCompat.fromHtml(html,
HtmlCompat.FROM_HTML_MODE_COMPACT) }
        )
    }
}
```

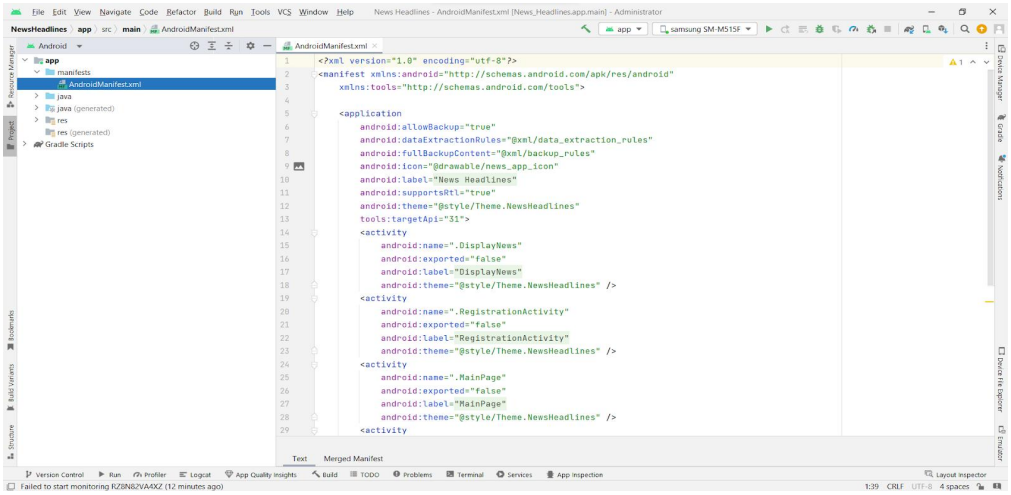
Creating DisplayNews.Kt File



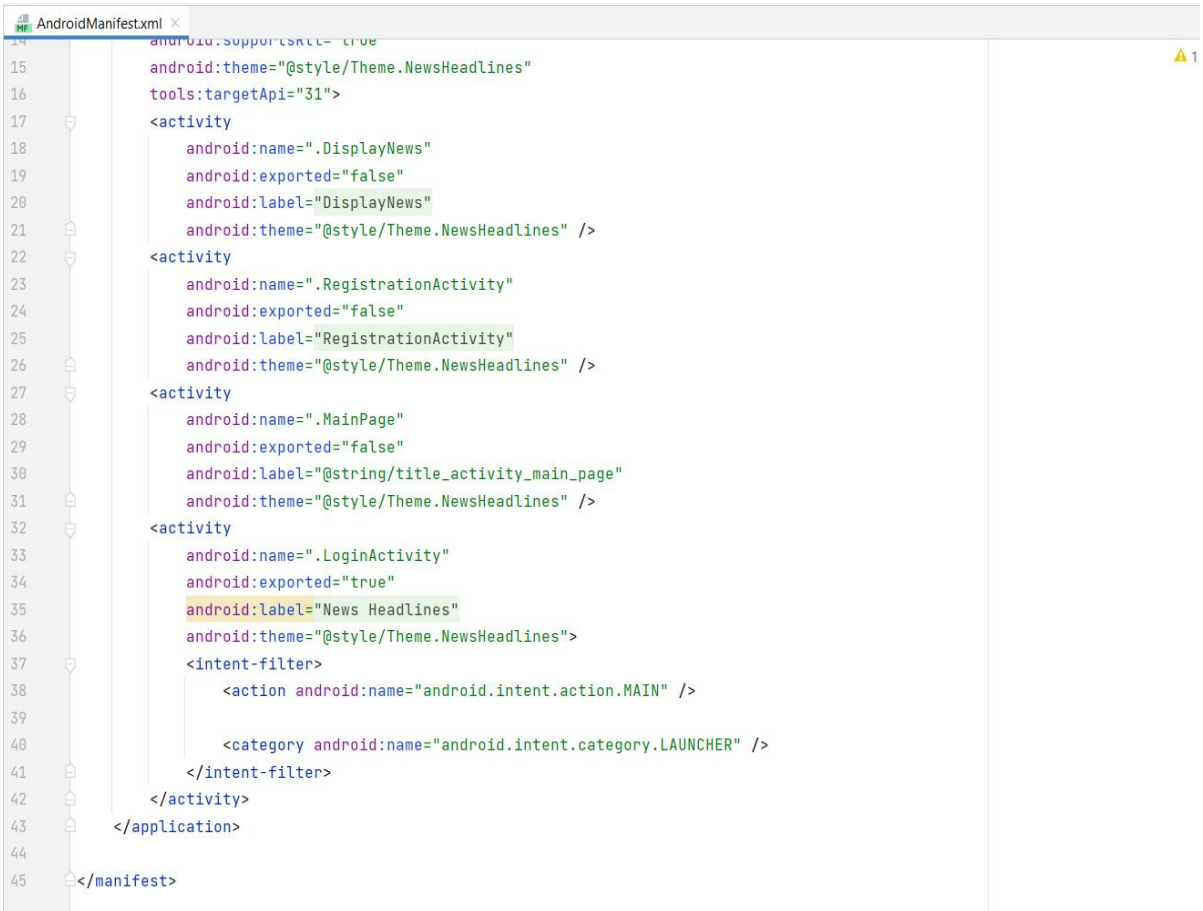



```
class DisplayNews : ComponentActivity() {
    override fun onCreate(savedInstanceState: Bundle?) {
        super.onCreate(savedInstanceState)
        setContent {
            NewsHeadlinesTheme {
                // A surface container using the 'background' color from the theme
                Surface(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

                    var desk = getIntent().getStringExtra( name: "desk")
                    var title = getIntent().getStringExtra( name: "title")
                    var urlImage = 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(urlImage),
                            contentDescription = "My content description",
                        )
                    }
                    //   Greeting(desk.toString())
```

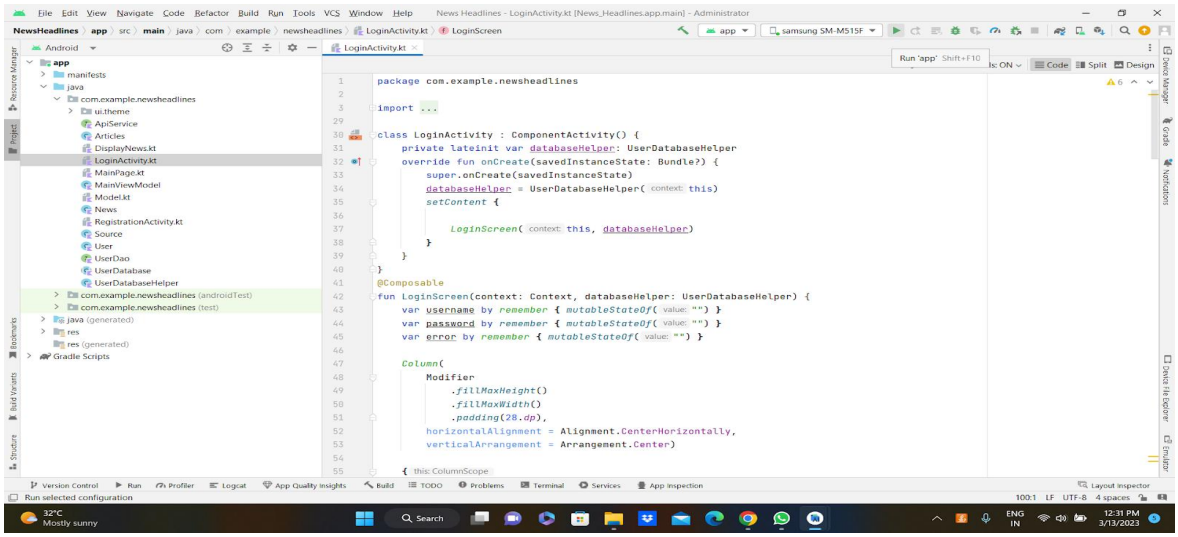
Modifying AndroidManifest.Xml



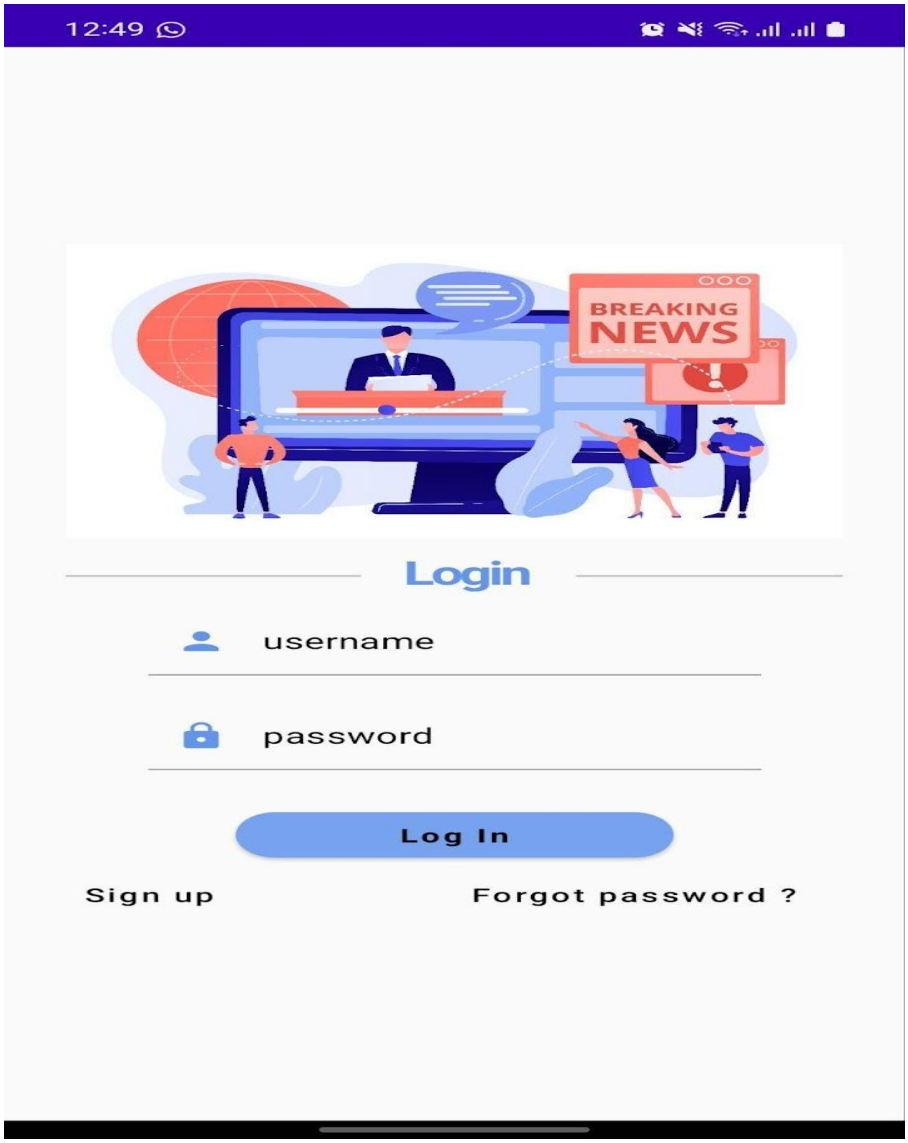
When we run the app we will get the MainActivity.kt file as our first screen , but we want LoginActivity.kt , So we need to change in AndroidManifest.xml.



Run The Application In Mobile


File Edit View Navigate Code Refactor Build Run Tools VCS Window Help
News Headlines - LoginActivity.kt [News_Headlines.app.main] - Administrator
NewsHeadlines app src main java com example newsheadlines LoginActivity.kt LoginScreen
Android
app
manifests
java
com.example.newsheadlines
ui.theme
ApiService
Articles
DisplayNews.kt
LoginActivity.kt
MainPage.kt
MainViewModel
Model.kt
News
RegistrationActivity.kt
Source
User
UserDao
UserDatabase
UserDatabaseHelper
com.example.newsheadlines (androidTest)
com.example.newsheadlines (test)
java (generated)
res
res (generated)
Gradle Scripts
LoginActivity.kt
Run 'app' Shift+F10
Code Split Design
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: "") }
Column(
Modifier
.fillMaxHeight()
.fillMaxWidth()
.padding(28.dp),
horizontalAlignment = Alignment.CenterHorizontally,
verticalArrangement = Arrangement.Center)
{ this: ColumnScope
Version Control Run Profiler Logcat App Quality Insights Build TODO Problems Terminal Services App Inspection Layout Inspector
Run selected configuration
100:1 LF UTF-8 4 spaces
32°C Mostly sunny
Search
ENG IN
12:31 PM 3/13/2023

Output:


12:49
BREAKING NEWS
Login
username
password
Log In
Sign up
Forgot password ?

12:49

Sign Up

username

password

email

Register

Have an account? Log in

12:49

Latest NEWS

Top Crypto Trader Issues Bitcoin Warning, Says BTC Flashing Vibes of March 2020 Meltdown - The Daily Hodl

A closely followed crypto trader is

Saudi oil giant Aramco posts record $161.1 billion profit for 2022 - CNBC

Saudi state-controlled Aramco achieved a record profit last year, boosted by higher energy prices

Silicon Valley Bank: the spectacular unravelling of the tech industry's banker - Financial Times

News, analysis and comment from the Financial Times, the

Meta looking into new social platform that could rival Twitter - The Hill

Meta, Facebook's parent company, is looking into a new social media platform that could

Crypto Exhales as USDC Stablecoin Rebounds

12:49

Top Crypto Trader Issues Bitcoin Warning, Says BTC Flashing Vibes of March 2020 Meltdown - The Daily Hodl

A closely followed crypto trader is issuing an alert to Bitcoin (BTC) holders, saying that the king crypto's current market structure looks similar to its price action prior to a deep plunge about three years ago.