

Project Manual

Date	06 November 2023
Team ID	SI-GuidedProject-587104-1696656485
Project Name	An Android Application for Keeping Up with the Latest Headlines
Team Members	Aarush Jain, Hemantha Kumar Karthikeya, Sanjana Sivakumar, Siddhi Kadam
Maximum Marks	

Android Application for Keeping Up with the Latest Headlines

In today's era of the internet, a multitude of news articles and gossip pieces are easily accessed, by a simple click of a button. But how many of these news articles are written by verified news outlets, and how many sources of the gossip pieces have verifiable sources? Sometimes, the amount of unorganized news floating through the web seems to be too much to handle, and this discourages people from even knowing the events occurring all around the world.

An Android application for keeping up with the latest headlines, can be just the solution for this problem. The application organizes news depending on the news outlets, and the topics it contains. You can now read news straight from your most dependable news outlets.

The application's main feature is to fetch news articles from different news outlets. The Home Page, News Page and Specific News Page are all scrollable. The app is available in dark mode, and in light mode. The app also stores user data when they first sign up.

Architecture

Learning Outcomes

Once you have successfully completed creating an android application that allows users to keep up with the latest headlines, you will be able to:

- Work on Android Studio to create any application
- Integrate databases to your application to store data
- Integrate API's into your application

Project Workflow

1. First, the user will open the app's login page and type in their login information (username and password). This information is stored in the database.
2. If the user does not have an account, they can click on "Not Registered? Click Here". This will navigate them to the sign up page, where they can type their username, password and email account. Once they have signed up, they will be redirected to the log-in page. If the user already has an account, they can click on "Already Registered? Click Here" to redirect them to the login page.
3. After logging in successfully, the user is directed to the Home Page. Here, there is a number of news outlets that the user can choose to get their news from.
4. Once they have selected a news outlet, they will be directed to the News Page. Here, there are a variety of news pieces from said news outlet.
5. When the user has selected a news piece, they will be directed to a Specific News Page, where a detailed essay about the new piece that they have chosen will be visible.

TASKS

- 1. Start with the essential first actions.**
- 2. Begin a new project.**
- 3. Include necessary dependencies.**
- 4. Set up authorization and access permissions.**
- 5. Develop the database structure.**
- 6. Establish an API Service and the relevant classes to incorporate external services.**
- 7. Construct the user interface of the application and establish connections with the database.**
- 8. Make adjustments to the AndroidManifest.xml file.**
- 9. Execute the application to test it.**

TASK 1: Start with essential first actions

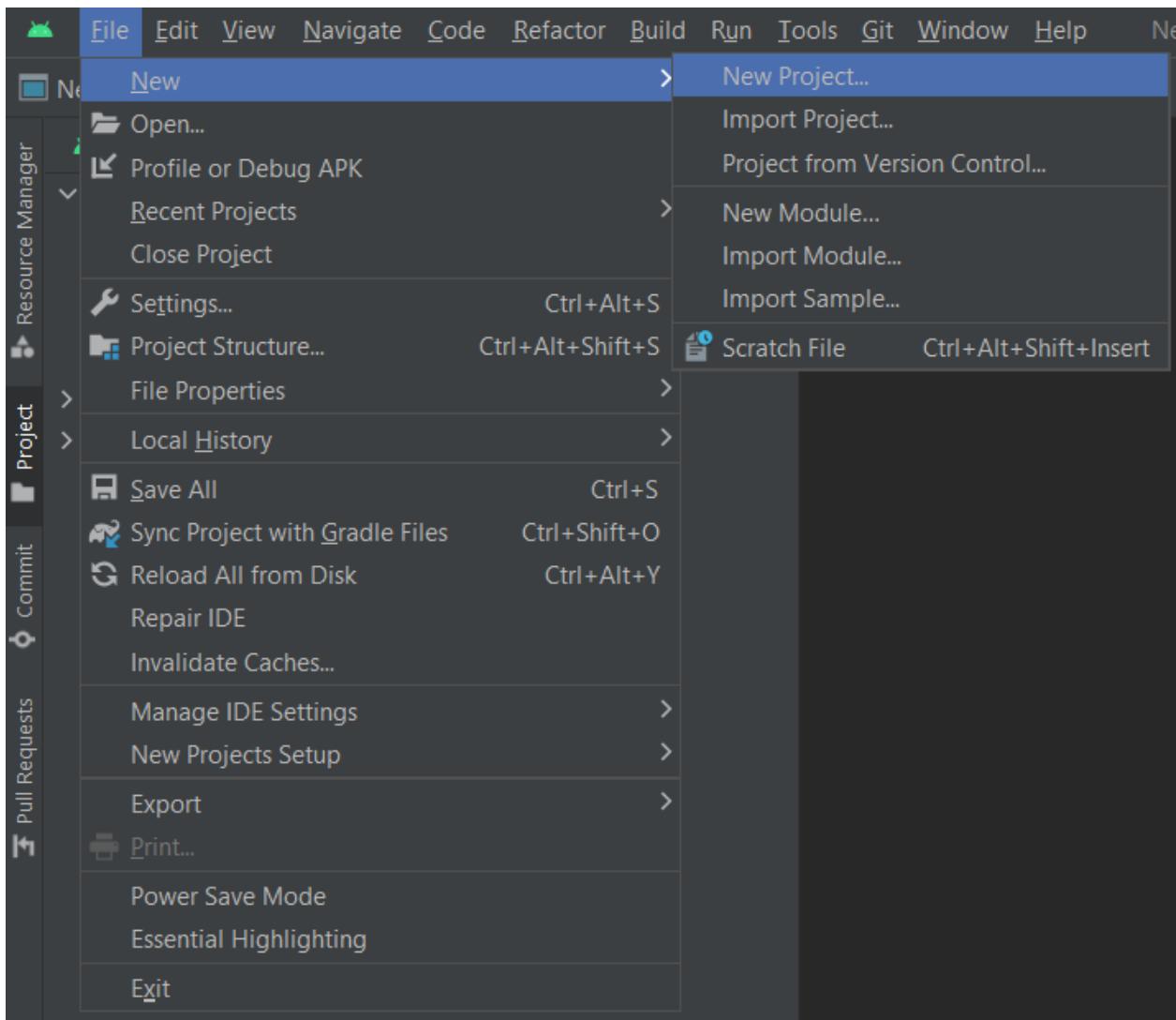
1. Download the latest version of Android Studio.

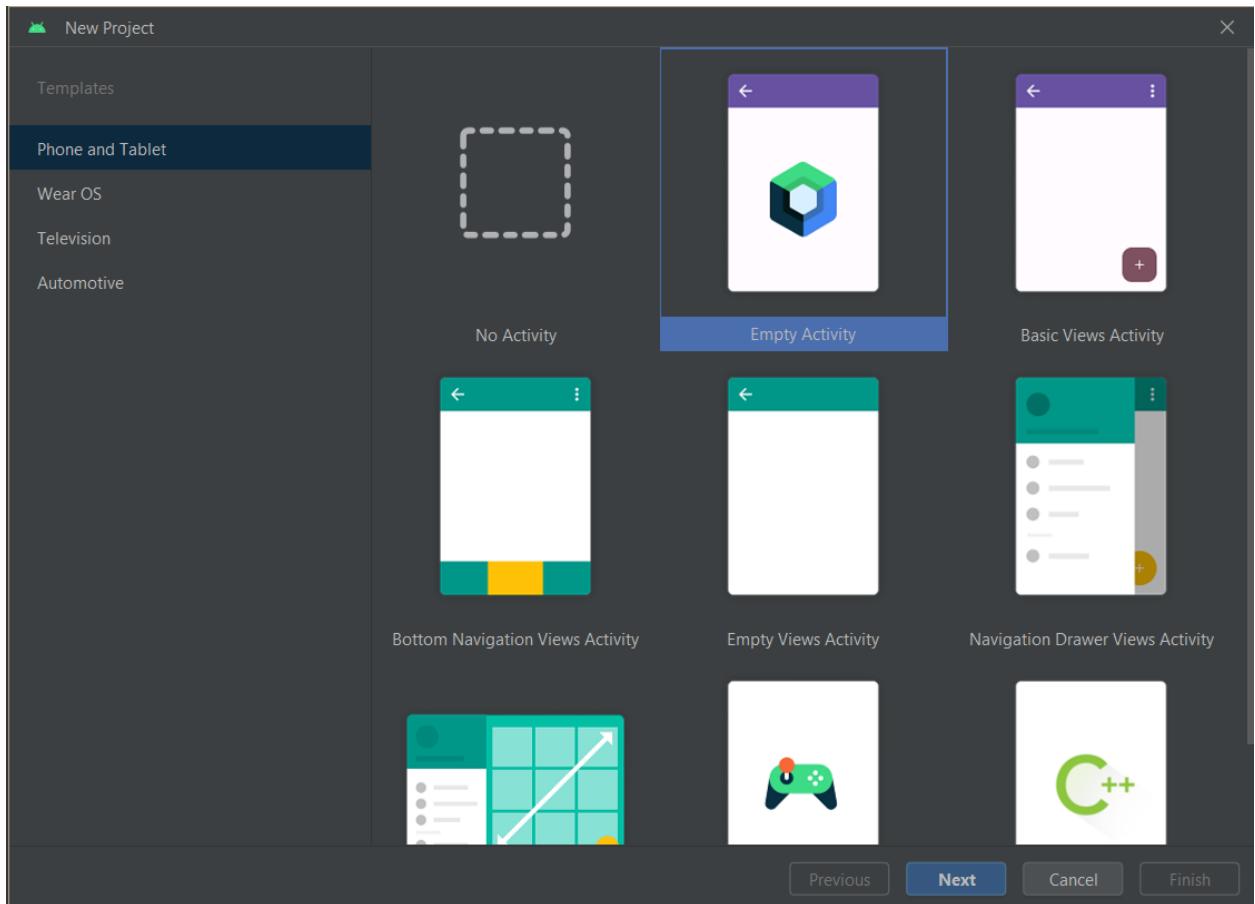
<https://developer.android.com/studio>

2. Open Android Studio.

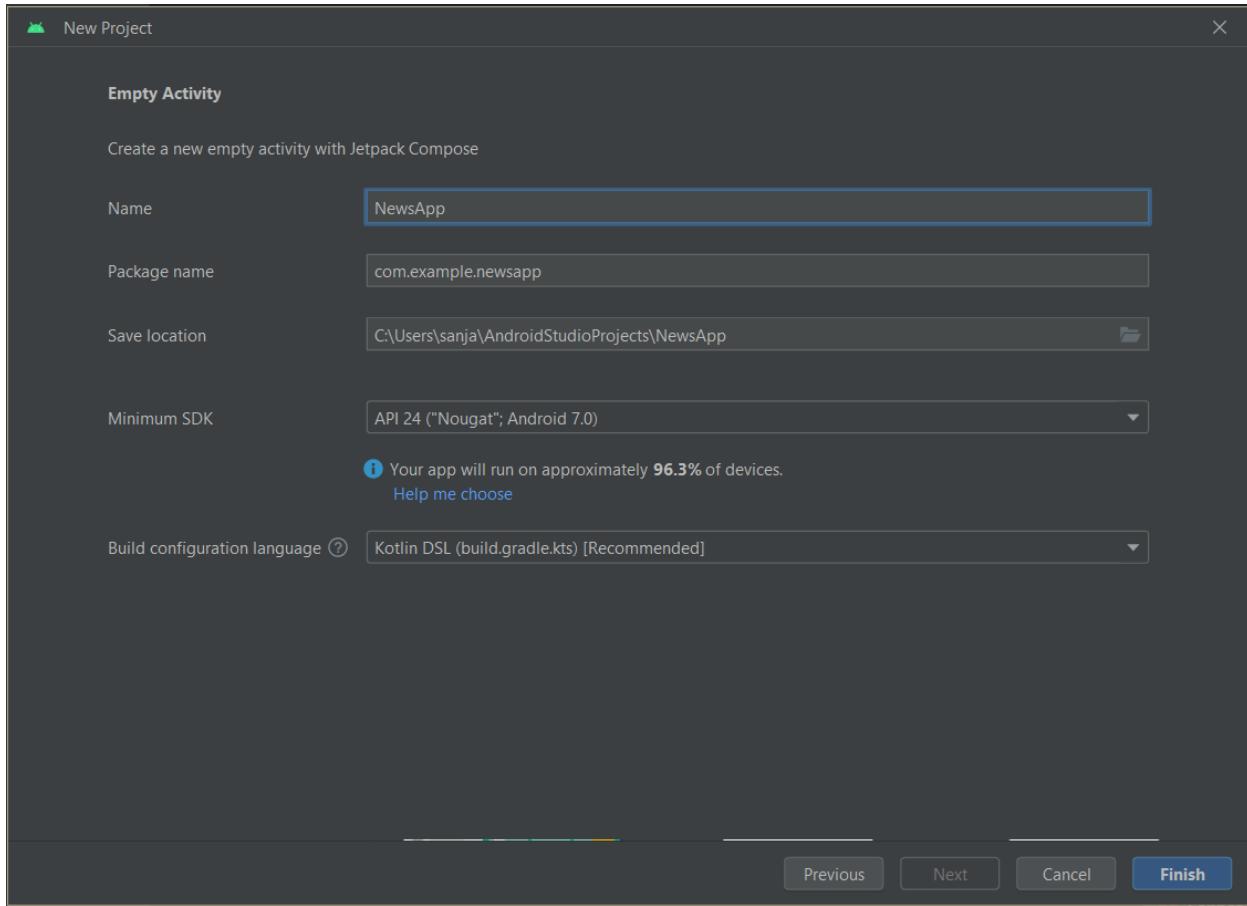
TASK 2: Begin a new project

1. In Android Studio, go to File, New, New Project and then Empty Activity.





2. Give a name to the project and give the minimum SDK value.



TASK 3: Include necessary dependencies

1. Go to Gradle Scripts on the left panel, and go to build.gradle(Module : app).
2. Add the required dependencies.
3. Click on Sync now.

```
build.gradle (:app) X
1  plugins {
2      id 'com.android.application'
3      id 'dagger.hilt.android.plugin'
4      id 'org.jetbrains.kotlin.android'
5      id 'kotlin-android'
6      id 'kotlin-kapt'
7  }
8
9  android {
10     namespace 'com.aarush.newsapp'
11     compileSdk 33
12
13     defaultConfig {
14         applicationId "com.aarush.newsapp"
15         minSdk 24
16         targetSdk 33
17         versionCode 1
18         versionName "1.0"
19
20         testInstrumentationRunner "androidx.test.runner.AndroidJUnitRunner"
21     }
22
23     buildTypes {
24         release {
25             minifyEnabled false
26             proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
27         }
28     }
29
30     compileOptions {
31         ...
32     }
33 }
```

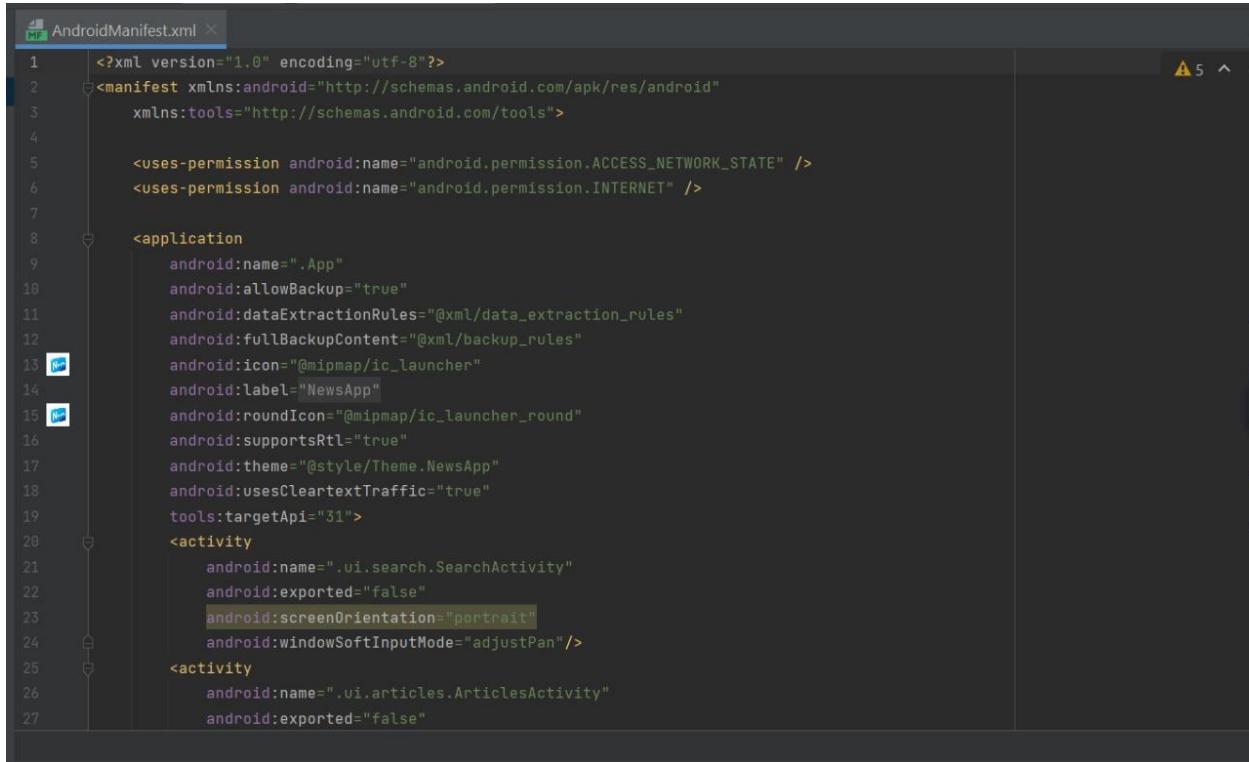
```
build.gradle (app) X

25        minifyEnabled false
26        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
27    }
28 }
29 compileOptions {
30     sourceCompatibility JavaVersion.VERSION_1_8
31     targetCompatibility JavaVersion.VERSION_1_8
32 }
33 kotlinOptions {
34     jvmTarget = '1.8'
35 }
36 buildFeatures {
37     viewBinding true
38 }
39 }
40
41 apply from: '../shared_dependencies.gradle'
42
43 dependencies {
44     implementation project(":core")
45     implementation 'androidx.appcompat:appcompat:1.6.1'
46     implementation 'com.google.android.material:material:1.5.0'
47     implementation 'androidx.constraintlayout:constraintlayout:2.1.4'
48     implementation 'androidx.test.ext:junit-ktx:1.1.5'
49     implementation 'androidx.test:monitor:1.6.1'
50     androidTestImplementation 'junit:junit:4.12'
51     androidTestImplementation 'junit:junit:4.12'
52 }

dependencies{}
```

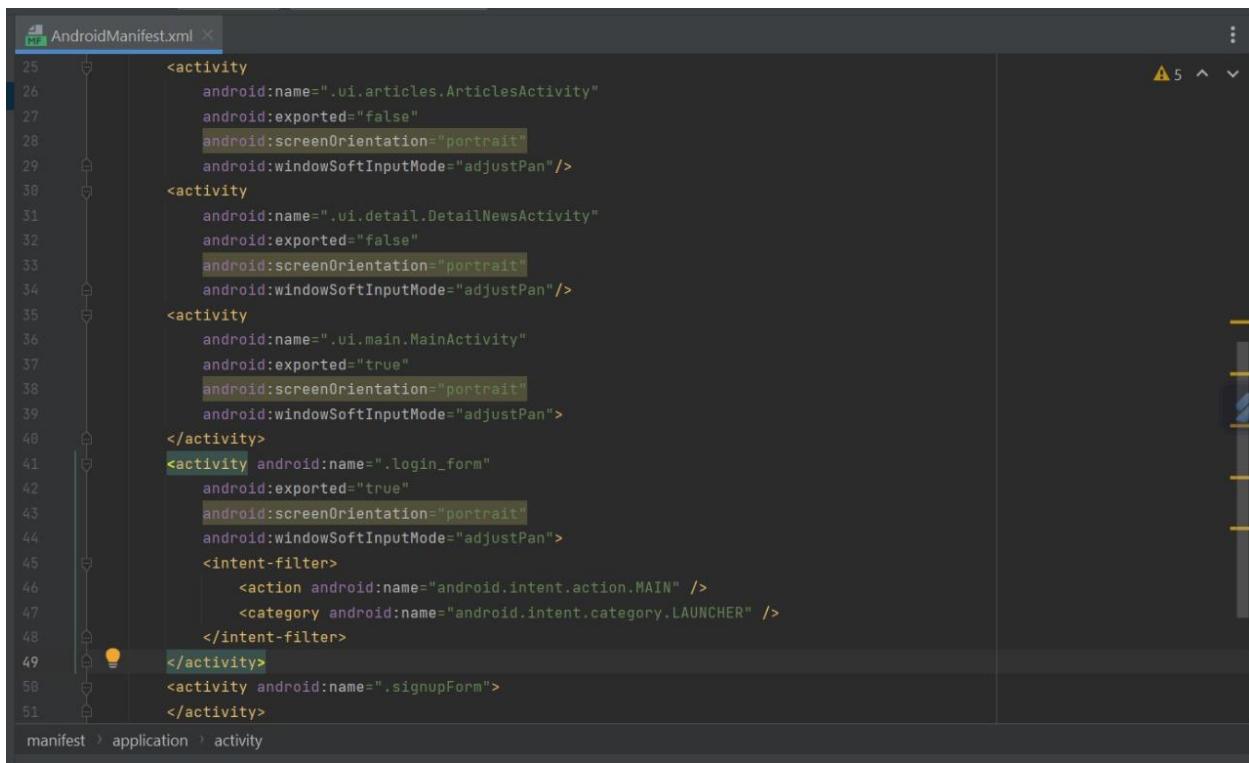
TASK 4: Set up authorization and access permissions

1. Go to app on the left panel, go to manifests, and click on AndroidManifest.xml.
2. Add permissions.



```

1  <?xml version="1.0" encoding="utf-8"?>
2  <manifest xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:tools="http://schemas.android.com/tools">
4
5      <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
6      <uses-permission android:name="android.permission.INTERNET" />
7
8      <application
9          android:name=".App"
10         android:allowBackup="true"
11         android:dataExtractionRules="@xml/data_extraction_rules"
12         android:fullBackupContent="@xml/backup_rules"
13         android:icon="@mipmap/ic_launcher"
14         android:label="NewsApp"
15         android:roundIcon="@mipmap/ic_launcher_round"
16         android:supportsRtl="true"
17         android:theme="@style/Theme.NewsApp"
18         android:usesCleartextTraffic="true"
19         tools:targetApi="31">
20             <activity
21                 android:name=".ui.search.SearchActivity"
22                 android:exported="false"
23                 android:screenOrientation="portrait"
24                 android:windowSoftInputMode="adjustPan"/>
25             <activity
26                 android:name=".ui.articles.ArticlesActivity"
27                 android:exported="false"
28             </activity>
29         </application>
30     </manifest>
31 
```



```

25         <activity
26             android:name=".ui.articles.ArticlesActivity"
27             android:exported="false"
28             android:screenOrientation="portrait"
29             android:windowSoftInputMode="adjustPan"/>
30         <activity
31             android:name=".ui.detail.DetailNewsActivity"
32             android:exported="false"
33             android:screenOrientation="portrait"
34             android:windowSoftInputMode="adjustPan"/>
35         <activity
36             android:name=".ui.main.MainActivity"
37             android:exported="true"
38             android:screenOrientation="portrait"
39             android:windowSoftInputMode="adjustPan">
40         </activity>
41         <activity android:name=".login_form"
42             android:exported="true"
43             android:screenOrientation="portrait"
44             android:windowSoftInputMode="adjustPan">
45             <intent-filter>
46                 <action android:name="android.intent.action.MAIN" />
47                 <category android:name="android.intent.category.LAUNCHER" />
48             </intent-filter>
49         </activity>
50         <activity android:name=".signupForm">
51             </activity>
52     </manifest>
53 
```

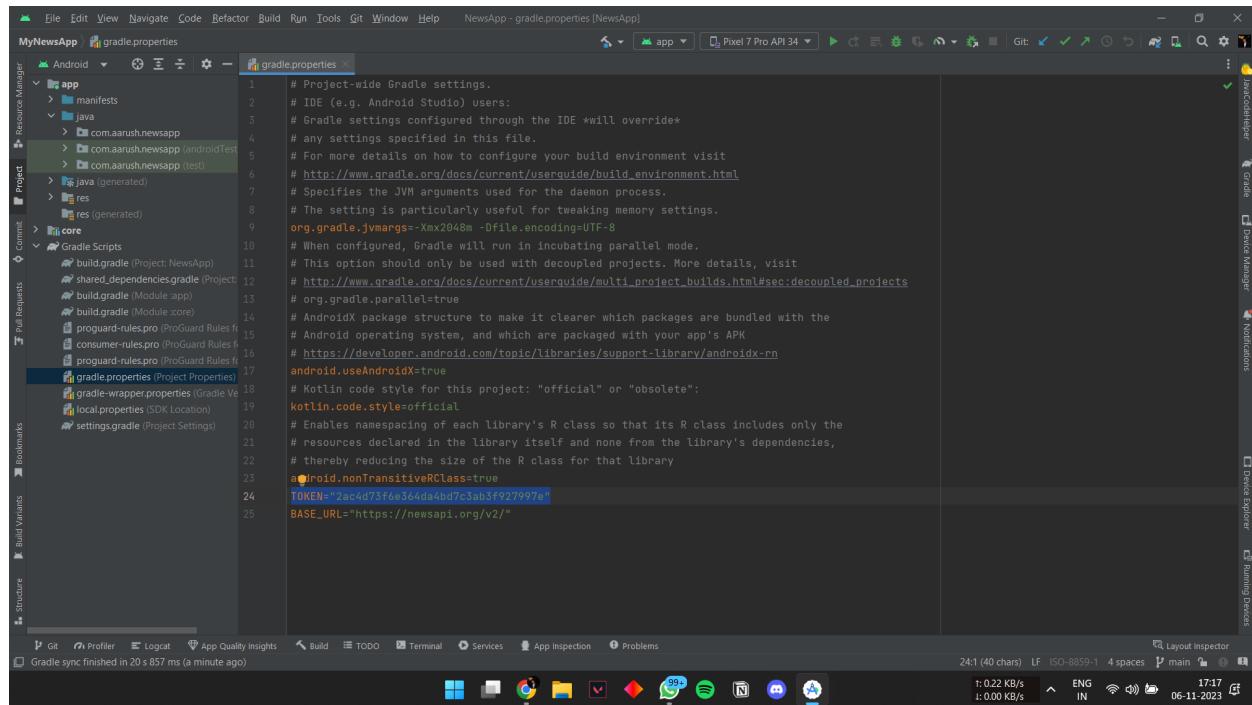
TASK 5: Develop the database structure

1. In your XML layout files, you design the user interface components for the login and registration screens. You create input fields for the username and password, along with buttons to handle login and registration actions. These XML files define the visual structure of your app, allowing users to interact with the authentication system.
2. You implement the SQLite database in the backend using Java. You create a DatabaseHelper class that extends SQLiteOpenHelper. This class manages the database creation, version management, and provides methods for performing CRUD (Create, Read, Update, Delete) operations on the user data. Within this class, you define the database schema, including the table structure for storing usernames and passwords securely.
3. In SQLite, the database is stored locally on your device. DB_class stores data (from signup page) in a relational database

TASK 6: Establish an API Service and the relevant classes to incorporate external services

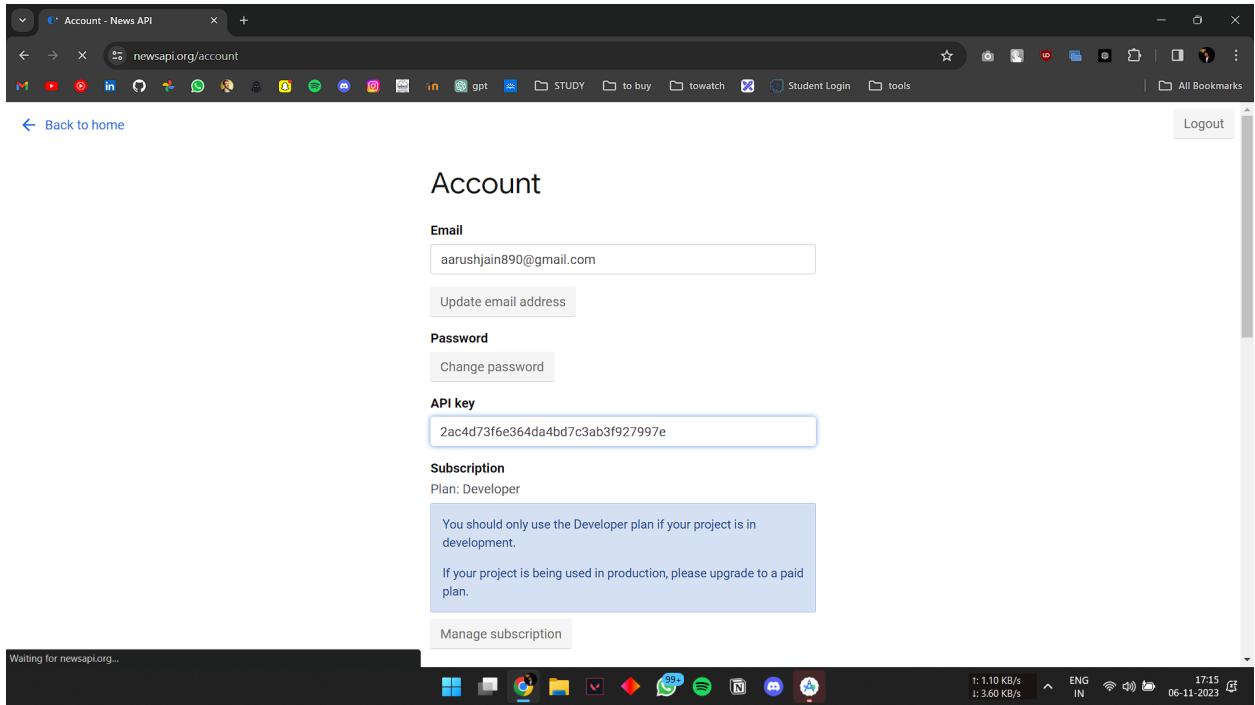
<https://github.com/smartzinternz02/SI-GuidedProject-587104-1696656485/releases/tag/apk1.0.1>

1. ArticlesViewModel.kt, ArcticlesActivity.kt and ArticlesAdapter.kt are used for the API to fetch news articles and data from respective news articles



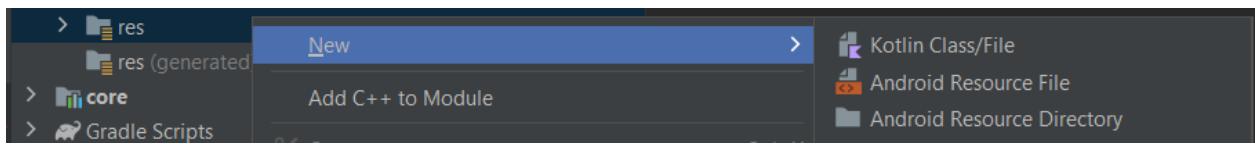
The screenshot shows the Android Studio interface with the 'gradle.properties' file open in the center editor pane. The file contains configuration settings for the project, including build options and API keys. The code is as follows:

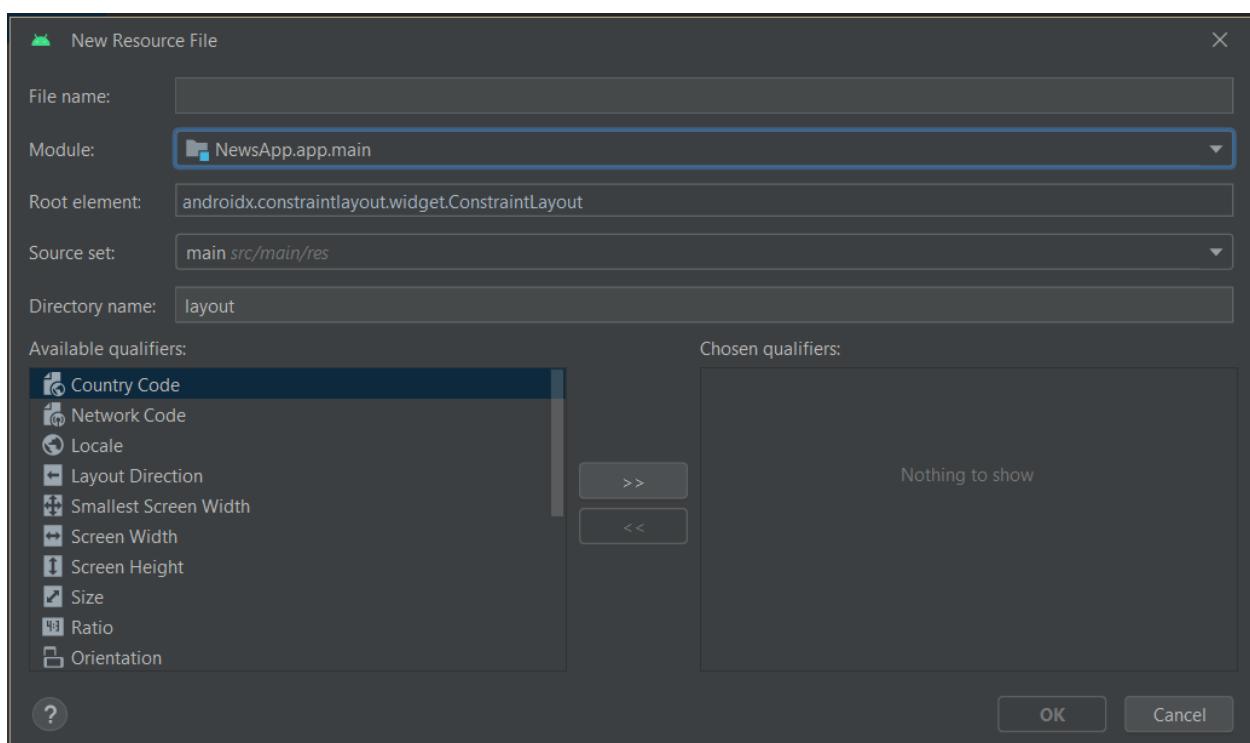
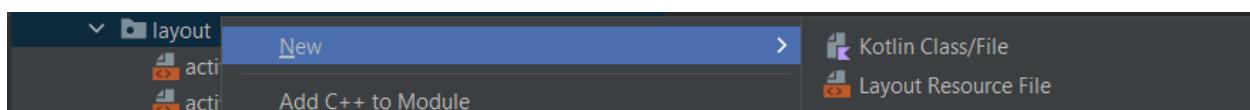
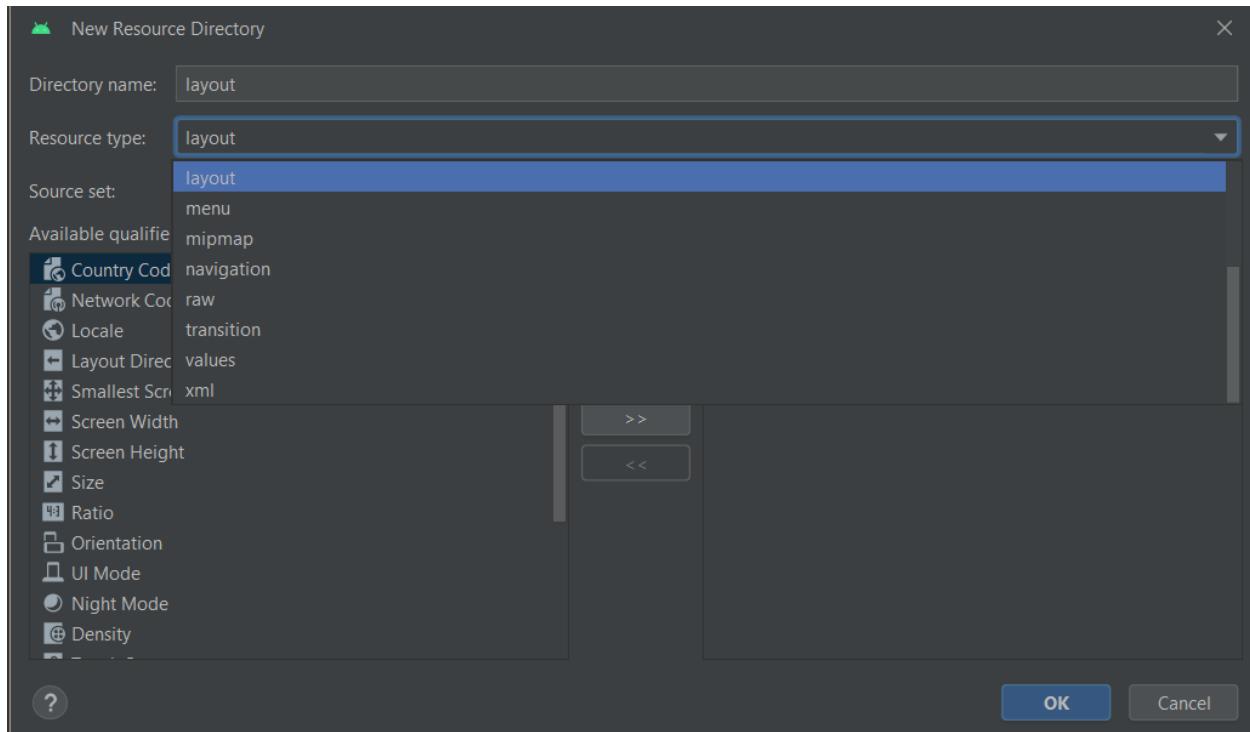
```
1 # Project-wide Gradle settings.
2 # IDE (e.g. Android Studio) users:
3 # Gradle settings configured through the IDE *will override*
4 # any settings specified in this file.
5 # For more details on how to configure your build environment visit
6 # http://www.gradle.org/docs/current/userguide/build_environment.html
7 # Specifies the JVM arguments used for the daemon process.
8 # The setting is particularly useful for tweaking memory settings.
9 org.gradle.jvmargs=-Xmx2048m -Dfile.encoding=UTF-8
10 # When configured, Gradle will run in incubating parallel mode.
11 # This option should only be used with decoupled projects. More details, visit
12 # http://www.gradle.org/docs/current/userguide/multi_project_builds.html#sec:decoupled_projects
13 # org.gradle.parallel=true
14 # AndroidX package structure to make it clearer which packages are bundled with the
15 # Android operating system, and which are packaged with your app's APK
16 # https://developer.android.com/topic/libraries/support-library/androidx-rn
17 android.useAndroidX=true
18 # Kotlin code style for this project: "official" or "obsolete":
19 kotlin.code.style=official
20 # Enables namespacing of each library's R class so that its R class includes only the
21 # resources declared in the library itself and none from the library's dependencies,
22 # thereby reducing the size of the R class for that library
23 android.nonTransitiveRClass=true
24 TOKEN="2ac4d73f6e364da4bd7c3ab5f927997e"
25 BASE_URL="https://newsapi.org/v2/"
```



TASK 7: Construct the user interface of the application and establish connections with the database

1. Use xml files to construct user interfaces for each screen, and they must be stored in an Android Resource Directory.
2. Right click on the resource folder, click on New, and click on Android Resource Directory.
3. Change the directory name to layout and change resource type to layout.
4. There will be an xml file for (). To create xml file, right click on resource folder, click on New, and click on the Layout resource file. Change the file name, depending on what screen the xml file will represent.
5. Use design and code to create a UI.



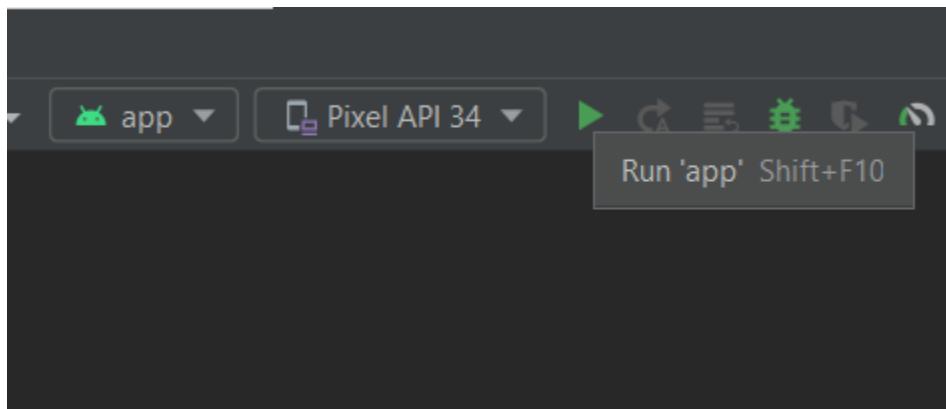


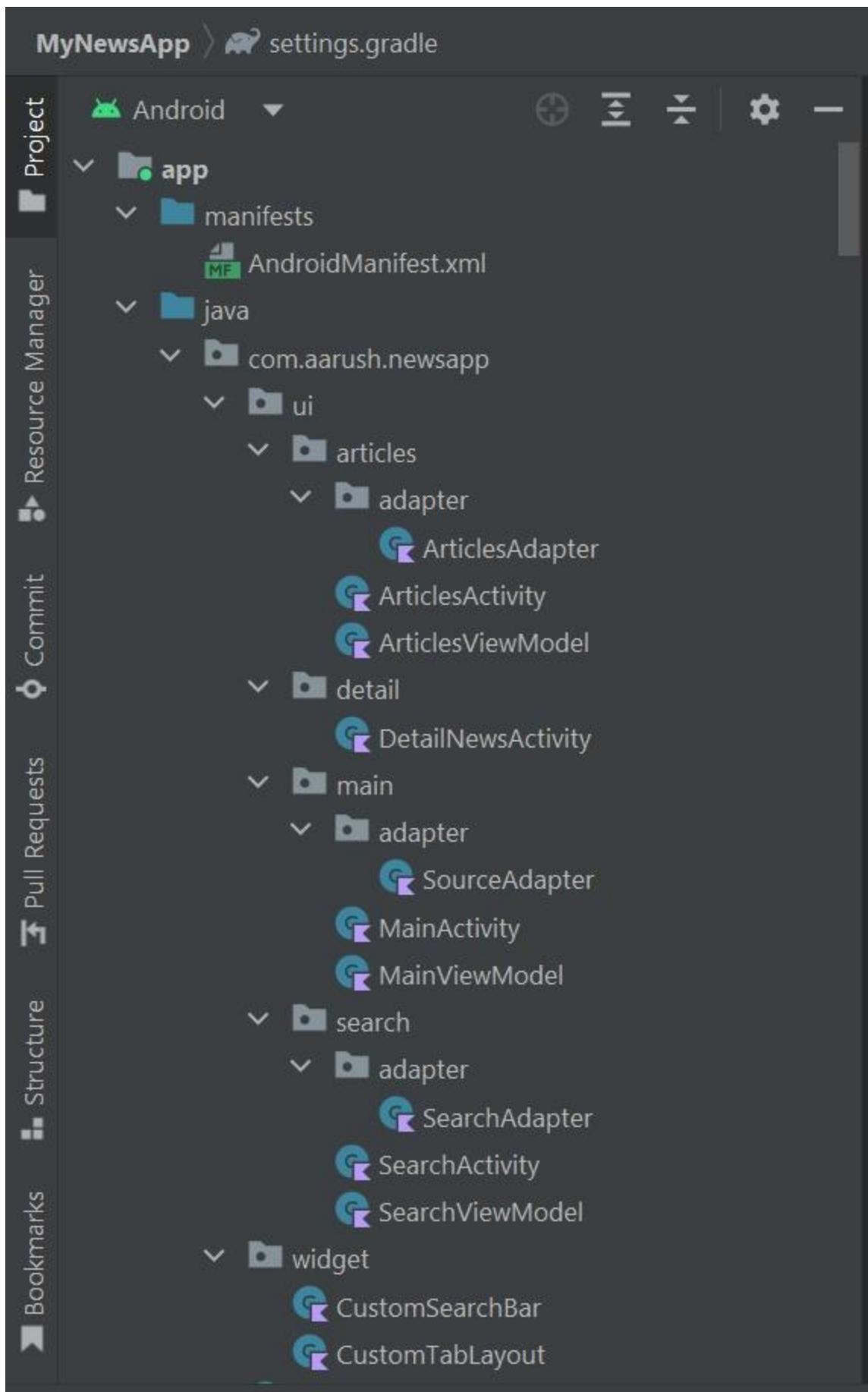
TASK 8: Construct the logic for the application and establish connections with the database

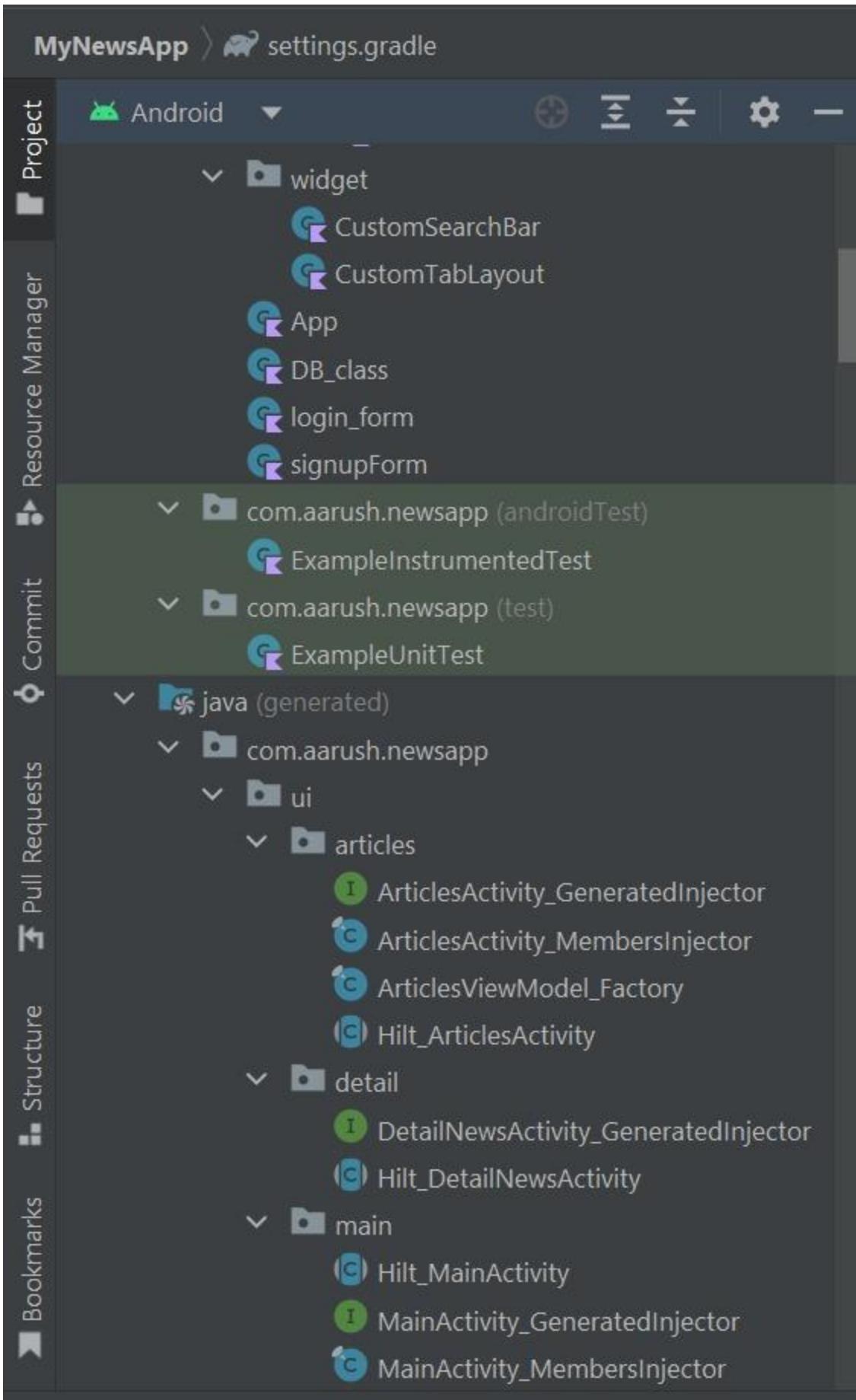
1. Use Kotlin files to construct logic for each screen.
2. Right click on the desired location (folder), click on New and click on Kotlin file.
3. Code the logic that would be used for each screen and concept of the application.

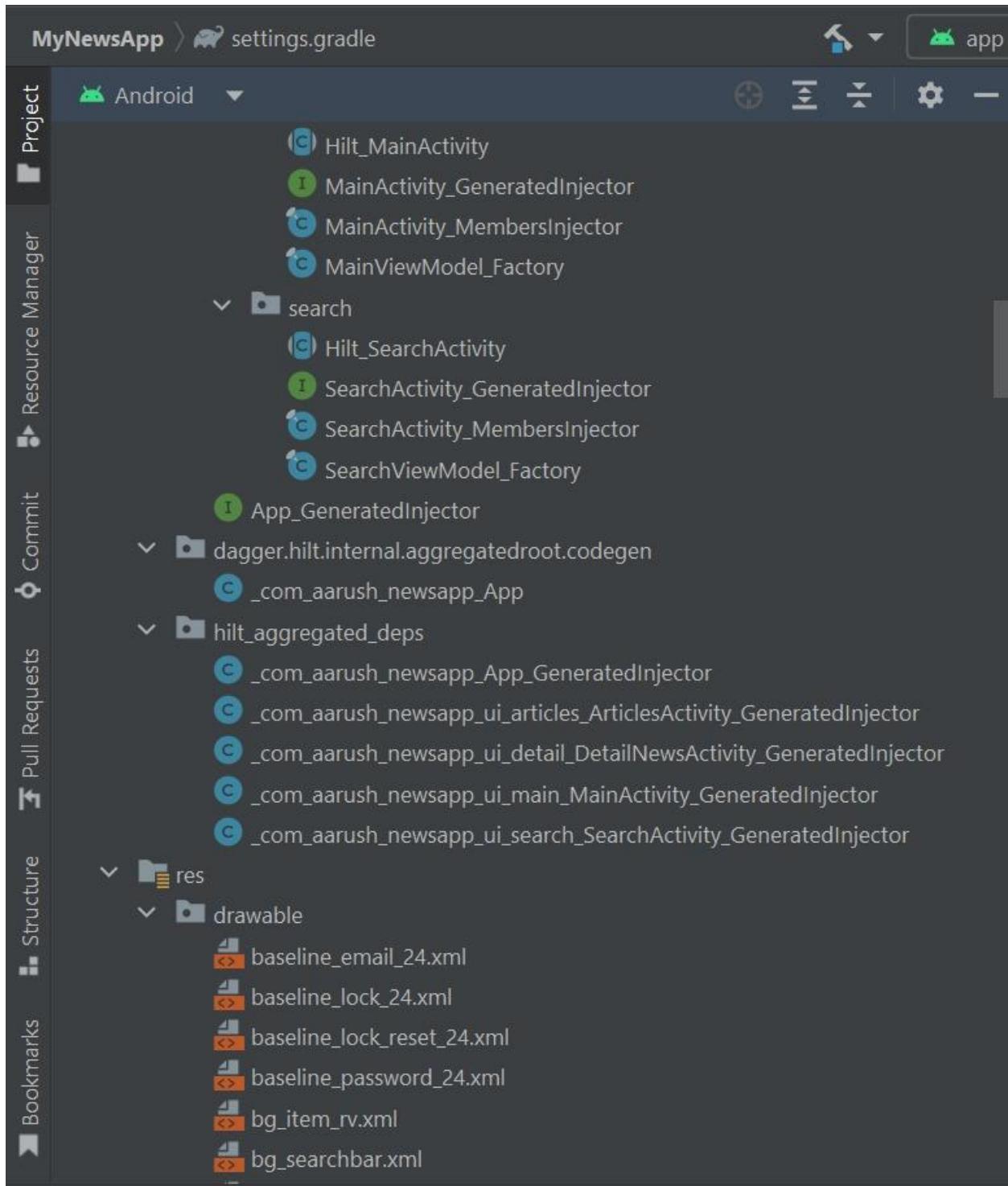
TASK 9: Execute the application to test it

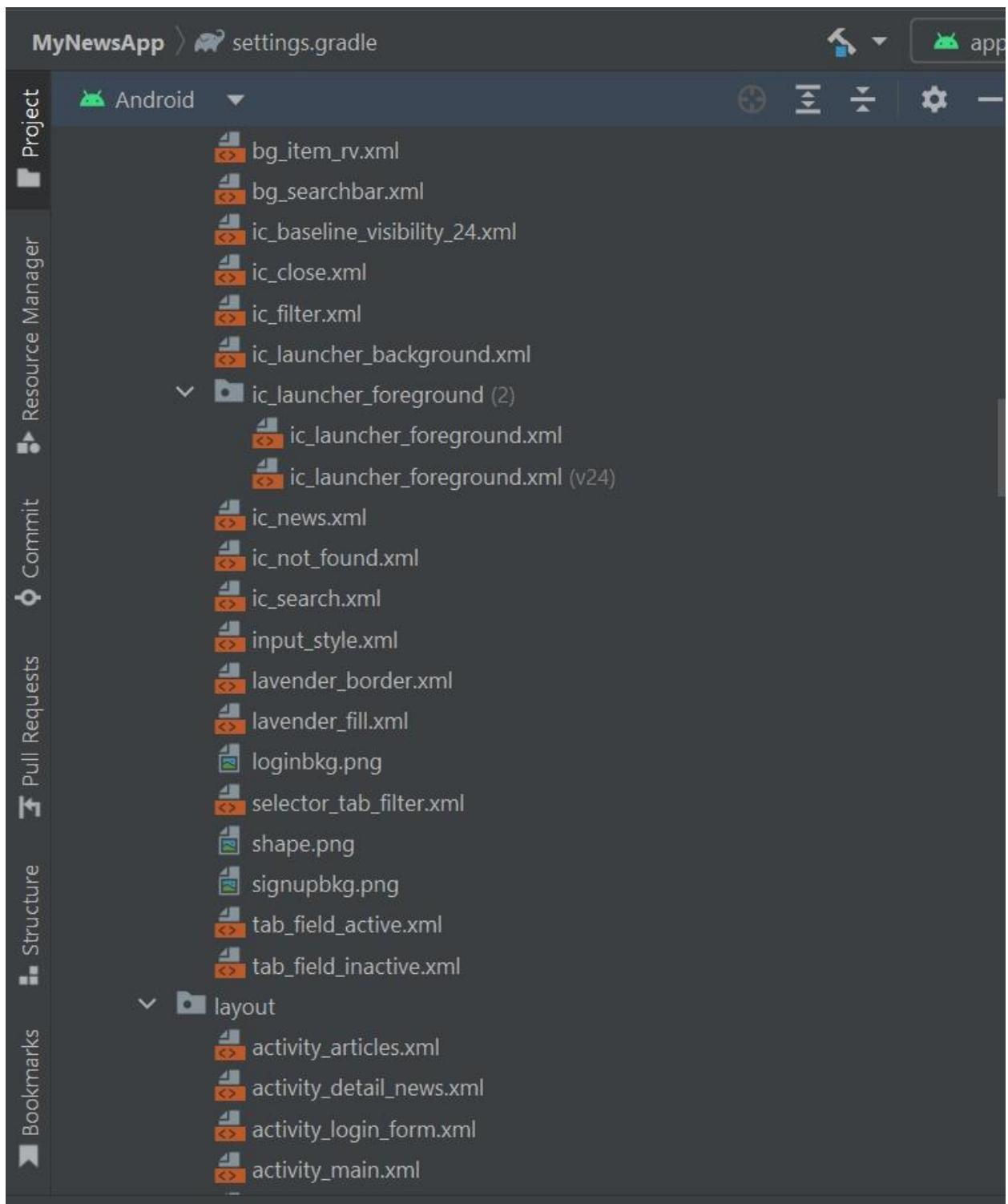
1. Create or find the right emulator.
2. Build and run the application.

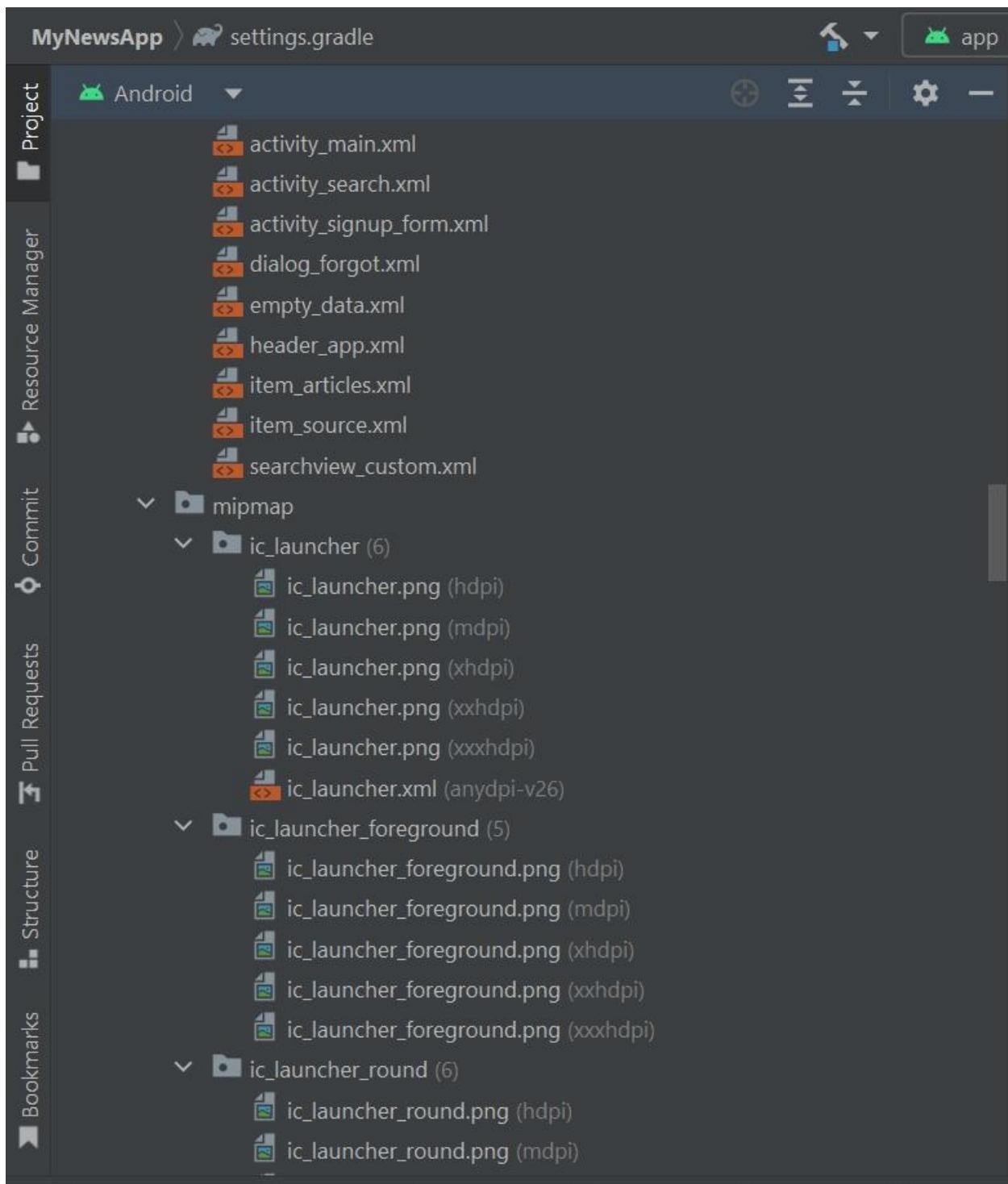


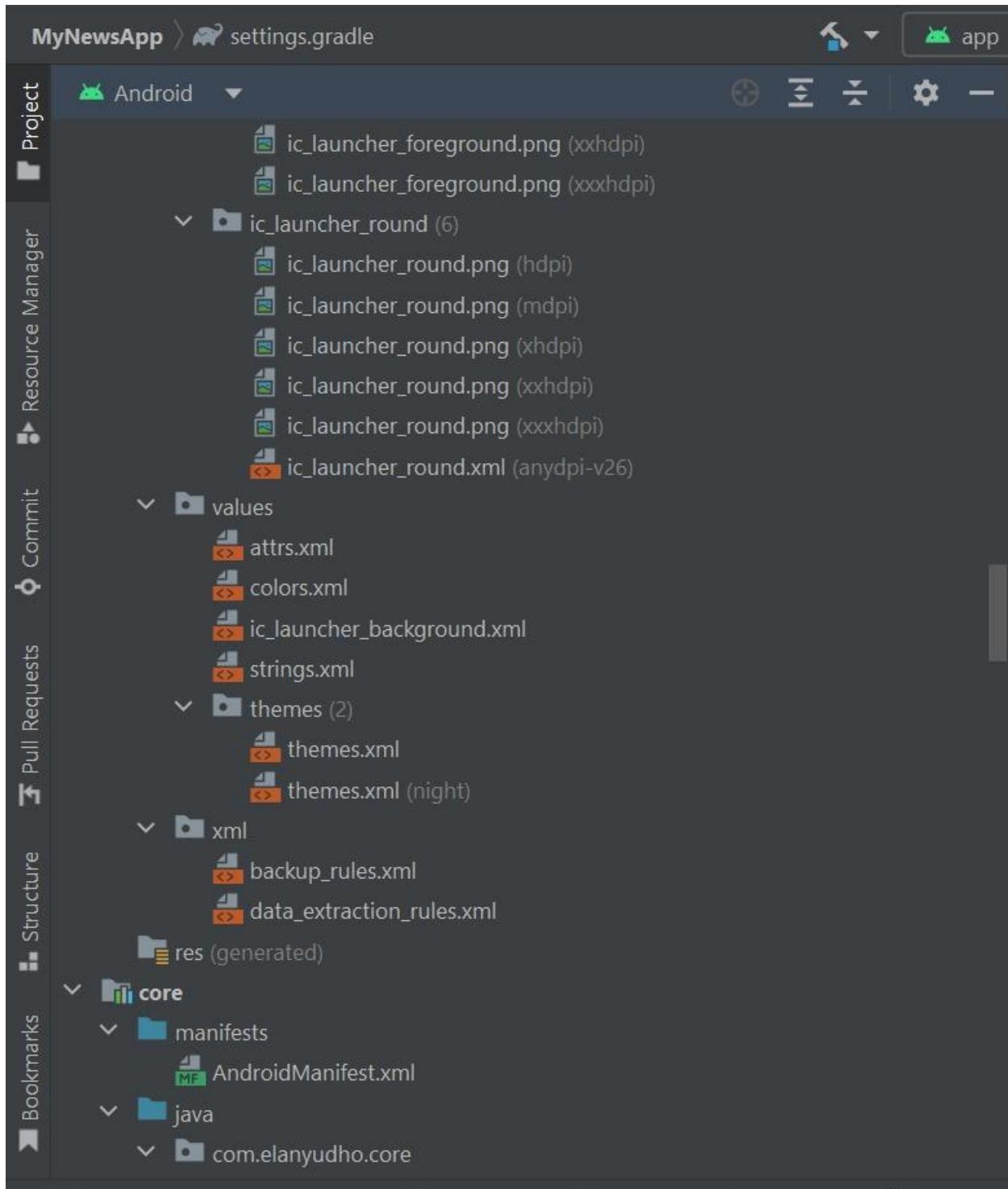


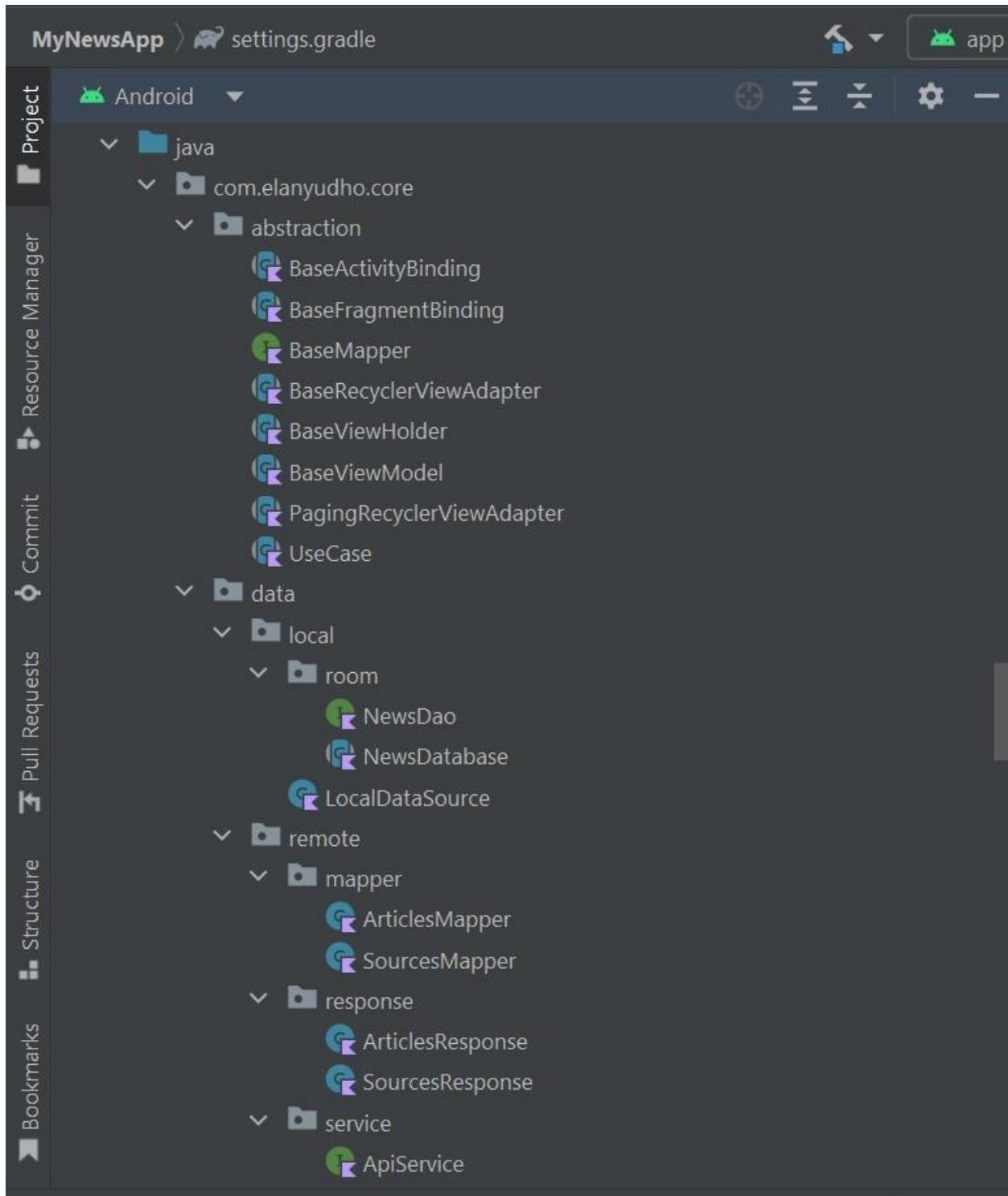


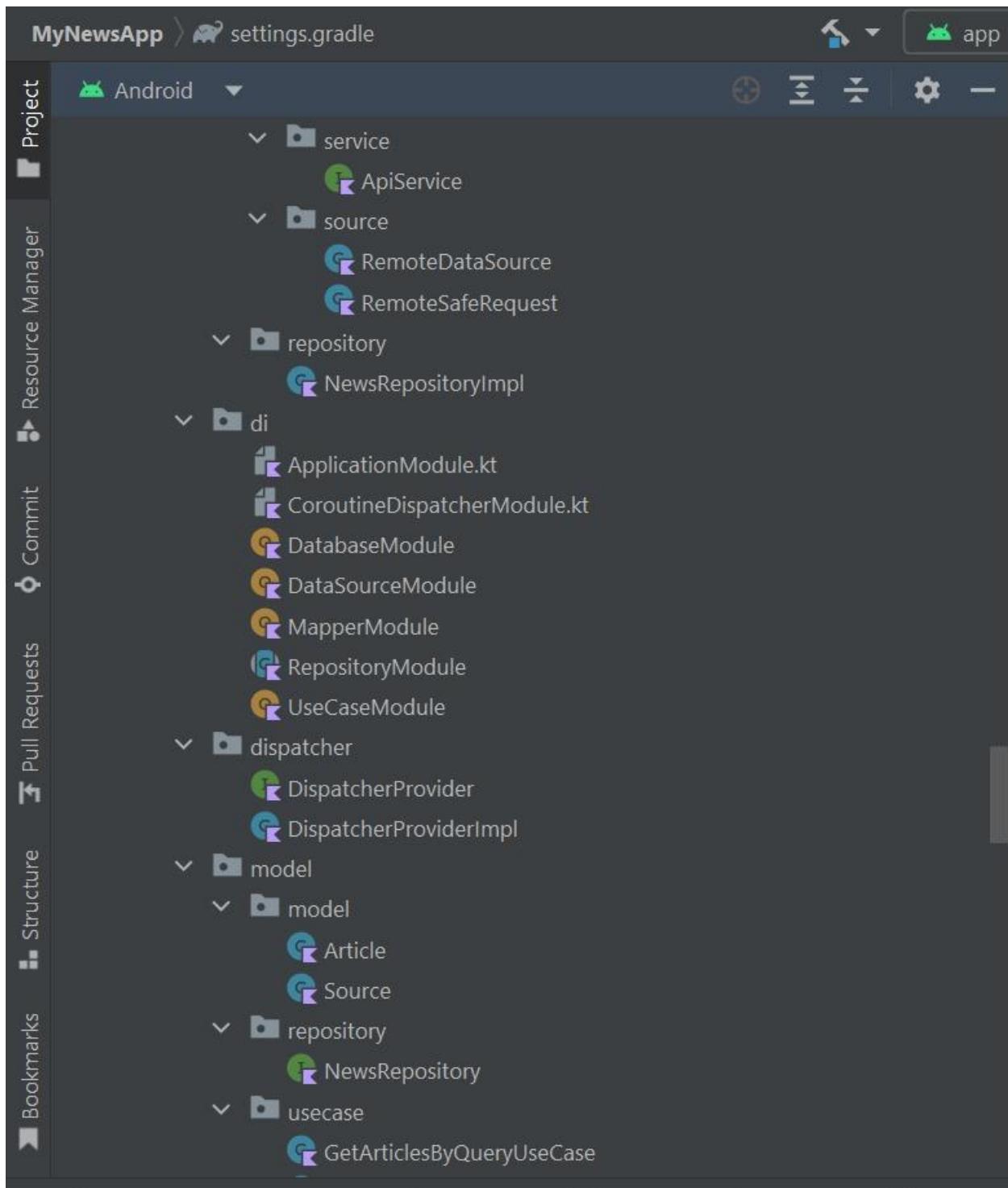


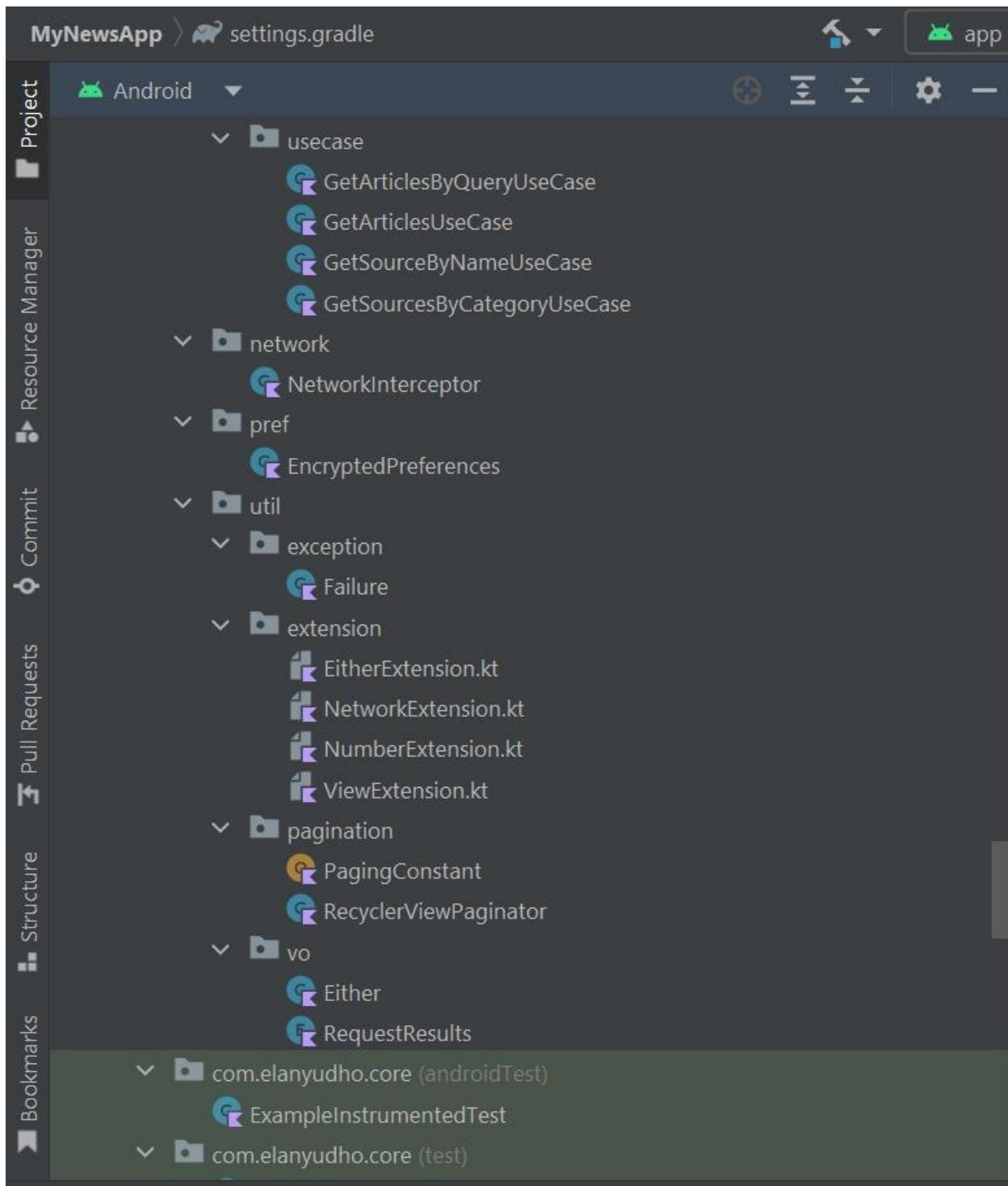


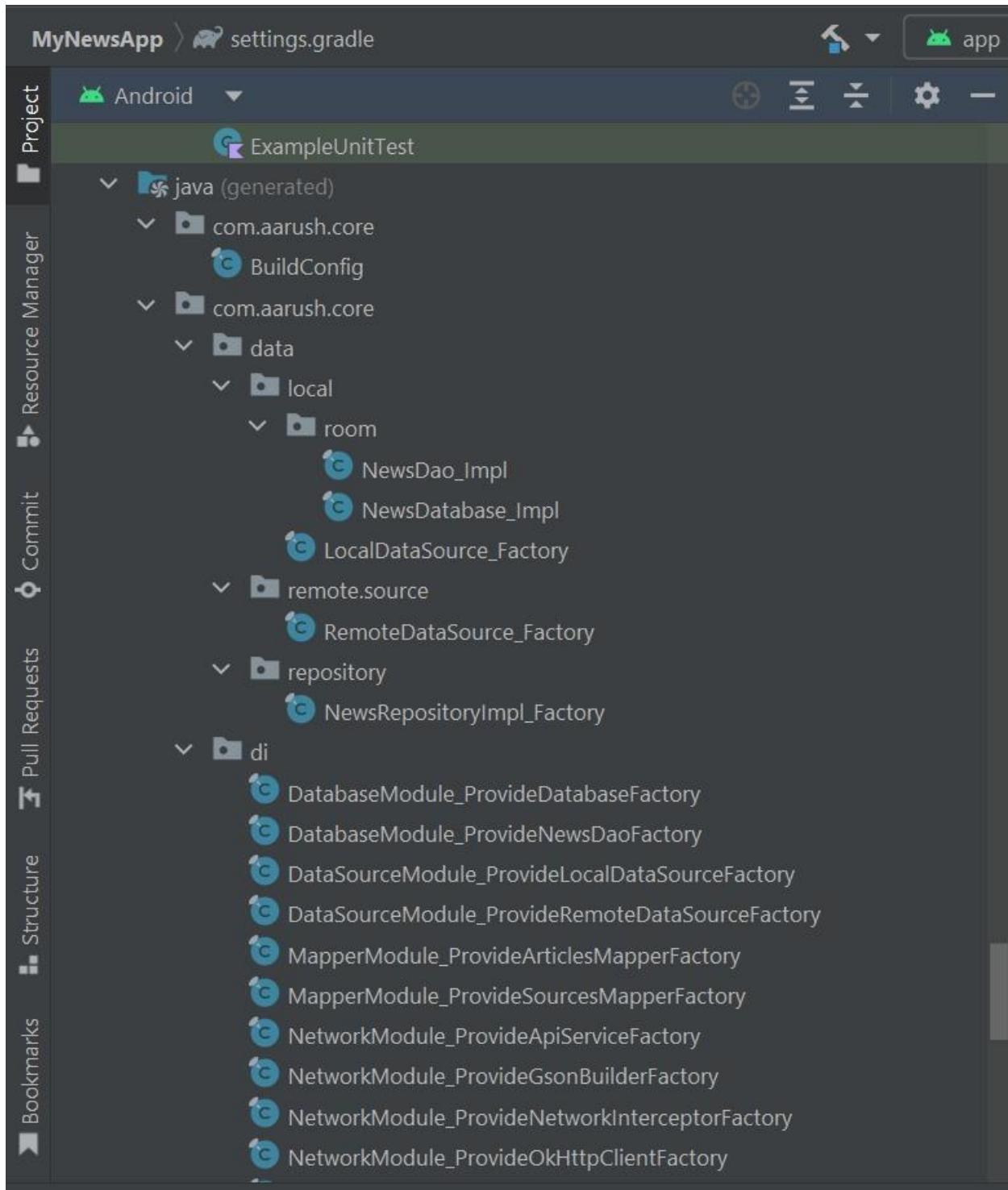


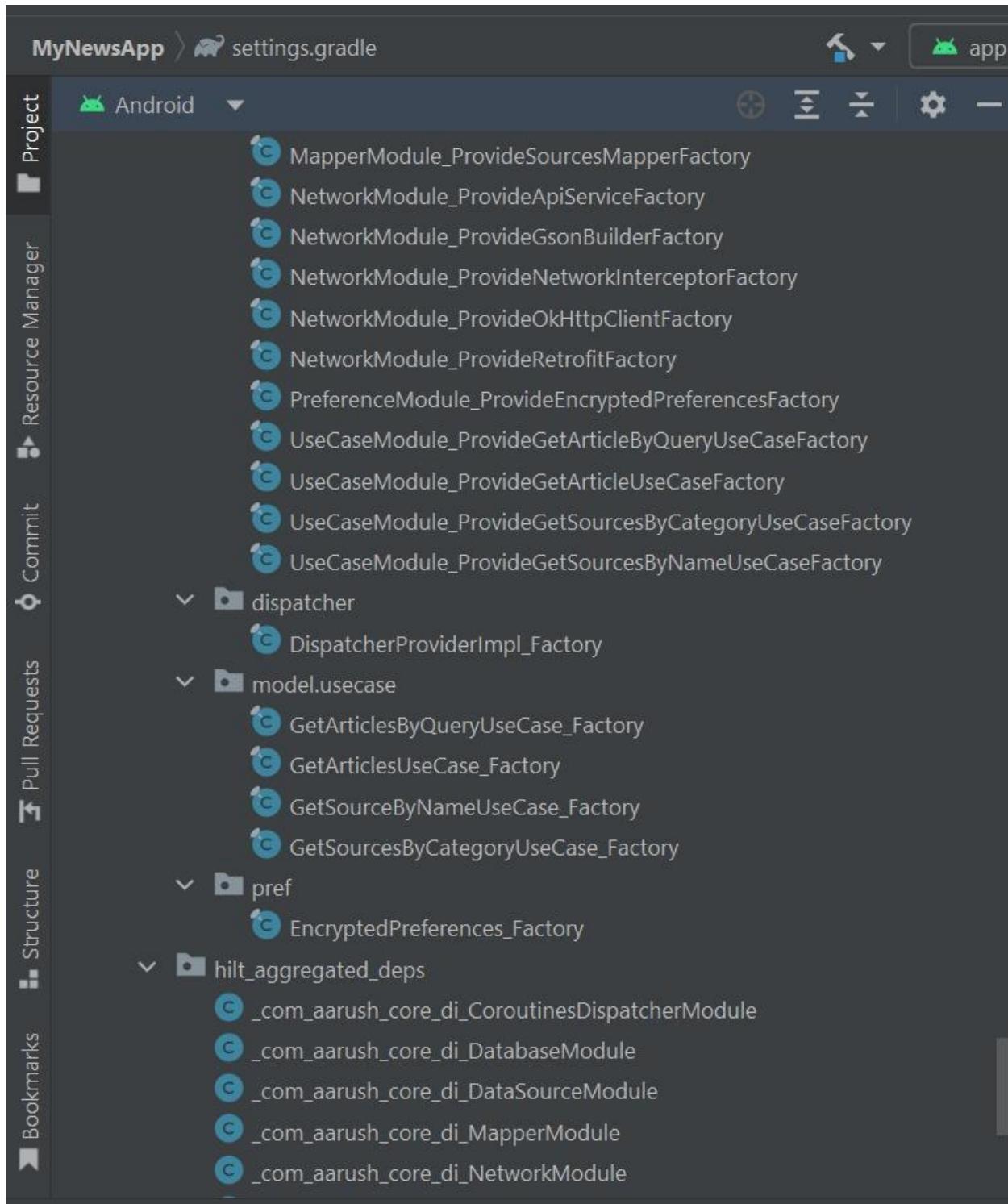


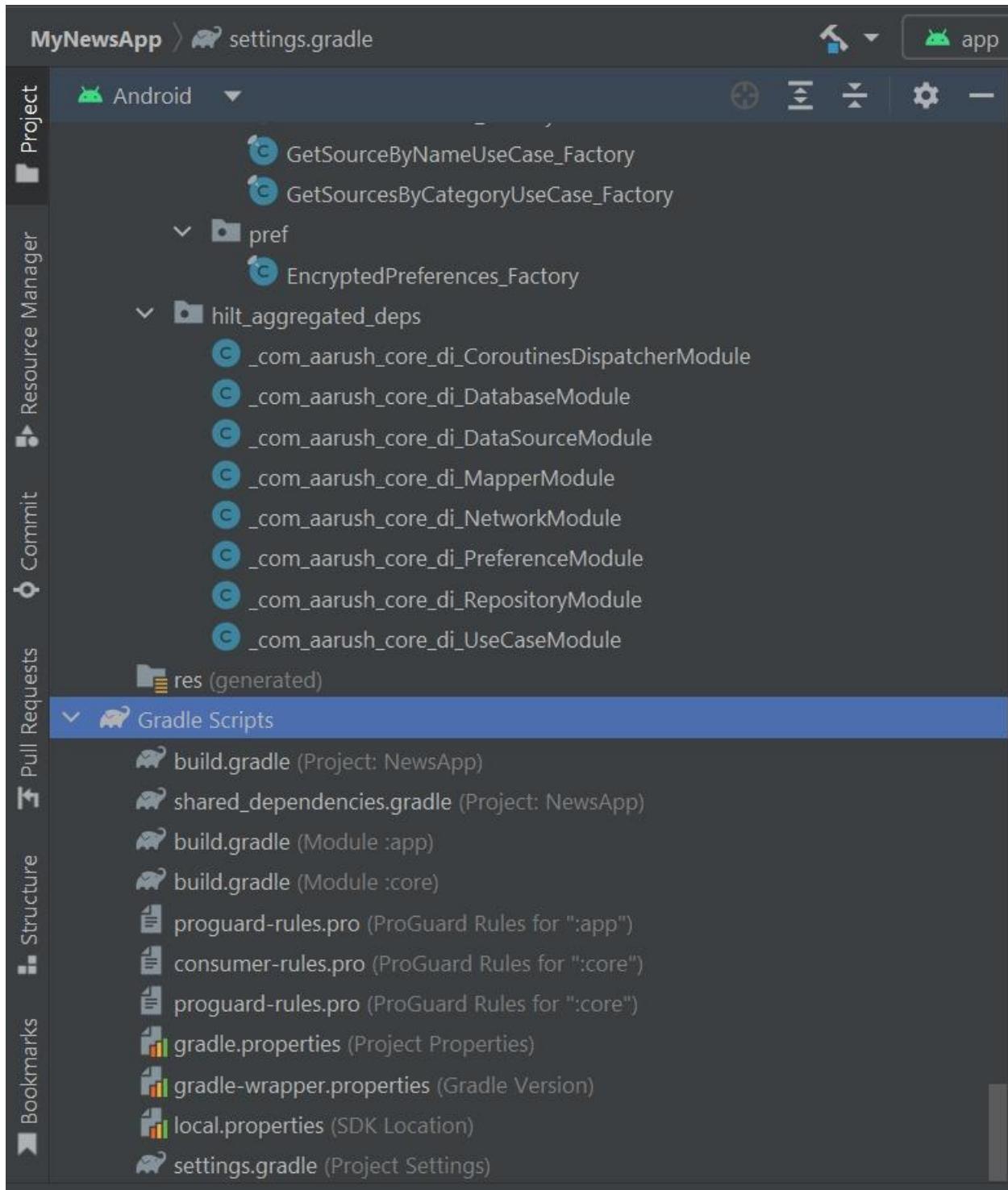








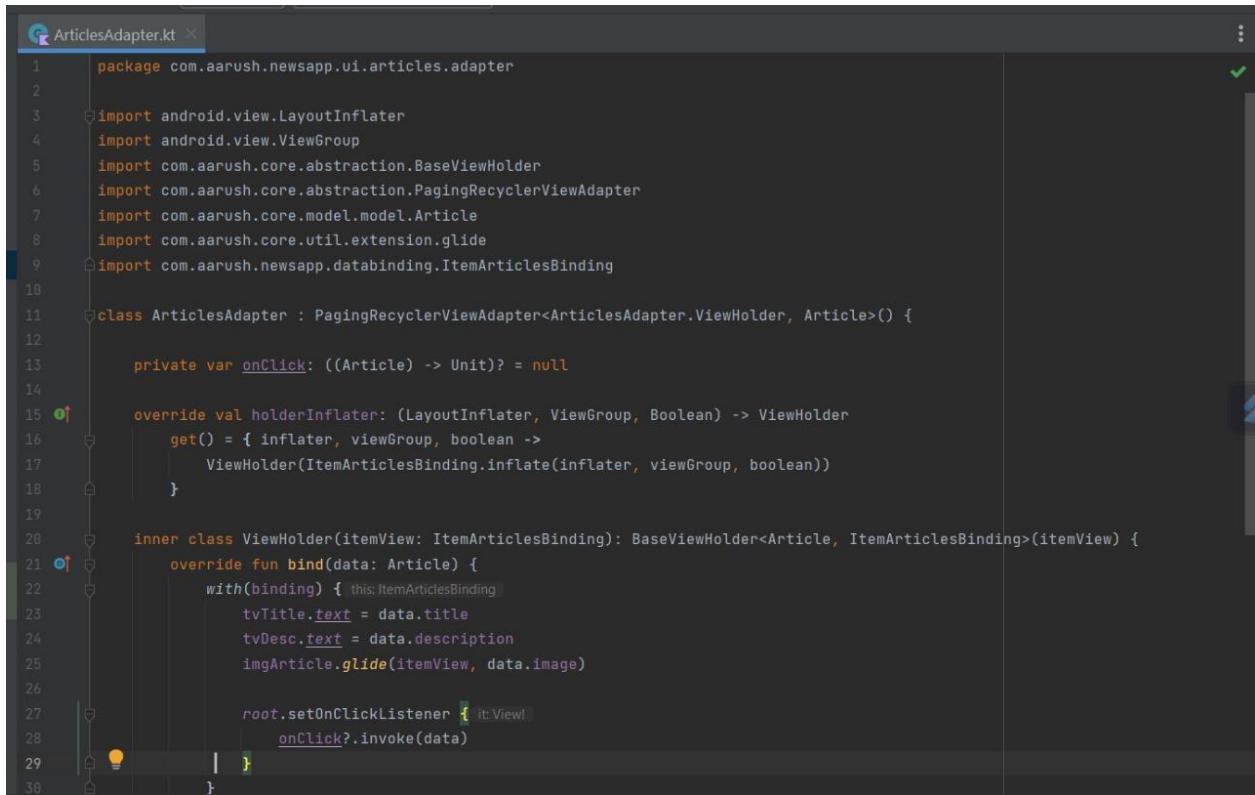




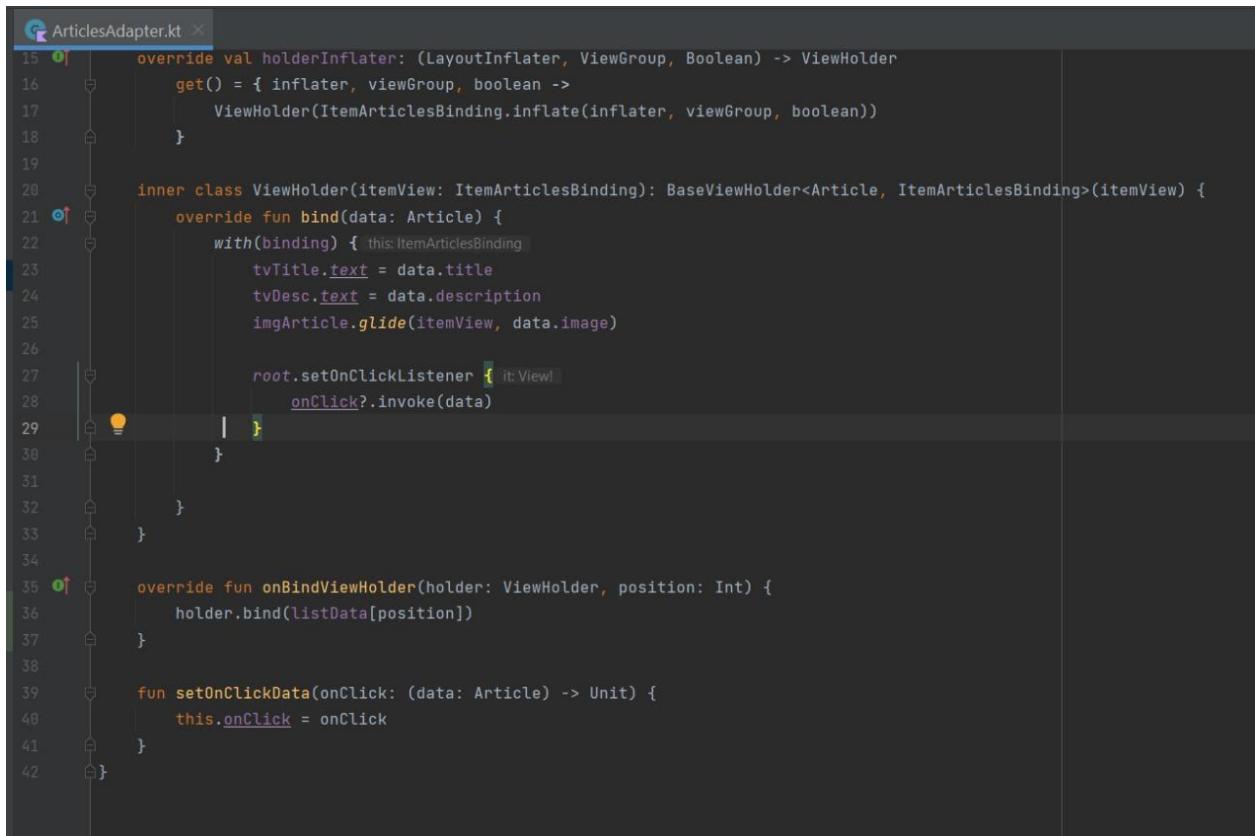
CODES:

1. ArticlesAdapter.kt
2. ArticlesActivity.kt
3. ArticlesViewModel.kt
4. DetailNewsActivity.kt
5. SourceAdapter.kt
6. MainActivity.kt
7. MainViewModel.kt
8. SearchAdapter.kt
9. SearchActivity.kt
10. SearchViewModel.kt
11. App.kt
12. DB_class.kt
13. login_form.kt
14. signupForm.kt
15. CustomSearchBar.kt
16. CustomTabLayout.kt
17. ExampleUnitTest.kt
18. activity_articles.xml
19. activity_detail_news.xml
20. activity_login_form.xml
21. activity_main.xml
22. activity_search.xml
23. activity_signup_form.xml
24. empty_data.xml
25. header_app.xml
26. item.xml
27. searchview_custom.xml

CODE 1: ArticlesAdapter.kt



```
1 package com.aarush.newsapp.ui.articles.adapter
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import com.aarush.core.abstraction.BaseViewHolder
6 import com.aarush.core.abstraction.PagingRecyclerViewAdapter
7 import com.aarush.core.model.model.Article
8 import com.aarush.core.util.extension.glide
9 import com.aarush.newsapp.databinding.ItemArticlesBinding
10
11 class ArticlesAdapter : PagingRecyclerViewAdapter<ArticlesAdapter.ViewHolder, Article>() {
12
13     private var onClick: ((Article) -> Unit)? = null
14
15     override val holderInflater: (LayoutInflater, ViewGroup, Boolean) -> ViewHolder
16         get() = { inflater, viewGroup, boolean -
17             ViewHolder(ItemArticlesBinding.inflate(inflater, viewGroup, boolean))
18         }
19
20     inner class ViewHolder(itemView: ItemArticlesBinding): BaseViewHolder<Article, ItemArticlesBinding>(itemView) {
21         override fun bind(data: Article) {
22             with(binding) {
23                 tvTitle.text = data.title
24                 tvDesc.text = data.description
25                 imgArticle.glide(itemView, data.image)
26
27                 root.setOnClickListener { it: View!
28                     onClick?.invoke(data)
29                 }
30             }
31         }
32     }
33 }
```



The screenshot shows the code editor of an Android application in the Android Studio IDE. The file being edited is `ArticlesAdapter.kt`. The code is written in Kotlin and defines a custom adapter for displaying articles. It includes methods for inflating view holders, binding data to views, and setting up click listeners.

```
15     override val holderInflater: (LayoutInflater, ViewGroup, Boolean) -> ViewHolder
16         get() = { inflater, viewGroup, boolean -
17             ViewHolder(ItemArticlesBinding.inflate(inflater, viewGroup, boolean))
18         }
19
20     inner class ViewHolder(itemView: ItemArticlesBinding): BaseViewHolder<Article, ItemArticlesBinding>(itemView) {
21         override fun bind(data: Article) {
22             with(binding) { this:ItemArticlesBinding
23                 tvTitle.text = data.title
24                 tvDesc.text = data.description
25                 imgArticle.glide(itemView, data.image)
26
27                 root.setOnClickListener { it: View!
28                     onClick?.invoke(data)
29                 }
30             }
31         }
32     }
33
34
35     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
36         holder.bind(listData[position])
37     }
38
39     fun setOnClickData(onClick: (data: Article) -> Unit) {
40         this.onClick = onClick
41     }
42 }
```

CODE 2: *ArticlesActivity.kt*

```
ArticlesActivity.kt
```

```
1 package com.aarush.newsapp.ui.articles
2
3 import android.content.Intent
4 import android.view.LayoutInflater
5 import android.view.View
6 import android.widget.Toast
7 import androidx.lifecycle.Observer
8 import androidx.recyclerview.widget.LinearLayoutManager
9 import com.aarush.core.abstraction BaseActivityBinding
10 import com.aarush.core.util.exception.Failure
11 import com.aarush.core.util.pagination.RecyclerViewPaginator
12 import com.aarush.newsapp.R
13 import com.aarush.newsapp.databinding.ActivityArticlesBinding
14 import com.aarush.newsapp.ui.articles.adapter.ArticlesAdapter
15 import com.aarush.newsapp.ui.detail.DetailNewsActivity
16 import dagger.hilt.android.AndroidEntryPoint
17 import javax.inject.Inject
18
19 @AndroidEntryPoint
20 class ArticlesActivity : BaseActivityBinding<ActivityArticlesBinding>(), Observer<ArticlesViewModel.ArticlesUiState> {
21
22     @Inject
23     lateinit var articlesViewModel: ArticlesViewModel
24
25     private val articlesAdapter: ArticlesAdapter by lazy { ArticlesAdapter() }
26
27     private var paginator: RecyclerViewPaginator? = null
28
29     private lateinit var sourceId: String
30 }
```

```
ArticlesActivity.kt
```

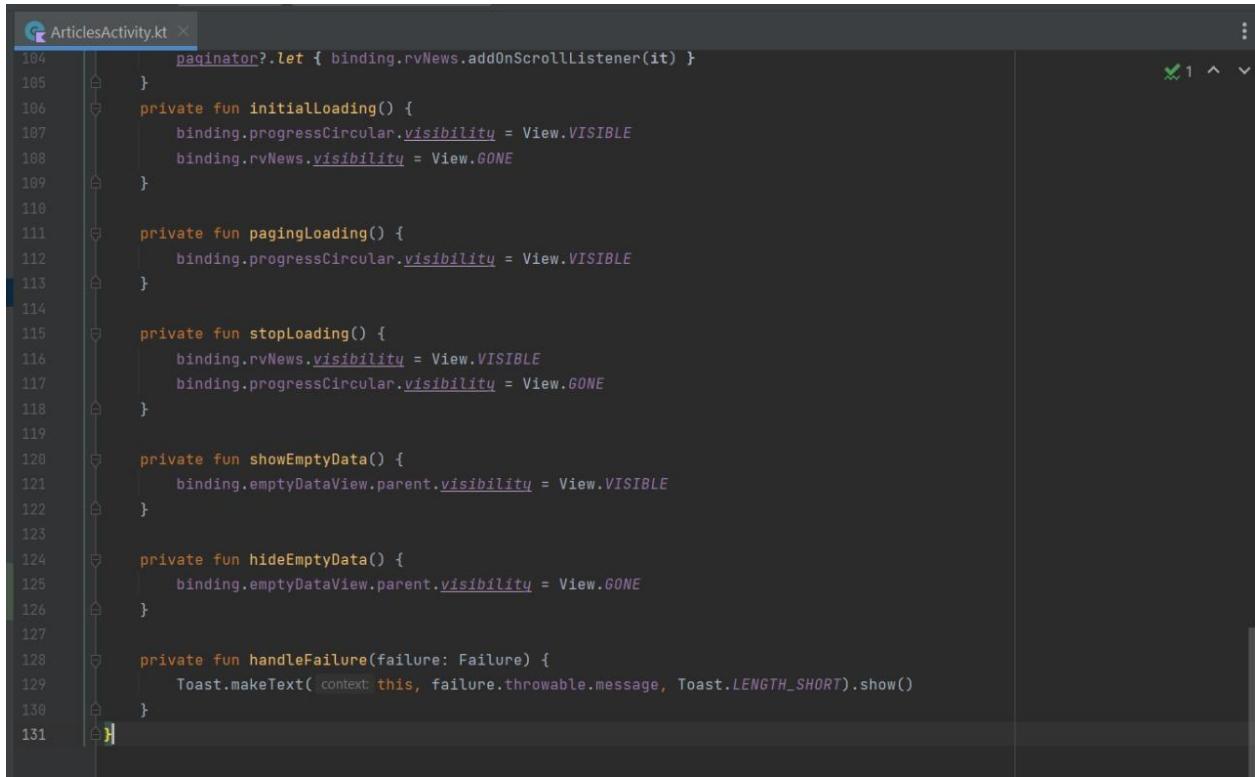
```
28
29     private lateinit var sourceId: String
30
31     override val bindingInflater: (LayoutInflater) -> ActivityArticlesBinding
32         get() = { ActivityArticlesBinding.inflate(layoutInflater) }
33
34     override fun setupView() {
35         getDataIntent()
36         initViewModel()
37         setHeader()
38         setAdapter()
39         setPagination()
40     }
41
42     override fun onChanged(state: ArticlesViewModel.ArticlesUiState?) {
43         when(state) {
44             is ArticlesViewModel.ArticlesUiState.ArticlesLoaded -> {
45                 stopLoading()
46                 if (state.data.isEmpty() && paginator?.isFirstGet == true) {
47                     showEmptyData()
48                 } else {
49                     hideEmptyData()
50                     articlesAdapter.appendList(state.data)
51                 }
52             }
53             is ArticlesViewModel.ArticlesUiState.InitialLoading -> {
54                 initialLoading()
55             }
56             is ArticlesViewModel.ArticlesUiState.PagingLoading -> {
57             }
58         }
59     }
60 }
```

```
ArticlesActivity.kt
```

```
55     initialLoading()
56 }
57     is ArticlesViewModel.ArticlesUiState.PagingLoading -> {
58         pagingLoading()
59     }
60     is ArticlesViewModel.ArticlesUiState.FailedLoaded -> {
61         stopLoading()
62         handleFailure(state.failure)
63     }
64     else -> {}
65 }
66 }
67
68 private fun getDataIntent() {
69     sourceId = intent.getStringExtra(name: "sourceId") ?: ""
70 }
71
72 private fun initViewModel() {
73     articlesViewModel.viewModel.observe(owner: this, observer: this)
74     articlesViewModel.getArticles(sourceId, page: 1)
75 }
76
77 private fun setHeader() {
78     with(binding) { this: ActivityArticlesBinding
79         headerDetail.tvHeader.text = "Articles"
80         headerDetail.btnBack.setOnClickListener { onBackPressedDispatcher.onBackPressed() }
81     }
82 }
83
84 private fun setAdapter() {
```

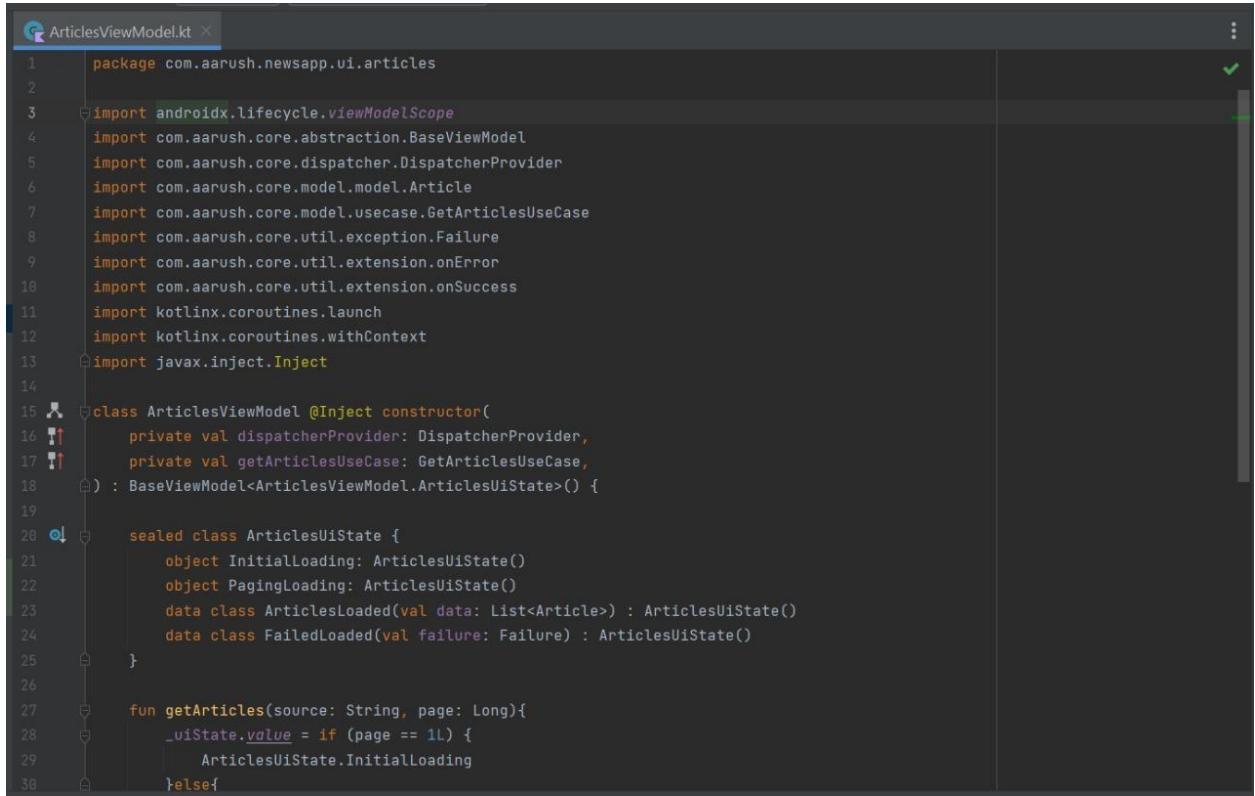
```
ArticlesActivity.kt
```

```
82     }
83
84     private fun setAdapter() {
85         with(binding.rvNews) { this: RecyclerView
86             adapter = articlesAdapter
87             setHasFixedSize(true)
88
89             articlesAdapter.setOnItemClickListener { it: Article
90                 val intent = Intent(packageContext: this@ArticlesActivity, DetailNewsActivity::class.java)
91                 intent.putExtra(name: "sourceNews", it.source)
92                 intent.putExtra(name: "urlNews", it.url)
93                 startActivity(intent)
94             }
95         }
96     }
97
98     private fun setPagination() {
99         paginator = RecyclerViewPaginator(binding.rvNews.layoutManager as LinearLayoutManager)
100        paginator?.setOnLoadMoreListener { page ->
101            paginator?.isFirstGet = false
102            articlesViewModel.getArticles(sourceId, page)
103        }
104        paginator?.let { binding.rvNews.addOnScrollListener(it) }
105    }
106
107    private fun initialLoading() {
108        binding.progressCircular.visibility = View.VISIBLE
109        binding.rvNews.visibility = View.GONE
110    }
111
112    private fun pagingLoading() {
```



```
184     paginator?.let { binding.rvNews.addOnScrollListener(it) }
185   }
186 
187   private fun initialLoading() {
188     binding.progressCircular.visibility = View.VISIBLE
189     binding.rvNews.visibility = View.GONE
190   }
191 
192   private fun pagingLoading() {
193     binding.progressCircular.visibility = View.VISIBLE
194   }
195 
196   private fun stopLoading() {
197     binding.rvNews.visibility = View.VISIBLE
198     binding.progressCircular.visibility = View.GONE
199   }
200 
201   private fun showEmptyData() {
202     binding.emptyDataView.parent.visibility = View.VISIBLE
203   }
204 
205   private fun hideEmptyData() {
206     binding.emptyDataView.parent.visibility = View.GONE
207   }
208 
209   private fun handleFailure(failure: Failure) {
210     Toast.makeText(context, failure.throwable.message, Toast.LENGTH_SHORT).show()
211   }
212 }
```

CODE 3: ArticlesViewModel.kt



```
1 package com.aarush.newsapp.ui.articles
2 
3 import androidx.lifecycle.viewModelScope
4 import com.aarush.core.abstraction.BaseViewModel
5 import com.aarush.core.dispatcher.DispatcherProvider
6 import com.aarush.core.model.model.Article
7 import com.aarush.core.model.usecase.GetArticlesUseCase
8 import com.aarush.core.util.exception.Failure
9 import com.aarush.core.util.extension.onError
10 import com.aarush.core.util.extension.onSuccess
11 import kotlinx.coroutines.launch
12 import kotlinx.coroutines.withContext
13 import javax.inject.Inject
14 
15 class ArticlesViewModel @Inject constructor(
16   private val dispatcherProvider: DispatcherProvider,
17   private val getArticlesUseCase: GetArticlesUseCase,
18 ) : BaseViewModel<ArticlesViewModel.ArticlesUiState>() {
19 
20   sealed class ArticlesUiState {
21     object InitialLoading: ArticlesUiState()
22     object PagingLoading: ArticlesUiState()
23     data class ArticlesLoaded(val data: List<Article>) : ArticlesUiState()
24     data class FailedLoaded(val failure: Failure) : ArticlesUiState()
25   }
26 
27   fun getArticles(source: String, page: Long){
28     _uiState.value = if (page == 1L) {
29       ArticlesUiState.InitialLoading
30     } else {
```

```
object InitialLoading: ArticlesUiState()
object PagingLoading: ArticlesUiState()
data class ArticlesLoaded(val data: List<Article>) : ArticlesUiState()
data class FailedLoaded(val failure: Failure) : ArticlesUiState()

fun getArticles(source: String, page: Long){
    _uiState.value = if (page == 1L) {
        ArticlesUiState.InitialLoading
    }else{
        ArticlesUiState.PagingLoading
    }
    viewModelScope.launch(dispatcherProvider.io) { this: CoroutineScope
        getArticlesUseCase.run(GetArticlesUseCase.Params(source, page.toString()))
            .onSuccess { it: List<Article>
                withContext(dispatcherProvider.main) { this: CoroutineScope
                    _uiState.value = ArticlesUiState.ArticlesLoaded(it)
                }
            }
            .onError { it: Failure
                withContext(dispatcherProvider.main) { this: CoroutineScope
                    _uiState.value = ArticlesUiState.FailedLoaded(it)
                }
            }
    }
}
```

CODE 4: DetailNewsActivity.kt

```
package com.aarush.newsapp.ui.detail

import android.annotation.SuppressLint
import android.view.LayoutInflater
import android.view.View
import android.webkit.WebChromeClient
import android.webkit.WebSettings
import android.webkit.WebView
import androidx.lifecycle.lifecycleScope
import com.aarush.core.abstraction BaseActivityBinding
import com.aarush.newsapp.R
import com.aarush.newsapp.databinding.ActivityDetailNewsBinding
import dagger.hilt.android.AndroidEntryPoint
import kotlinx.coroutines.Dispatchers
import kotlinx.coroutines.launch
import java.lang.ref.WeakReference

@AndroidEntryPoint
class DetailNewsActivity : BaseActivityBinding<ActivityDetailNewsBinding>() {

    private lateinit var urlNews: String
    private lateinit var sourceNews: String

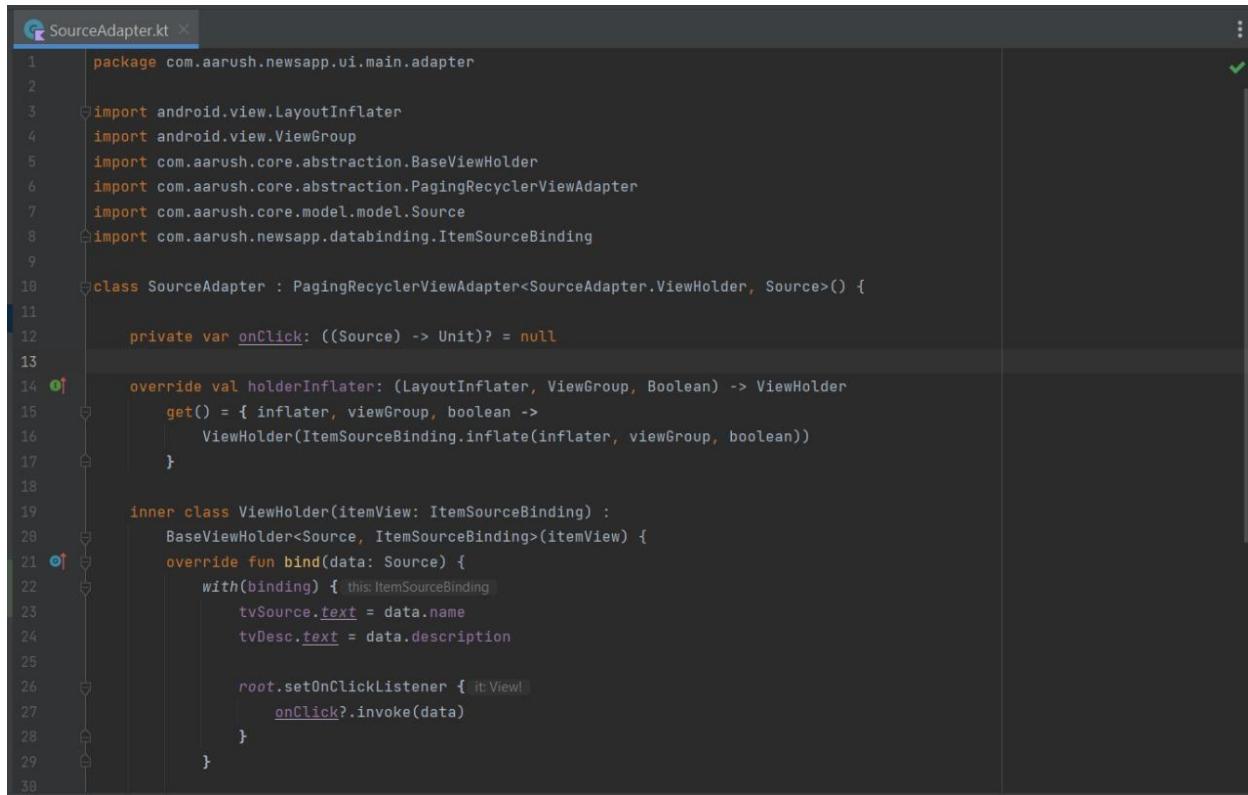
    override val bindingInflater: (LayoutInflater) -> ActivityDetailNewsBinding
        get() = { ActivityDetailNewsBinding.inflate(layoutInflater) }

    override fun setupView() {
        getDataIntent()
        setHeader()
        setWebView()
    }
}
```

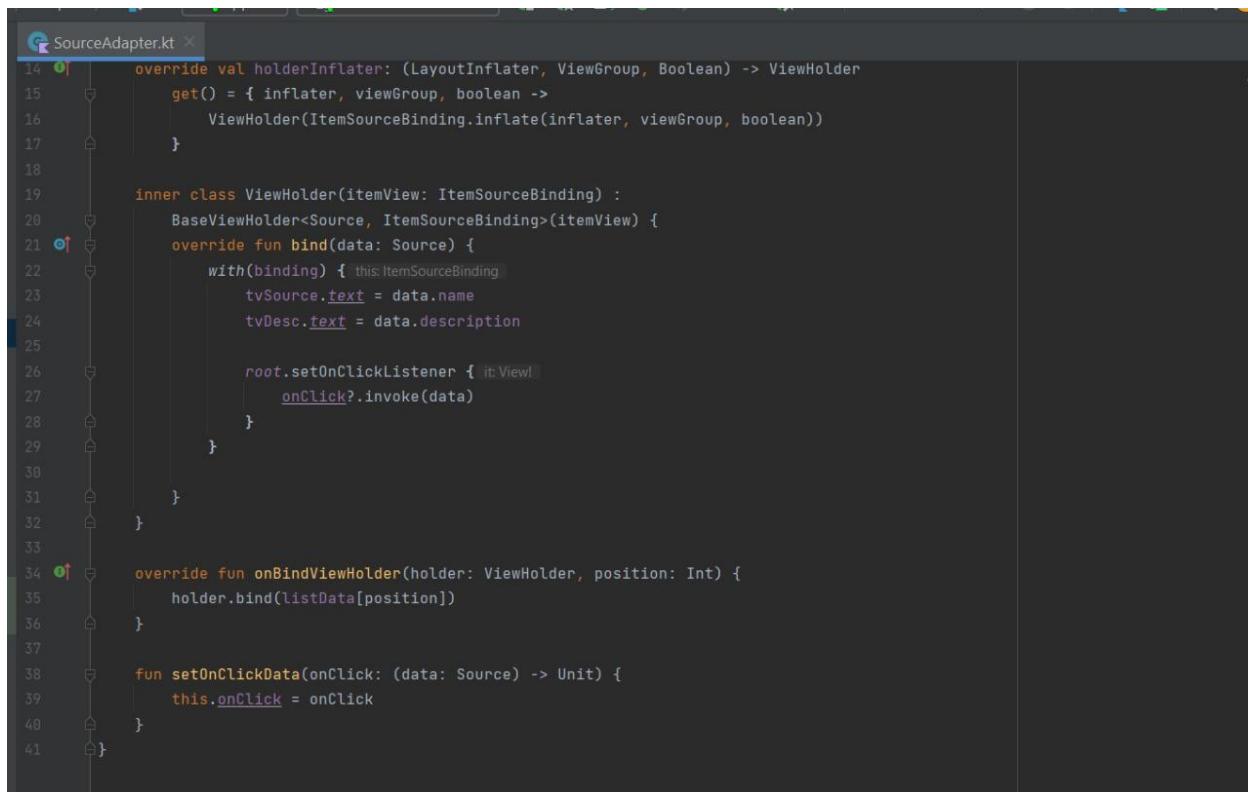
```
28     getDataIntent()
29     setHeader()
30     setWebView()
31 }
32
33 private fun getDataIntent() {
34     urlNews = intent.getStringExtra( name: "urlNews" ) ?: ""
35     sourceNews = intent.getStringExtra( name: "sourceNews" ) ?: ""
36 }
37
38 private fun setHeader() {
39     with(binding) { this: ActivityDetailNewsBinding
40         headerDetail.tvHeader.text = sourceNews
41         headerDetail.btnBack.setOnClickListener { onBackPressedDispatcher.onBackPressed() }
42     }
43 }
44
45 @SuppressLint("SetJavaScriptEnabled")
46 private fun setWebView() {
47     val webSettings: WebSettings = binding.webview.settings
48     webSettings.javaScriptEnabled = true
49     webSettings.mediaPlaybackRequiresUserGesture = false
50     binding.webview.webChromeClient = object : WebChromeClient() {
51         override fun onProgressChanged(view: WebView?, newProgress: Int) {
52             super.onProgressChanged(view, newProgress)
53             if (newProgress == 100) {
54                 if (!this@DetailNewsActivity.isDestroyed) {
55                     binding.progressCircular.visibility = View.GONE
56                 }
57             }
58         }
59     }
60     binding.webview.loadUrl(urlNews)
61 }
62
63 }
```

```
36 }
37
38 private fun setHeader() {
39     with(binding) { this: ActivityDetailNewsBinding
40         headerDetail.tvHeader.text = sourceNews
41         headerDetail.btnBack.setOnClickListener { onBackPressedDispatcher.onBackPressed() }
42     }
43 }
44
45 @SuppressLint("SetJavaScriptEnabled")
46 private fun setWebView() {
47     val webSettings: WebSettings = binding.webview.settings
48     webSettings.javaScriptEnabled = true
49     webSettings.mediaPlaybackRequiresUserGesture = false
50     binding.webview.webChromeClient = object : WebChromeClient() {
51         override fun onProgressChanged(view: WebView?, newProgress: Int) {
52             super.onProgressChanged(view, newProgress)
53             if (newProgress == 100) {
54                 if (!this@DetailNewsActivity.isDestroyed) {
55                     binding.progressCircular.visibility = View.GONE
56                 }
57             }
58         }
59     }
60     binding.webview.loadUrl(urlNews)
61 }
62
63 }
```

CODE 5: SourceAdapter.kt

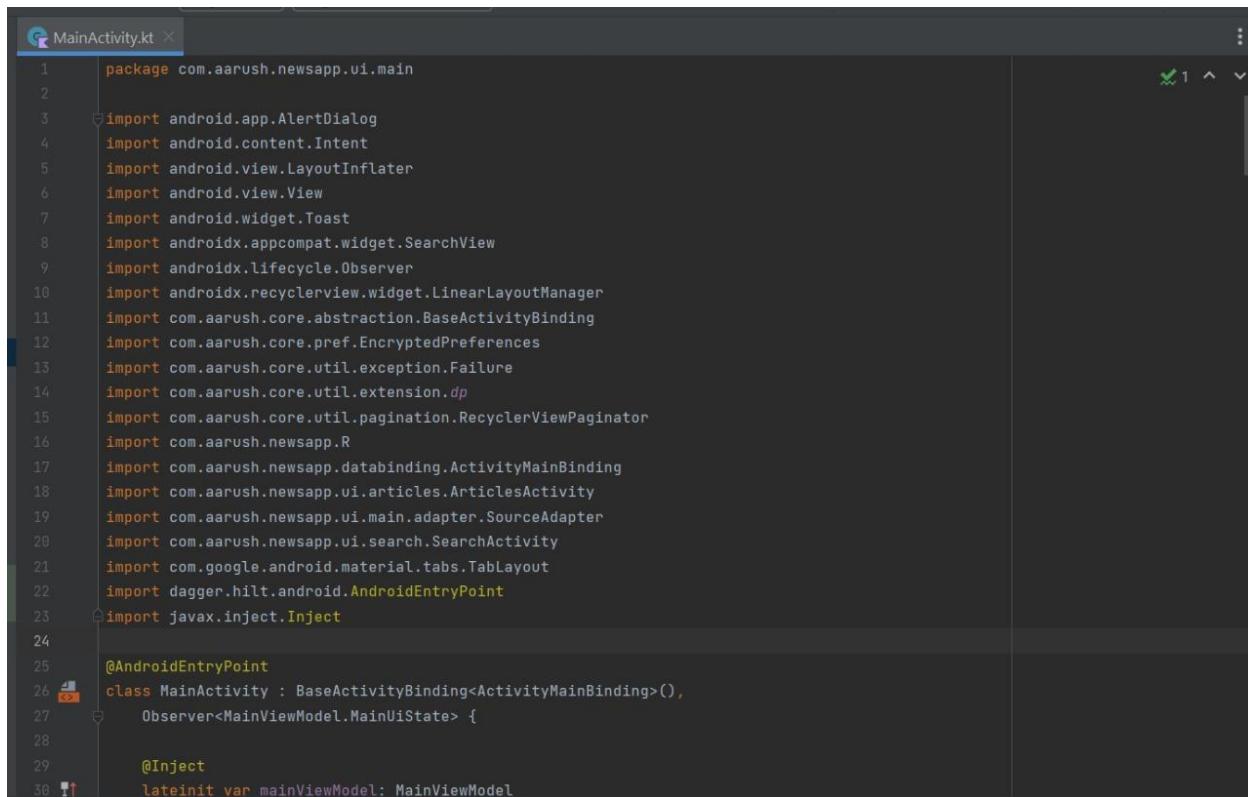


```
SourceAdapter.kt
1 package com.aarush.newsapp.ui.main.adapter
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import com.aarush.core.abstraction.BaseViewHolder
6 import com.aarush.core.abstraction.PagingRecyclerViewAdapter
7 import com.aarush.core.model.model.Source
8 import com.aarush.newsapp.databinding.ItemSourceBinding
9
10 class SourceAdapter : PagingRecyclerViewAdapter<SourceAdapter.ViewHolder, Source>() {
11
12     private var onClick: ((Source) -> Unit)? = null
13
14     override val holderInflater: (LayoutInflater, ViewGroup, Boolean) -> ViewHolder
15         get() = { inflator, viewGroup, boolean ->
16             ViewHolder(ItemSourceBinding.inflate(inflator, viewGroup, boolean))
17         }
18
19     inner class ViewHolder(itemView: ItemSourceBinding) :
20         BaseViewHolder<Source, ItemSourceBinding>(itemView) {
21         override fun bind(data: Source) {
22             with(binding) {
23                 tvSource.text = data.name
24                 tvDesc.text = data.description
25
26                 root.setOnClickListener { it: View -
27                     onClick?.invoke(data)
28                 }
29             }
30         }
31     }
32
33     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
34         holder.bind(listData[position])
35     }
36
37     fun setOnClickData(onClick: (data: Source) -> Unit) {
38         this.onClick = onClick
39     }
40
41 }
```

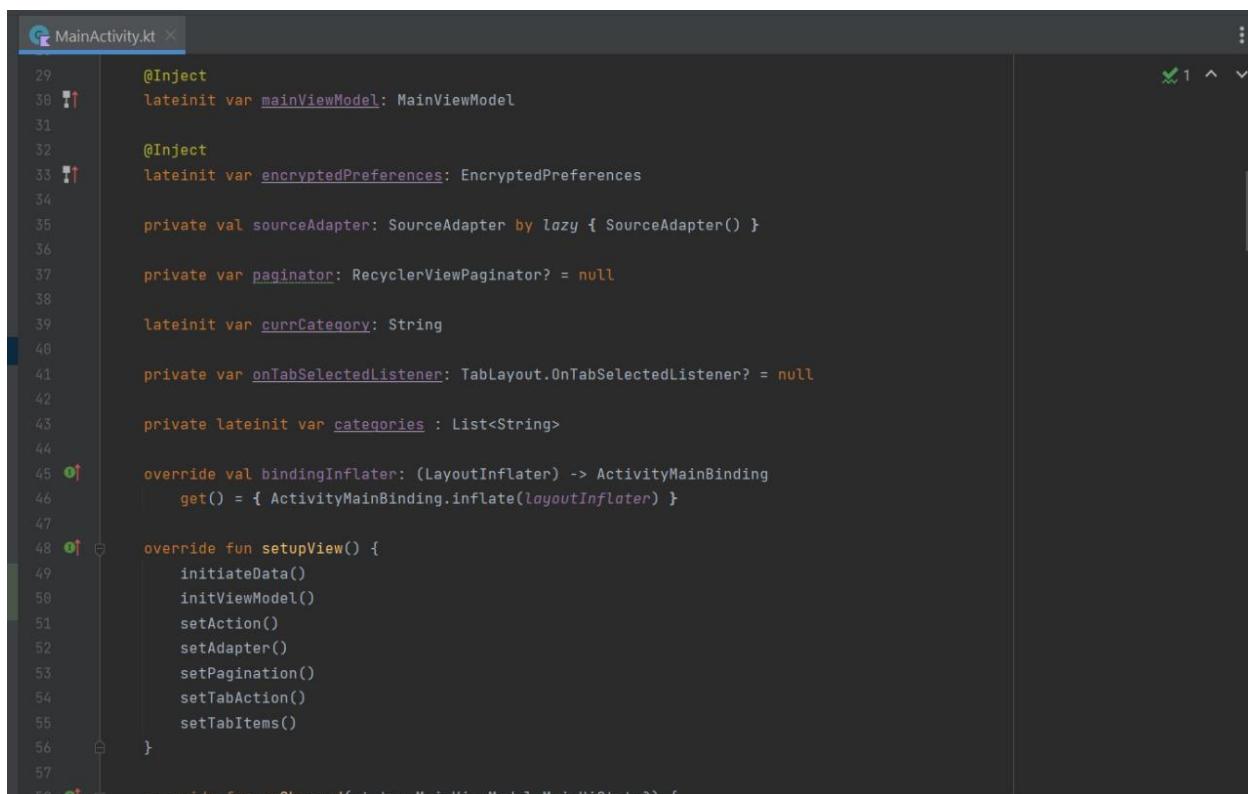


```
SourceAdapter.kt
14     override val holderInflater: (LayoutInflater, ViewGroup, Boolean) -> ViewHolder
15         get() = { inflator, viewGroup, boolean ->
16             ViewHolder(ItemSourceBinding.inflate(inflator, viewGroup, boolean))
17         }
18
19     inner class ViewHolder(itemView: ItemSourceBinding) :
20         BaseViewHolder<Source, ItemSourceBinding>(itemView) {
21         override fun bind(data: Source) {
22             with(binding) {
23                 tvSource.text = data.name
24                 tvDesc.text = data.description
25
26                 root.setOnClickListener { it: View -
27                     onClick?.invoke(data)
28                 }
29             }
30         }
31     }
32
33     override fun onBindViewHolder(holder: ViewHolder, position: Int) {
34         holder.bind(listData[position])
35     }
36
37     fun setOnClickData(onClick: (data: Source) -> Unit) {
38         this.onClick = onClick
39     }
40
41 }
```

CODE 6: MainActivity.kt



```
1 package com.aarush.newsapp.ui.main
2
3 import android.app.AlertDialog
4 import android.content.Intent
5 import android.view.LayoutInflater
6 import android.view.View
7 import android.widget.Toast
8 import androidx.appcompat.widget.SearchView
9 import androidx.lifecycle.Observer
10 import androidx.recyclerview.widget.LinearLayoutManager
11 import com.aarush.core.abstraction BaseActivityBinding
12 import com.aarush.core.pref.EncryptedPreferences
13 import com.aarush.core.util.exception.Failure
14 import com.aarush.core.util.extension.dp
15 import com.aarush.core.util.pagination.RecyclerViewPaginator
16 import com.aarush.newsapp.R
17 import com.aarush.newsapp.databinding.ActivityMainBinding
18 import com.aarush.newsapp.ui.articles.ArticlesActivity
19 import com.aarush.newsapp.ui.main.adapter.SourceAdapter
20 import com.aarush.newsapp.ui.search.SearchActivity
21 import com.google.android.material.tabs.TabLayout
22 import dagger.hilt.android.AndroidEntryPoint
23 import javax.inject.Inject
24
25 @AndroidEntryPoint
26 class MainActivity : BaseActivityBinding<ActivityMainBinding>(),
27     Observer<MainViewModel.MainUiState> {
28
29     @Inject
30     lateinit var mainViewModel: MainViewModel
```



```
29
30     @Inject
31     lateinit var mainViewModel: MainViewModel
32
33     @Inject
34     lateinit var encryptedPreferences: EncryptedPreferences
35
36     private val sourceAdapter: SourceAdapter by lazy { SourceAdapter() }
37
38     private var paginator: RecyclerViewPaginator? = null
39
40     lateinit var currCategory: String
41
42     private var onTabSelectedListener: TabLayout.OnTabSelectedListener? = null
43
44     private lateinit var categories: List<String>
45
46     override val bindingInflater: (LayoutInflater) -> ActivityMainBinding
47         get() = { ActivityMainBinding.inflate(layoutInflater) }
48
49     override fun setupView() {
50         initiateData()
51         initViewModel()
52         setAction()
53         setAdapter()
54         setPagination()
55         setTabAction()
56         setTabItems()
57     }
```

```
56     }
57
58     override fun onChanged(state: MainViewModel.MainUiState?) {
59         when (state) {
60             is MainViewModel.MainUiState.SourcesLoaded -> {
61                 stopLoading()
62                 if (state.data.isEmpty() && paginator?.isFirstGet == true) {
63                     showEmptyData()
64                 } else {
65                     hideEmptyData()
66                     sourceAdapter.appendList(state.data)
67                 }
68             }
69             is MainViewModel.MainUiState.InitialLoading -> {
70                 initialLoading()
71             }
72             is MainViewModel.MainUiState.PagingLoading -> {
73                 pagingLoading()
74             }
75             is MainViewModel.MainUiState.FailedLoaded -> {
76                 stopLoading()
77                 handleFailure(state.failure)
78             }
79             else -> {}
80         }
81     }
82
83     private fun initiateData() {
```

```
83
84     private fun initiateData() {
85         currCategory = "general"
86
87         categories= listOf(
88             "general",
89             "business",
90             "entertainment",
91             "health",
92             "science",
93             "sports",
94             "technology"
95         )
96     }
97
98     private fun initViewModel() {
99         mainViewModel.uiState.observe(owner: this, observer: this)
100         mainViewModel.getSourceByCategory(currCategory, page: 1)
101     }
102
103     private fun setAction() {
104         with(binding) { this.ActivityMainBinding
105             svNews.setOnQueryChangeListener(object : SearchView.OnQueryTextListener {
106                 override fun onQueryTextSubmit(query: String?): Boolean {
107                     hideKeyboard(svNews)
108                     if (query.toString().isNotEmpty()) {
109                         val intent = Intent(packageContext: this@MainActivity, SearchActivity::class.java)
110                         intent.putExtra(name: "searchQuery", query!!)
111                         startActivity(intent)
112                     }
113                 }
114             })
115         }
116     }
117 }
```

```
MainActivity.kt
```

```
110
111
112
113
114
115
116
117 ①
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
```

```
    intent.putExtra( name: "searchQuery", query!!)
    startActivity(intent)
    binding.svNews.setQuery("")

}

return true

}

override fun onQueryTextChange(newText: String?): Boolean {
    return false
}

)

svNews.setOnAdditionalButtonListener {
    showFilterDialog()
}

}

private fun setAdapter() {
    with(binding.rvNews) { this:RecyclerView
        adapter = sourceAdapter
        setHasFixedSize(true)

        sourceAdapter.setOnClickData { it:Source
            val intent = Intent( packageContext: this@MainActivity, ArticlesActivity::class.java)
            intent.putExtra( name: "sourceId", it.id)
            startActivity(intent)
        }
    }
}
```

```
MainActivity.kt
```

```
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
```

```
    startActivity(intent)
}

}

private fun setPagination() {
    paginator = RecyclerViewPaginator(binding.rvNews.layoutManager as LinearLayoutManager)
    paginator?.setOnLoadMoreListener { page ->
        paginator?.isFirstGet = false
        mainViewModel.getSourceByCategory(currCategory, page)
    }
    paginator?.let { binding.rvNews.addOnScrollListener(it) }
}

private fun setTabAction() {
    onTabSelectedListener = object : TabLayout.OnTabSelectedListener {
        override fun onTabSelected(tab: TabLayout.Tab?) {
            val tabPosition = tab?.position?.let { categories[it] }
            currCategory = tabPosition ?: "general"

            paginator?.isFirstGet = true
            sourceAdapter.clearList()
            mainViewModel.getSourceByCategory(currCategory, page: 1)
        }

        override fun onTabUnselected(tab: TabLayout.Tab?) {
        }

        override fun onTabReselected(tab: TabLayout.Tab?) {
        }
    }
}
```

```
164     override fun onTabReselected(tab: TabLayout.Tab?) {
165         ...
166     }
167 }
168
169 onTabSelectedListener?.let { it:TabLayout.OnTabSelectedListener
170     binding.tabLayoutCategory.addOnTabSelectedListener(it)
171 }
172 }
173
174 private fun setTabItems() {
175     with(binding.tabLayoutCategory) { this:CustomTabLayout
176         val tabList = mutableListOf<String>()
177         categories.forEach { it: String
178             tabList.add(it)
179         }
180         // Add tabs
181         addTitleOnlyTabs(tabList)
182
183         // Set margins
184         setTabsMargin(6.dp, 6.dp, 8.dp, 6.dp)
185     }
186 }
187
188 private fun initialLoading() {
189     binding.progressCircular.visibility = View.VISIBLE
190     binding.rvNews.visibility = View.GONE
191 }
192
```

```
188     private fun initialLoading() {
189         binding.progressCircular.visibility = View.VISIBLE
190         binding.rvNews.visibility = View.GONE
191     }
192
193     private fun pagingLoading() {
194         binding.progressCircular.visibility = View.VISIBLE
195     }
196
197     private fun stopLoading() {
198         binding.rvNews.visibility = View.VISIBLE
199         binding.progressCircular.visibility = View.GONE
200     }
201
202     private fun showEmptyData() {
203         binding.emptyDataView.parent.visibility = View.VISIBLE
204     }
205
206     private fun hideEmptyData() {
207         binding.emptyDataView.parent.visibility = View.GONE
208     }
209
210     private fun handleFailure(failure: Failure) {
211         Toast.makeText(context, failure.throwable.message, Toast.LENGTH_SHORT).show()
212     }
213
214     private fun showFilterDialog() {
215         val filterOptions = arrayOf("Search for Source", "Search for Article")
216         var filter = encryptedPreferences.filter
217     }
218
219     private fun updateUI(data: NewsResponse) {
220         binding.rvNews.adapter = NewsAdapter(data.articles)
221         binding.rvNews.layoutManager = LinearLayoutManager(context)
222     }
223
224     private fun checkForUpdates() {
225         binding.rvNews.layoutManager = LinearLayoutManager(context)
226         binding.rvNews.adapter = NewsAdapter(emptyList())
227     }
228
229     private fun searchSource() {
230         binding.rvNews.layoutManager = LinearLayoutManager(context)
231         binding.rvNews.adapter = NewsAdapter(emptyList())
232     }
233
234     private fun searchArticle() {
235         binding.rvNews.layoutManager = LinearLayoutManager(context)
236         binding.rvNews.adapter = NewsAdapter(emptyList())
237     }
238
239     private fun updateEmptyView() {
240         binding.emptyDataView.parent.visibility = View.GONE
241     }
242
243     private fun updateProgress() {
244         binding.progressCircular.visibility = View.GONE
245     }
246
247     private fun updateFailure() {
248         binding.progressCircular.visibility = View.GONE
249     }
250
251     private fun updatePaging() {
252         binding.progressCircular.visibility = View.GONE
253     }
254
255     private fun updateStop() {
256         binding.progressCircular.visibility = View.GONE
257     }
258
259     private fun updateEmpty() {
260         binding.emptyDataView.parent.visibility = View.GONE
261     }
262
263     private fun updateFailure() {
264         binding.emptyDataView.parent.visibility = View.GONE
265     }
266
267     private fun updatePaging() {
268         binding.emptyDataView.parent.visibility = View.GONE
269     }
270
271     private fun updateStop() {
272         binding.emptyDataView.parent.visibility = View.GONE
273     }
274
275     private fun updateEmpty() {
276         binding.emptyDataView.parent.visibility = View.GONE
277     }
278
279     private fun updateFailure() {
280         binding.emptyDataView.parent.visibility = View.GONE
281     }
282
283     private fun updatePaging() {
284         binding.emptyDataView.parent.visibility = View.GONE
285     }
286
287     private fun updateStop() {
288         binding.emptyDataView.parent.visibility = View.GONE
289     }
290
291     private fun updateEmpty() {
292         binding.emptyDataView.parent.visibility = View.GONE
293     }
294
295     private fun updateFailure() {
296         binding.emptyDataView.parent.visibility = View.GONE
297     }
298
299     private fun updatePaging() {
300         binding.emptyDataView.parent.visibility = View.GONE
301     }
302
303     private fun updateStop() {
304         binding.emptyDataView.parent.visibility = View.GONE
305     }
306
307     private fun updateEmpty() {
308         binding.emptyDataView.parent.visibility = View.GONE
309     }
310
311     private fun updateFailure() {
312         binding.emptyDataView.parent.visibility = View.GONE
313     }
314
315     private fun updatePaging() {
316         binding.emptyDataView.parent.visibility = View.GONE
317     }
318
319     private fun updateStop() {
320         binding.emptyDataView.parent.visibility = View.GONE
321     }
322
323     private fun updateEmpty() {
324         binding.emptyDataView.parent.visibility = View.GONE
325     }
326
327     private fun updateFailure() {
328         binding.emptyDataView.parent.visibility = View.GONE
329     }
330
331     private fun updatePaging() {
332         binding.emptyDataView.parent.visibility = View.GONE
333     }
334
335     private fun updateStop() {
336         binding.emptyDataView.parent.visibility = View.GONE
337     }
338
339     private fun updateEmpty() {
340         binding.emptyDataView.parent.visibility = View.GONE
341     }
342
343     private fun updateFailure() {
344         binding.emptyDataView.parent.visibility = View.GONE
345     }
346
347     private fun updatePaging() {
348         binding.emptyDataView.parent.visibility = View.GONE
349     }
350
351     private fun updateStop() {
352         binding.emptyDataView.parent.visibility = View.GONE
353     }
354
355     private fun updateEmpty() {
356         binding.emptyDataView.parent.visibility = View.GONE
357     }
358
359     private fun updateFailure() {
360         binding.emptyDataView.parent.visibility = View.GONE
361     }
362
363     private fun updatePaging() {
364         binding.emptyDataView.parent.visibility = View.GONE
365     }
366
367     private fun updateStop() {
368         binding.emptyDataView.parent.visibility = View.GONE
369     }
370
371     private fun updateEmpty() {
372         binding.emptyDataView.parent.visibility = View.GONE
373     }
374
375     private fun updateFailure() {
376         binding.emptyDataView.parent.visibility = View.GONE
377     }
378
379     private fun updatePaging() {
380         binding.emptyDataView.parent.visibility = View.GONE
381     }
382
383     private fun updateStop() {
384         binding.emptyDataView.parent.visibility = View.GONE
385     }
386
387     private fun updateEmpty() {
388         binding.emptyDataView.parent.visibility = View.GONE
389     }
390
391     private fun updateFailure() {
392         binding.emptyDataView.parent.visibility = View.GONE
393     }
394
395     private fun updatePaging() {
396         binding.emptyDataView.parent.visibility = View.GONE
397     }
398
399     private fun updateStop() {
400         binding.emptyDataView.parent.visibility = View.GONE
401     }
402
403     private fun updateEmpty() {
404         binding.emptyDataView.parent.visibility = View.GONE
405     }
406
407     private fun updateFailure() {
408         binding.emptyDataView.parent.visibility = View.GONE
409     }
410
411     private fun updatePaging() {
412         binding.emptyDataView.parent.visibility = View.GONE
413     }
414
415     private fun updateStop() {
416         binding.emptyDataView.parent.visibility = View.GONE
417     }
418
419     private fun updateEmpty() {
420         binding.emptyDataView.parent.visibility = View.GONE
421     }
422
423     private fun updateFailure() {
424         binding.emptyDataView.parent.visibility = View.GONE
425     }
426
427     private fun updatePaging() {
428         binding.emptyDataView.parent.visibility = View.GONE
429     }
430
431     private fun updateStop() {
432         binding.emptyDataView.parent.visibility = View.GONE
433     }
434
435     private fun updateEmpty() {
436         binding.emptyDataView.parent.visibility = View.GONE
437     }
438
439     private fun updateFailure() {
440         binding.emptyDataView.parent.visibility = View.GONE
441     }
442
443     private fun updatePaging() {
444         binding.emptyDataView.parent.visibility = View.GONE
445     }
446
447     private fun updateStop() {
448         binding.emptyDataView.parent.visibility = View.GONE
449     }
450
451     private fun updateEmpty() {
452         binding.emptyDataView.parent.visibility = View.GONE
453     }
454
455     private fun updateFailure() {
456         binding.emptyDataView.parent.visibility = View.GONE
457     }
458
459     private fun updatePaging() {
460         binding.emptyDataView.parent.visibility = View.GONE
461     }
462
463     private fun updateStop() {
464         binding.emptyDataView.parent.visibility = View.GONE
465     }
466
467     private fun updateEmpty() {
468         binding.emptyDataView.parent.visibility = View.GONE
469     }
470
471     private fun updateFailure() {
472         binding.emptyDataView.parent.visibility = View.GONE
473     }
474
475     private fun updatePaging() {
476         binding.emptyDataView.parent.visibility = View.GONE
477     }
478
479     private fun updateStop() {
480         binding.emptyDataView.parent.visibility = View.GONE
481     }
482
483     private fun updateEmpty() {
484         binding.emptyDataView.parent.visibility = View.GONE
485     }
486
487     private fun updateFailure() {
488         binding.emptyDataView.parent.visibility = View.GONE
489     }
490
491     private fun updatePaging() {
492         binding.emptyDataView.parent.visibility = View.GONE
493     }
494
495     private fun updateStop() {
496         binding.emptyDataView.parent.visibility = View.GONE
497     }
498
499     private fun updateEmpty() {
500         binding.emptyDataView.parent.visibility = View.GONE
501     }
502
503     private fun updateFailure() {
504         binding.emptyDataView.parent.visibility = View.GONE
505     }
506
507     private fun updatePaging() {
508         binding.emptyDataView.parent.visibility = View.GONE
509     }
510
511     private fun updateStop() {
512         binding.emptyDataView.parent.visibility = View.GONE
513     }
514
515     private fun updateEmpty() {
516         binding.emptyDataView.parent.visibility = View.GONE
517     }
518
519     private fun updateFailure() {
520         binding.emptyDataView.parent.visibility = View.GONE
521     }
522
523     private fun updatePaging() {
524         binding.emptyDataView.parent.visibility = View.GONE
525     }
526
527     private fun updateStop() {
528         binding.emptyDataView.parent.visibility = View.GONE
529     }
530
531     private fun updateEmpty() {
532         binding.emptyDataView.parent.visibility = View.GONE
533     }
534
535     private fun updateFailure() {
536         binding.emptyDataView.parent.visibility = View.GONE
537     }
538
539     private fun updatePaging() {
540         binding.emptyDataView.parent.visibility = View.GONE
541     }
542
543     private fun updateStop() {
544         binding.emptyDataView.parent.visibility = View.GONE
545     }
546
547     private fun updateEmpty() {
548         binding.emptyDataView.parent.visibility = View.GONE
549     }
550
551     private fun updateFailure() {
552         binding.emptyDataView.parent.visibility = View.GONE
553     }
554
555     private fun updatePaging() {
556         binding.emptyDataView.parent.visibility = View.GONE
557     }
558
559     private fun updateStop() {
560         binding.emptyDataView.parent.visibility = View.GONE
561     }
562
563     private fun updateEmpty() {
564         binding.emptyDataView.parent.visibility = View.GONE
565     }
566
567     private fun updateFailure() {
568         binding.emptyDataView.parent.visibility = View.GONE
569     }
570
571     private fun updatePaging() {
572         binding.emptyDataView.parent.visibility = View.GONE
573     }
574
575     private fun updateStop() {
576         binding.emptyDataView.parent.visibility = View.GONE
577     }
578
579     private fun updateEmpty() {
580         binding.emptyDataView.parent.visibility = View.GONE
581     }
582
583     private fun updateFailure() {
584         binding.emptyDataView.parent.visibility = View.GONE
585     }
586
587     private fun updatePaging() {
588         binding.emptyDataView.parent.visibility = View.GONE
589     }
590
591     private fun updateStop() {
592         binding.emptyDataView.parent.visibility = View.GONE
593     }
594
595     private fun updateEmpty() {
596         binding.emptyDataView.parent.visibility = View.GONE
597     }
598
599     private fun updateFailure() {
599     }
```

```
215     val filterOptions = arrayOf("Search for Source", "Search for Article")
216     var filter = encryptedPreferences.filter
217     val builder = AlertDialog.Builder(context)
218     builder.setTitle("Search For")
219     .setSingleChoiceItems(filterOptions, if (filter == SEARCH_SOURCE) 0 else 1) { _, which ->
220         when (which) {
221             0 -> {
222                 filter = SEARCH_SOURCE
223             }
224             1 -> {
225                 filter = SEARCH_ARTICLE
226             }
227         }
228     }
229     .setNegativeButton("Cancel") { dialog, _ ->
230         dialog.dismiss()
231     }
232     .setPositiveButton("Apply") { dialog, _ ->
233         // Apply button clicked
234         encryptedPreferences.filter = filter
235         dialog.dismiss()
236     }
237     val dialog = builder.create()
238     dialog.show()
239 }
240 companion object {
241     const val SEARCH_SOURCE = 0
242     const val SEARCH_ARTICLE = 1
243 }
244
```

CODE 7: MainViewModel.kt

```
1 package com.aarush.newsapp.ui.main
2
3 import androidx.lifecycle.viewModelScope
4 import com.aarush.core.abstraction.BaseViewModel
5 import com.aarush.core.dispatcher.DispatcherProvider
6 import com.aarush.core.model.model.Source
7 import com.aarush.core.model.usecase.GetSourcesByCategoryUseCase
8 import com.aarush.core.util.exception.Failure
9 import com.aarush.core.util.extension.onError
10 import com.aarush.core.util.extension.onSuccess
11 import kotlinx.coroutines.flow.collectLatest
12 import kotlinx.coroutines.flow.onStart
13 import kotlinx.coroutines.launch
14 import kotlinx.coroutines.withContext
15 import javax.inject.Inject
16
17 class MainViewModel @Inject constructor(
18     private val dispatcherProvider: DispatcherProvider,
19     private val getSourcesByCategory: GetSourcesByCategoryUseCase,
20 ) : BaseViewModel<MainViewModel.MainUiState>() {
21
22     sealed class MainUiState {
23         object InitialLoading: MainUiState()
24         object PagingLoading: MainUiState()
25         data class SourcesLoaded(val data: List<Source>): MainUiState()
26         data class FailedLoaded(val failure: Failure): MainUiState()
27     }
28
29     fun getSourceByCategory(category: String, page: Long){
30         _uiState.value = if (page == 1L) f
```

```
22     sealed class MainUiState {
23         object InitialLoading: MainUiState()
24         object PagingLoading: MainUiState()
25         data class SourcesLoaded(val data: List<Source>) : MainUiState()
26         data class FailedLoaded(val failure: Failure) : MainUiState()
27     }
28
29     fun getSourceByCategory(category: String, page: Long){
30         _uiState.value = if (page == 1L) {
31             MainUiState.InitialLoading
32         }else{
33             MainUiState.PagingLoading
34         }
35         viewModelScope.launch(dispatcherProvider.io) { this:CoroutineScope
36             getSourcesByCategory.run(GetSourcesByCategoryUseCase.Params(category, page.toInt()))
37                 .onSuccess { it:List<Source>
38                     withContext(dispatcherProvider.main) { this:CoroutineScope
39                         _uiState.value = MainUiState.SourcesLoaded(it)
40                     }
41                 }
42                 .onError { it:Failure
43                     withContext(dispatcherProvider.main) { this:CoroutineScope
44                         _uiState.value = MainUiState.FailedLoaded(it)
45                     }
46                 }
47             }
48         }
49     }
50 }
```

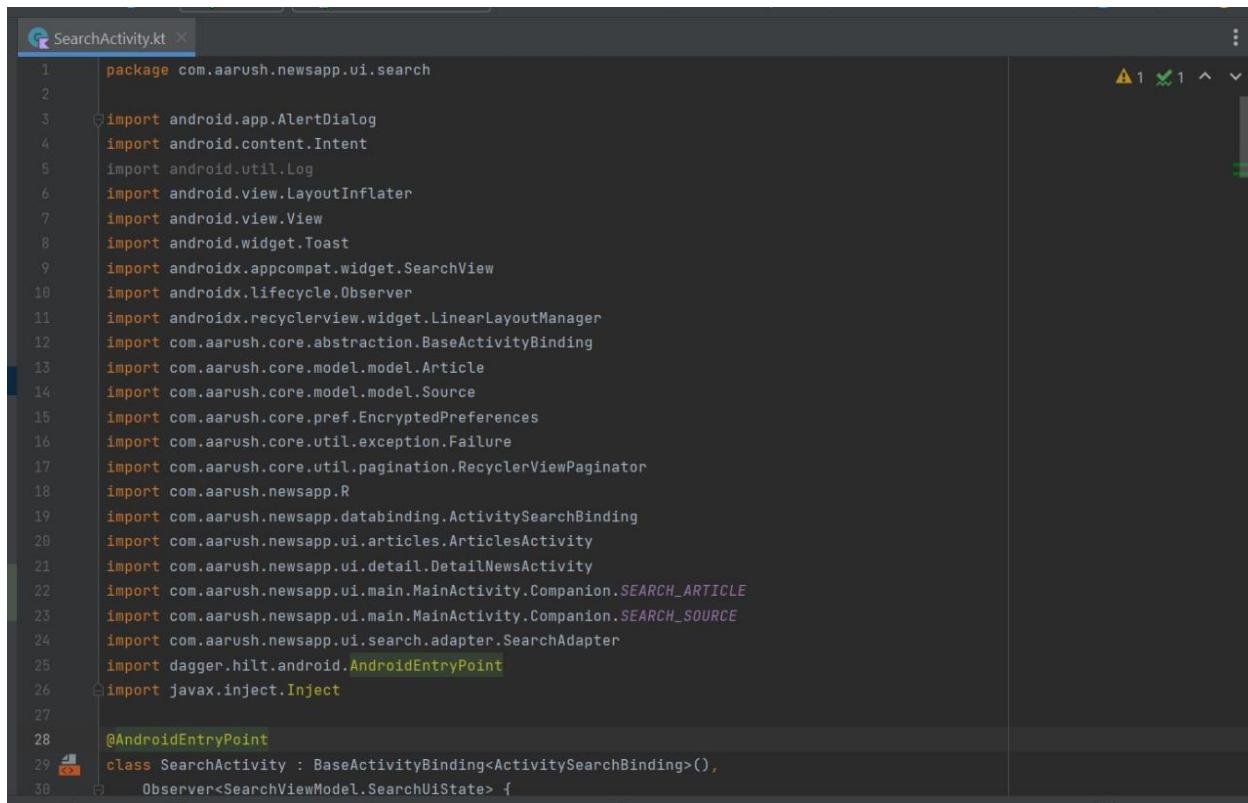
CODE 8: SearchAdapter.kt

```
1 package com.aarush.newsapp.ui.search.adapter
2
3 import android.view.LayoutInflater
4 import android.view.ViewGroup
5 import androidx.recyclerview.widget.RecyclerView
6 import com.aarush.core.abstraction.PagingRecyclerViewAdapter
7 import com.aarush.core.model.model.Article
8 import com.aarush.core.model.model.Source
9 import com.aarush.core.util.extension.glide
10 import com.aarush.newsapp.databinding.ItemArticlesBinding
11 import com.aarush.newsapp.databinding.ItemSourceBinding
12 import com.aarush.newsapp.ui.main.MainActivity.Companion.SEARCH_ARTICLE
13 import com.aarush.newsapp.ui.main.MainActivity.Companion.SEARCH_SOURCE
14
15 class SearchAdapter : PagingRecyclerViewAdapter<RecyclerView.ViewHolder, Any>() {
16
17     private var onClick: ((Any, Int) -> Unit)? = null
18     private var viewType = SEARCH_SOURCE
19
20     fun setTypeAdapter(type: Int) {
21         viewType = type
22         notifyDataSetChanged()
23     }
24
25     override val holderInflater: (LayoutInflater, ViewGroup, Boolean) -> RecyclerView.ViewHolder
26         get() = { inflater, viewGroup, boolean -
27             when (viewType) {
28                 SEARCH_SOURCE -> {
29                     SourceViewHolder(ItemSourceBinding.inflate(inflater, viewGroup, boolean)) ^lambda
30                 }
31             }
32         }
33 }
```

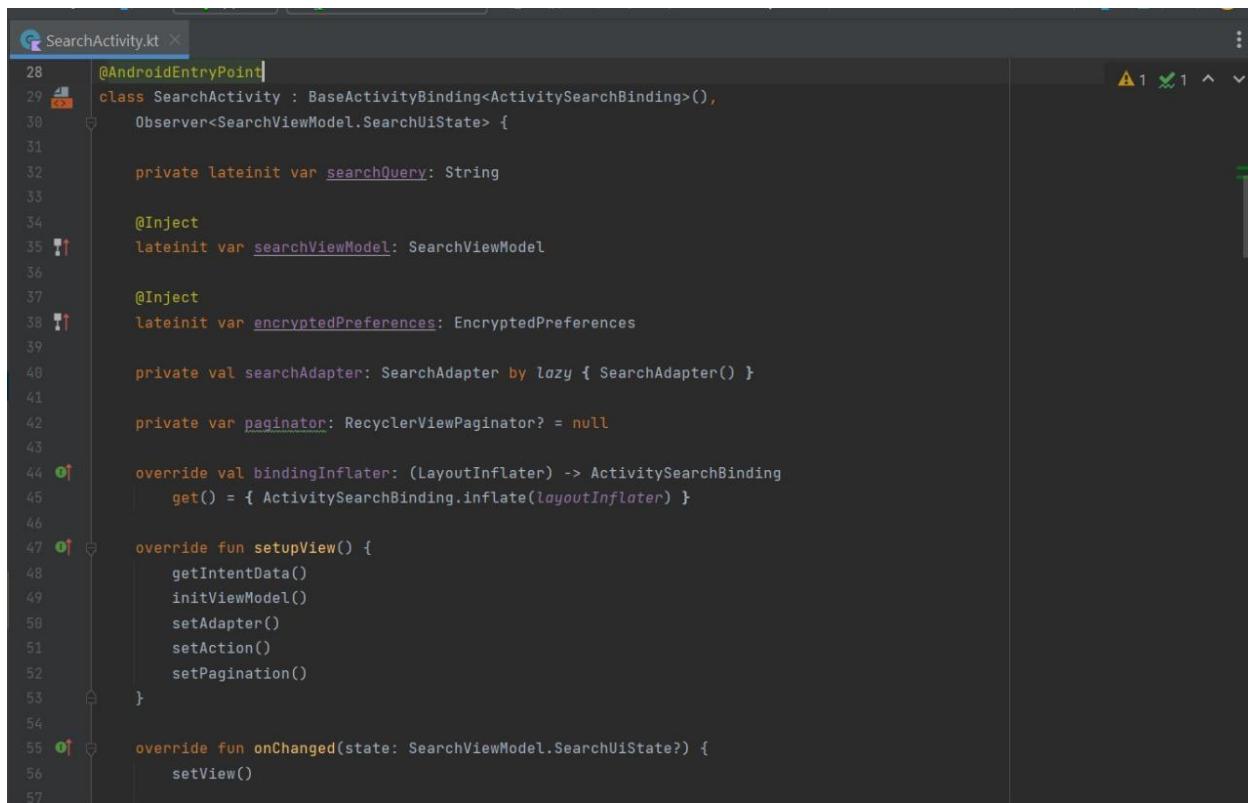
```
28     SEARCH_SOURCE -> {
29         SourceViewHolder(ItemSourceBinding.inflate(inflater, viewGroup, boolean)) ^lambda
30     }
31     SEARCH_ARTICLE -> {
32         ArticleViewHolder(ItemArticlesBinding.inflate(inflater, viewGroup, boolean)) ^lambda
33     }
34     else -> throw IllegalArgumentException("Invalid view type")
35 }
36 }
37
38 inner class SourceViewHolder(private val binding: ItemSourceBinding) : RecyclerView.ViewHolder(binding.root) {
39     fun bind(data: Source) {
40         with(binding) { this:ItemSourceBinding
41             tvSource.text = data.name
42             tvDesc.text = data.description
43
44             root.setOnClickListener { it:View!
45                 onClick?.invoke(data, viewType)
46             }
47         }
48     }
49 }
50
51 inner class ArticleViewHolder(private val binding: ItemArticlesBinding) : RecyclerView.ViewHolder(binding.root) {
52     fun bind(data: Article) {
53         with(binding) { this:ItemArticlesBinding
54             tvTitle.text = data.title
55             tvDesc.text = data.description
56             imgArticle.glide(itemView, data.image)
57         }
58     }
59 }
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76 }
```

```
49 }
50
51 inner class ArticleViewHolder(private val binding: ItemArticlesBinding) : RecyclerView.ViewHolder(binding.root) {
52     fun bind(data: Article) {
53         with(binding) { this:ItemArticlesBinding
54             tvTitle.text = data.title
55             tvDesc.text = data.description
56             imgArticle.glide(itemView, data.image)
57
58             root.setOnClickListener { it:View!
59                 onClick?.invoke(data, viewType)
60             }
61         }
62     }
63 }
64
65 ① override fun onBindViewHolder(holder: RecyclerView.ViewHolder, position: Int) {
66     when (holder) {
67         is SourceViewHolder -> holder.bind(listData[position] as Source)
68         is ArticleViewHolder -> holder.bind(listData[position] as Article)
69     }
70 }
71
72 fun setOnClickData(onClick: (Any, Int) -> Unit) {
73     this.onClick = onClick
74 }
75
76 }
```

CODE 9: SearchActivity.kt



```
1 package com.aarush.newsapp.ui.search
2
3 import android.app.AlertDialog
4 import android.content.Intent
5 import android.util.Log
6 import android.view.LayoutInflater
7 import android.view.View
8 import android.widget.Toast
9 import androidx.appcompat.widget.SearchView
10 import androidx.lifecycle.Observer
11 import androidx.recyclerview.widget.LinearLayoutManager
12 import com.aarush.core.abstraction BaseActivityBinding
13 import com.aarush.core.model.Article
14 import com.aarush.core.model.model.Source
15 import com.aarush.core.pref.EncryptedPreferences
16 import com.aarush.core.util.exception.Failure
17 import com.aarush.core.util.pagination.RecyclerViewPaginator
18 import com.aarush.newsapp.R
19 import com.aarush.newsapp.databinding.ActivitySearchBinding
20 import com.aarush.newsapp.articles.ArticlesActivity
21 import com.aarush.newsapp.ui.detail.DetailNewsActivity
22 import com.aarush.newsapp.ui.main.MainActivity.Companion.SEARCH_ARTICLE
23 import com.aarush.newsapp.ui.main.MainActivity.Companion.SEARCH_SOURCE
24 import com.aarush.newsapp.ui.search.adapter.SearchAdapter
25 import dagger.hilt.android.AndroidEntryPoint
26 import javax.inject.Inject
27
28 @AndroidEntryPoint
29 class SearchActivity : BaseActivityBinding<ActivitySearchBinding>(),
30     Observer<SearchViewModel.SearchUiState> {
```



```
28 @AndroidEntryPoint
29 class SearchActivity : BaseActivityBinding<ActivitySearchBinding>(),
30     Observer<SearchViewModel.SearchUiState> {
31
32     private lateinit var searchQuery: String
33
34     @Inject
35     lateinit var searchViewModel: SearchViewModel
36
37     @Inject
38     lateinit var encryptedPreferences: EncryptedPreferences
39
40     private val searchAdapter: SearchAdapter by lazy { SearchAdapter() }
41
42     private var paginator: RecyclerViewPaginator? = null
43
44     override val bindingInflater: (LayoutInflater) -> ActivitySearchBinding
45         get() = { ActivitySearchBinding.inflate(layoutInflater) }
46
47     override fun setupView() {
48         getIntentData()
49         initViewModel()
50         setAdapter()
51         setAction()
52         setPagination()
53     }
54
55     override fun onChanged(state: SearchViewModel.SearchUiState?) {
56         setView()
57     }
```

A screenshot of a code editor showing a section of `SearchActivity.kt`. The code is a large `when` expression based on `state`. It handles four main cases:

- `is SearchViewModel.SearchUiState.SourcesLoaded`:
 - `if (state.data.isEmpty() & & paginator?.isFirstGet == true)`:
 - `showEmptyData()`
 - `else`:
 - `hideEmptyData()`
 - `searchAdapter.appendList(state.data)`- `is SearchViewModel.SearchUiState.ArticleLoaded`:
 - `stopLoading()`
 - `if (state.data.isEmpty() & & paginator?.isFirstGet == true)`:
 - `showEmptyData()`
 - `else`:
 - `hideEmptyData()`
 - `searchAdapter.appendList(state.data)`
- `is SearchViewModel.SearchUiState.InitialLoading`:
 - `initialLoading()`
- `is SearchViewModel.SearchUiState.PagingLoading`:
 - `pagingLoading()`

The code editor interface includes a status bar at the top right with icons for file operations and a vertical scroll bar on the right.

A screenshot of a code editor showing implementation details in `SearchActivity.kt`. The code includes:

- `private fun getIntentData()`:
 - `searchQuery = intent.getStringExtra(name: "searchQuery") ?: ""`
- `private fun initViewModel()`:
 - `searchViewModel.uiState.observe(owner: this, observer: this)`
 - `if (encryptedPreferences.filter == SEARCH_SOURCE)`:
 - `searchViewModel.getSourceByName(searchQuery, page: 1)`
 - `else`:
 - `searchViewModel.getArticlesByQuery(searchQuery, page: 1)`
- `private fun setView()`:
 - `binding.tvResult.text = "Result for {searchQuery}"`
- `private fun setAction()`:
 - `with(binding) { this: ActivitySearchBinding`

The code editor interface includes a status bar at the top right with icons for file operations and a vertical scroll bar on the right.

```
114
115     private fun setAction() {
116         with(binding) { this: ActivitySearchBinding
117             svNews.setQuery(searchQuery)
118             svNews.setOnQueryChangeListener(object : SearchView.OnQueryTextListener {
119                 override fun onQueryTextSubmit(query: String?): Boolean {
120                     hideKeyboard(svNews)
121                     if (query.toString().isNotEmpty()) {
122                         searchQuery = query!!
123                         if (encryptedPreferences.filter == SEARCH_SOURCE) {
124                             searchViewModel.getArticlesByQuery(searchQuery, page: 1)
125                             searchAdapter.setTypeAdapter(SEARCH_SOURCE)
126                             searchAdapter.clearList()
127                             binding.rvNews.adapter = searchAdapter
128                         } else {
129                             searchViewModel.getArticlesByQuery(searchQuery, page: 1)
130                             searchAdapter.setTypeAdapter(SEARCH_ARTICLE)
131                             searchAdapter.clearList()
132                             binding.rvNews.adapter = searchAdapter
133                         }
134                     }
135                     return true
136                 }
137
138                 override fun onQueryTextChange(newText: String?): Boolean {
139
140                     return false
141                 }
142             })
143         }
144     }
145
146     svNews.setOnAdditionalButtonListener {
147         showFilterDialog()
148     }
149
150     btnBack.setOnClickListener { onBackPressedDispatcher.onBackPressed() }
151 }
152
153
154     private fun setAdapter() {
155         with(binding.rvNews) { this: RecyclerView
156             searchAdapter.setTypeAdapter(encryptedPreferences.filter)
157             adapter = searchAdapter
158             setHasFixedSize(true)
159
160             searchAdapter.setOnClickData { it, filter ->
161                 if (filter == SEARCH_SOURCE) {
162                     val data = it as Source
163                     val intent = Intent( packageContext: this@SearchActivity, ArticlesActivity::class.java)
164                     intent.putExtra( name: "sourceId", data.id)
165                     startActivity(intent)
166                 } else {
167                     val data = it as Article
168                     val intent = Intent( packageContext: this@SearchActivity, DetailNewsActivity::class.java)
169                     intent.putExtra( name: "sourceNews", it.source)
170                     intent.putExtra( name: "urlNews", data.url)
171             }
172         }
173     }
174 }
```

```
141     }
142
143     }
144
145     svNews.setOnAdditionalButtonListener {
146         showFilterDialog()
147     }
148
149     btnBack.setOnClickListener { onBackPressedDispatcher.onBackPressed() }
150
151 }
152
153
154     private fun setAdapter() {
155         with(binding.rvNews) { this: RecyclerView
156             searchAdapter.setTypeAdapter(encryptedPreferences.filter)
157             adapter = searchAdapter
158             setHasFixedSize(true)
159
160             searchAdapter.setOnClickData { it, filter ->
161                 if (filter == SEARCH_SOURCE) {
162                     val data = it as Source
163                     val intent = Intent( packageContext: this@SearchActivity, ArticlesActivity::class.java)
164                     intent.putExtra( name: "sourceId", data.id)
165                     startActivity(intent)
166                 } else {
167                     val data = it as Article
168                     val intent = Intent( packageContext: this@SearchActivity, DetailNewsActivity::class.java)
169                     intent.putExtra( name: "sourceNews", it.source)
170                     intent.putExtra( name: "urlNews", data.url)
171             }
172         }
173     }
174 }
```

```
171     startActivity(intent)
172 }
173 }
174 }
175 }
176
177 private fun setPagination() {
178     paginator = RecyclerViewPaginator(binding.rvNews.layoutManager as LinearLayoutManager)
179     paginator?.setOnLoadMoreListener { page ->
180         paginator?.isFirstGet = false
181         if (encryptedPreferences.filter == SEARCH_SOURCE) {
182             searchViewModel.getSourceByName(searchQuery, page)
183         } else {
184             searchViewModel.getArticlesByQuery(searchQuery, page)
185         }
186     }
187     paginator?.let { binding.rvNews.addOnScrollListener(it) }
188 }
189
190 private fun initialLoading() {
191     binding.progressCircular.visibility = View.VISIBLE
192     binding.rvNews.visibility = View.GONE
193 }
194
195 private fun pagingLoading() {
196     binding.progressCircular.visibility = View.VISIBLE
197 }
198
199 private fun stopLoading() {
200     binding.rvNews.visibility = View.VISIBLE

```

```
198
199 private fun stopLoading() {
200     binding.rvNews.visibility = View.VISIBLE
201     binding.progressCircular.visibility = View.GONE
202 }
203
204 private fun showEmptyData() {
205     binding.emptyDataView.parent.visibility = View.VISIBLE
206 }
207
208 private fun hideEmptyData() {
209     binding.emptyDataView.parent.visibility = View.GONE
210 }
211
212 private fun handleFailure(failure: Failure) {
213     Toast.makeText(context, failure.throwable.message, Toast.LENGTH_SHORT).show()
214 }
215
216 private fun showFilterDialog() {
217     val filterOptions = arrayOf("Search for Source", "Search for Article")
218     var filter = encryptedPreferences.filter
219     val builder = AlertDialog.Builder(context, this)
220     builder.setTitle("Search For")
221     .setSingleChoiceItems(filterOptions, if (filter == SEARCH_SOURCE) 0 else 1) { _, which ->
222         when (which) {
223             0 -> {
224                 filter = SEARCH_SOURCE
225             }
226             1 -> {
227                 filter = SEARCH_ARTICLE

```

```
1 val filterOptions = arrayOf("Search for Source", "Search for Article")
2 var filter = encryptedPreferences.filter
3 val builder = AlertDialog.Builder(context)
4 builder.setTitle("Search For")
5     .setSingleChoiceItems(filterOptions, if (filter == SEARCH_SOURCE) 0 else 1) { _, which ->
6         when (which) {
7             0 -> {
8                 filter = SEARCH_SOURCE
9             }
10            1 -> {
11                filter = SEARCH_ARTICLE
12            }
13        }
14    }
15    .setNegativeButton("Cancel") { dialog, _ ->
16        dialog.dismiss()
17    }
18    .setPositiveButton("Apply") { dialog, _ ->
19        // Apply button clicked
20        encryptedPreferences.filter = filter
21        dialog.dismiss()
22    }
23
24    val dialog = builder.create()
25    dialog.show()
26}
27
```

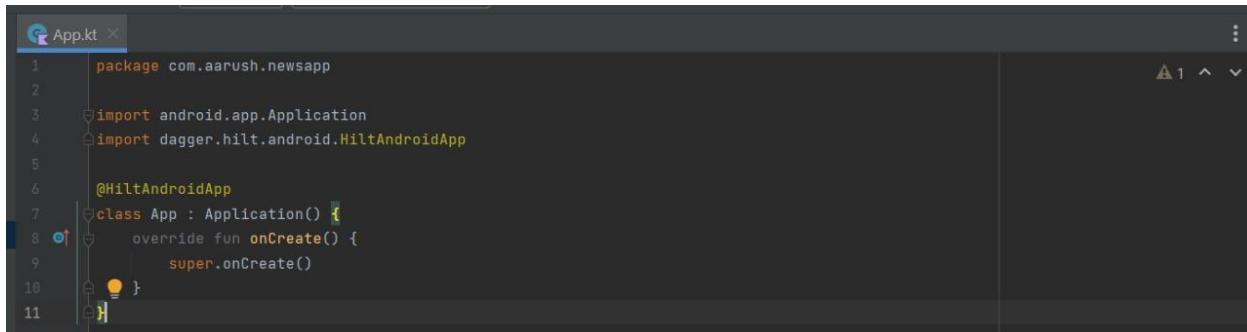
CODE 10: Search View Model

```
1 package com.aarush.newsapp.ui.search
2
3 import android.util.Log
4 import androidx.lifecycle.viewModelScope
5 import com.aarush.core.abstraction.BaseViewModel
6 import com.aarush.core.dispatcher.DispatcherProvider
7 import com.aarush.core.model.Article
8 import com.aarush.core.model.model.Source
9 import com.aarush.core.model.usecase.GetArticlesByQueryUseCase
10 import com.aarush.core.model.usecase.GetSourceByNameUseCase
11 import com.aarush.core.util.exception.Failure
12 import com.aarush.core.util.extension.onError
13 import com.aarush.core.util.extension.onSuccess
14 import kotlinx.coroutines.launch
15 import kotlinx.coroutines.withContext
16 import javax.inject.Inject
17
18 class SearchViewModel @Inject constructor(
19     private val dispatcherProvider: DispatcherProvider,
20     private val getSourceByNameUseCase: GetSourceByNameUseCase,
21     private val getArticlesByQueryUseCase: GetArticlesByQueryUseCase,
22 ) : BaseViewModel<SearchViewModel.SearchUiState>() {
23
24     sealed class SearchUiState {
25         object InitialLoading : SearchUiState()
26         object PagingLoading : SearchUiState()
27         data class SourcesLoaded(val data: List<Source>) : SearchUiState()
28         data class ArticleLoaded(val data: List<Article>) : SearchUiState()
29         data class FailedLoaded(val failure: Failure) : SearchUiState()
30     }
31 }
```

```
SearchViewModel.kt x
28         data class ArticleLoaded(val data: List<Article>) : SearchUiState()
29         data class FailedLoaded(val failure: Failure) : SearchUiState()
30     }
31
32     fun getSourceByName(name: String, page: Long) {
33         _uiState.value = if (page == 1L) {
34             SearchUiState.InitialLoading
35         } else {
36             SearchUiState.PagingLoading
37         }
38         viewModelScope.launch(dispatcherProvider.io) { this:CoroutineScope
39             val data = getSourceByNameUseCase.getSourceByName(name, page.toInt())
40             val dataString = data.joinToString(separator = ", ")
41             Log.d(tag: "datalistv", dataString)
42             withContext(dispatcherProvider.main) { this:CoroutineScope
43                 _uiState.value = SearchUiState.SourcesLoaded(data)
44             }
45         }
46     }
47
48     fun getArticlesByQuery(source: String, page: Long) {
49         _uiState.value = if (page == 1L) {
50             SearchUiState.InitialLoading
51         } else {
52             SearchUiState.PagingLoading
53         }
54         viewModelScope.launch(dispatcherProvider.io) { this:CoroutineScope
55             getArticlesByQueryUseCase.run(GetArticlesByQueryUseCase.Params(source, page.toString()))
56             .onSuccess { it:List<Article>
57                 withContext(dispatcherProvider.main) { this:CoroutineScope
58                     _uiState.value = SearchUiState.SourcesLoaded(it)
59                 }
60             }
61             .onError { it:Failure
62                 withContext(dispatcherProvider.main) { this:CoroutineScope
63                     _uiState.value = SearchUiState.FailedLoaded(it)
64                 }
65             }
66         }
67     }
68
69 }
```

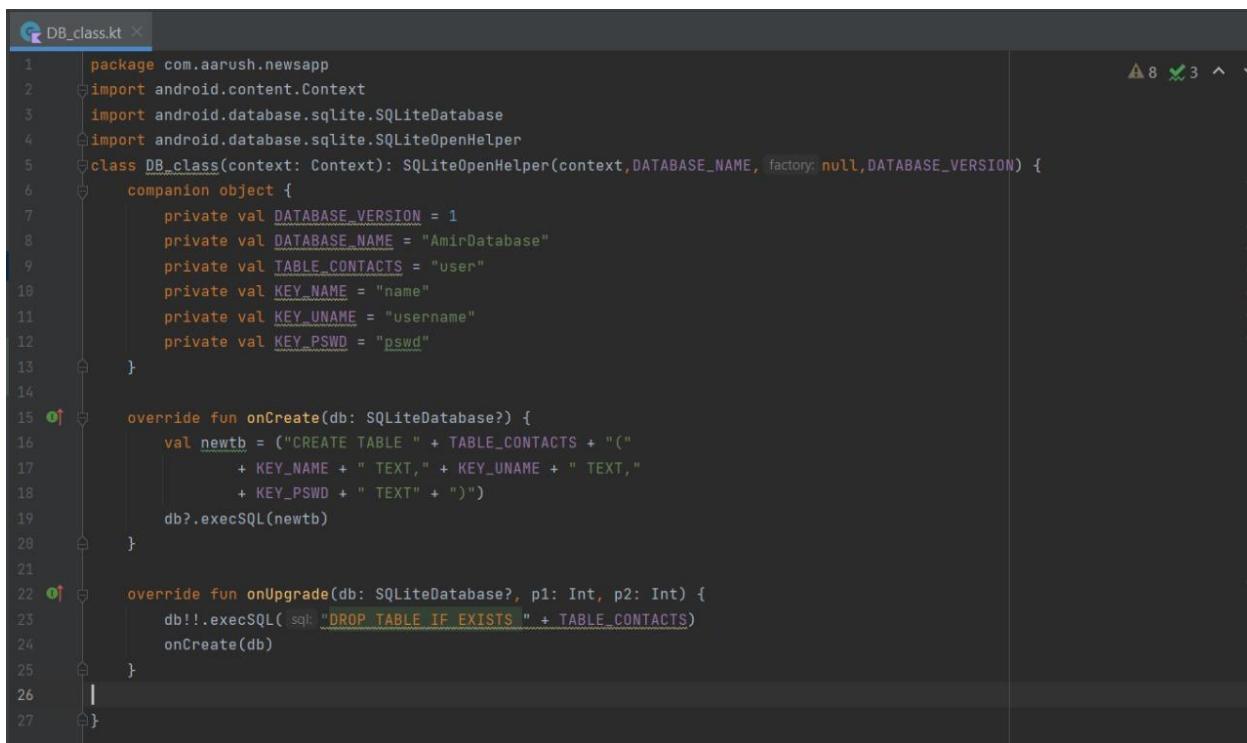
```
SearchViewModel.kt x
42         withContext(dispatcherProvider.main) { this:CoroutineScope
43             _uiState.value = SearchUiState.SourcesLoaded(data)
44         }
45     }
46
47
48     fun getArticlesByQuery(source: String, page: Long) {
49         _uiState.value = if (page == 1L) {
50             SearchUiState.InitialLoading
51         } else {
52             SearchUiState.PagingLoading
53         }
54         viewModelScope.launch(dispatcherProvider.io) { this:CoroutineScope
55             getArticlesByQueryUseCase.run(GetArticlesByQueryUseCase.Params(source, page.toString()))
56             .onSuccess { it:List<Article>
57                 withContext(dispatcherProvider.main) { this:CoroutineScope
58                     _uiState.value = SearchUiState.ArticleLoaded(it)
59                 }
60             }
61             .onError { it:Failure
62                 withContext(dispatcherProvider.main) { this:CoroutineScope
63                     _uiState.value = SearchUiState.FailedLoaded(it)
64                 }
65             }
66         }
67     }
68
69 }
```

CODE 11: App.kt



```
App.kt
1 package com.aarush.newsapp
2
3 import android.app.Application
4 import dagger.hilt.android.HiltAndroidApp
5
6 @HiltAndroidApp
7 class App : Application() {
8     override fun onCreate() {
9         super.onCreate()
10    }
11 }
```

CODE 12: DB_class.kt



```
DB_class.kt
1 package com.aarush.newsapp
2
3 import android.content.Context
4 import android.database.sqlite.SQLiteDatabase
5 import android.database.sqlite.SQLiteOpenHelper
6
7 class DB_class(context: Context): SQLiteOpenHelper(context, DATABASE_NAME, null, DATABASE_VERSION) {
8     companion object {
9         private val DATABASE_VERSION = 1
10        private val DATABASE_NAME = "AmirDatabase"
11        private val TABLE_CONTACTS = "user"
12        private val KEY_NAME = "name"
13        private val KEY_UNAME = "username"
14        private val KEY_PSWD = "pswd"
15    }
16
17    override fun onCreate(db: SQLiteDatabase?) {
18        val newtb = ("CREATE TABLE " + TABLE_CONTACTS + "("
19            + KEY_NAME + " TEXT," + KEY_UNAME + " TEXT,"
20            + KEY_PSWD + " TEXT" + ")")
21        db?.execSQL(newtb)
22    }
23
24    override fun onUpgrade(db: SQLiteDatabase?, p1: Int, p2: Int) {
25        db.execSQL("DROP TABLE IF EXISTS " + TABLE_CONTACTS)
26        onCreate(db)
27    }
28 }
```

CODE 13: login_form.kt

```
login_form.kt
1 package com.aarush.newsapp
2
3 import android.content.Intent
4 import androidx.appcompat.app.AppCompatActivity
5 import android.os.Bundle
6 import android.view.View
7 import android.widget.Toast
8 import androidx.appcompat.AlertDialog
9 import com.aarush.newsapp.databinding.ActivityLoginFormBinding
10 import com.aarush.newsapp.ui.main.MainActivity
11
12 class login_form : AppCompatActivity() {
13     private lateinit var bind: ActivityLoginFormBinding
14     private var isPasswordVisible = false
15
16     override fun onCreate(savedInstanceState: Bundle?) {
17         super.onCreate(savedInstanceState)
18         bind = ActivityLoginFormBinding.inflate(layoutInflater)
19         setContentView(bind.root)
20         val dbhelp = DB.class(applicationContext)
21         val db = dbhelp.readableDatabase
22
23         bind.btnlogin.setOnClickListener { it: View! }
24             val username = bind.logtxt.text.toString()
25             val password = bind.ed3.text.toString()
26             val query = "SELECT * FROM user WHERE username='$username' AND pswd='$password'"
27             val rs = db.rawQuery(query, selectionArgs: null)
28             if (rs.moveToFirst()) {
29                 val name = rs.getString(rs.getColumnIndex("name"))
30                 rs.close()
```

```
signupForm.kt
29
30         val name = rs.getString(rs.getColumnIndex("name"))
31         rs.close()
32         startActivity(Intent(packageContext: this, MainActivity::class.java).putExtra(name: "name", name))
33     } else {
34         val ad = AlertDialog.Builder(context: this)
35         ad.setTitle("Message")
36         ad.setMessage("Username or password is incorrect!")
37         ad.setPositiveButton(text: "Ok", listener: null)
38         ad.show()
39     }
40
41     bind.regisLink.setOnClickListener { it: View! }
42         val intent = Intent(packageContext: this, signupForm::class.java)
43         startActivity(intent)
44     }
45
46     // Toggle password visibility
47     bind.passwordVisibilityButton.setOnClickListener { it: View! }
48         isPasswordVisible = !isPasswordVisible
49         if (isPasswordVisible) {
50             bind.ed3.inputType = 129 // Show Password
51         } else {
52             bind.ed3.inputType = 128 // Hide Password
53         }
54         bind.ed3.setSelection(bind.ed3.text.length)
55     }
56
57 }
```

CODE 14: signupForm.kt

```
1 package com.aarush.newsapp
2
3 import android.app.AlertDialog;
4 import android.content.ContentValues;
5 import android.content.Intent;
6 import android.os.Bundle;
7 import android.widget.Toast;
8 import android.os.Handler;
9 import androidx.appcompat.app.AppCompatActivity;
10 import com.aarush.newsapp.databinding.ActivitySignupFormBinding;
11
12 class signupForm : AppCompatActivity() {
13     private lateinit var binding: ActivitySignupFormBinding
14
15     override fun onCreate(savedInstanceState: Bundle?) {
16         super.onCreate(savedInstanceState)
17         binding = ActivitySignupFormBinding.inflate(layoutInflater)
18         setContentView(binding.root)
19         var dbhelp = DB(context)
20         var db = dbhelp.writableDatabase
21         binding.btnrgs.setOnClickListener { it: View -
22             var name = binding.ed1.text.toString()
23             var username = binding.ed2.text.toString()
24             var password = binding.ed3.text.toString()
25             if (name.isNotEmpty() && username.isNotEmpty() && password.isNotEmpty()) {
26                 var data = ContentValues()
27                 data.put("name", binding.ed1.text.toString())
28                 data.put("username", binding.ed2.text.toString())
29                 data.put("pswd", binding.ed3.text.toString())
30                 var rs: Long = db.insert( table: "user", nullColumnHack: null, data)
31                 if (!rs.equals(-1)) {
32                     var ad = AlertDialog.Builder( context: this)
33                     ad.setTitle("Message")
34                     ad.setMessage("Account registered successfully")
35                     ad.setPositiveButton( text: "Ok", listener: null)
36                     ad.show()
37                     binding.ed1.text.clear()
38                     binding.ed2.text.clear()
39                     binding.ed3.text.clear()
40                     val handler = Handler()
41                     handler.postDelayed({
42                         // Start the login_form activity after a 1.5-second delay
43                         val intent = Intent( packageName: this, login_form::class.java)
44                         startActivity(intent)
45                     }, delayMillis: 1500)
46                 } else {
47                     var ad = AlertDialog.Builder( context: this)
48                     ad.setTitle("Message")
49                     ad.setMessage("Record not added")
50                     ad.setPositiveButton( text: "Ok", listener: null)
51                     ad.show()
52                     binding.ed1.text.clear()
53                     binding.ed2.text.clear()
54                     binding.ed3.text.clear()
55                 }
56             } else {
57                 Toast.makeText( context: this, text: "All fields required", Toast.LENGTH_SHORT).show()
58             }
59         }
60         binding.loginLink.setOnClickListener { it: View -
61             val intent = Intent( packageName: this, login_form::class.java)
62         }
63     }
64 }
```

```
64 }
```

The screenshot shows the Android Studio code editor for a file named `signupForm.kt`. The code is written in Kotlin and handles user input validation and navigation between activities.

```
38     binding.ed3.text.clear()
39
40     val handler = Handler()
41     handler.postDelayed({
42         // Start the login_form activity after a 1.5-second delay
43         val intent = Intent(packageContext: this, login_form::class.java)
44         startActivity(intent)
45     }, delayMillis: 1500)
46 } else {
47     var ad = AlertDialog.Builder(context: this)
48     ad.setTitle("Message")
49     ad.setMessage("Record not added")
50     ad.setPositiveButton(text: "Ok", listener: null)
51     ad.show()
52     binding.ed1.text.clear()
53     binding.ed2.text.clear()
54     binding.ed3.text.clear()
55 } else {
56     Toast.makeText(context: this, text: "All fields required", Toast.LENGTH_SHORT).show()
57 }
58
59 binding.loginLink.setOnClickListener { it: View!
60     val intent = Intent(packageContext: this, login_form::class.java)
61     startActivity(intent)
62 }
63
64 }
```

CODE 15: CustomSearchBar.kt

```
CustomSearchBar.kt
```

```
1 package com.aarush.newsapp.widget
2
3 import android.content.Context
4 import android.util.AttributeSet
5 import android.view.View
6 import android.widget.ImageView
7 import android.widget.LinearLayout
8 import androidx.appcompat.widget.SearchView
9 import com.bumptech.glide.Glide
10 import com.aarush.newsapp.R
11
12 class CustomSearchBar : LinearLayout {
13
14     private lateinit var searchView: SearchView
15     private lateinit var imageView: ImageView
16
17     constructor(context: Context, attrs: AttributeSet?) : super(context, attrs) {
18         initView(context, attrs)
19     }
20
21     constructor(context: Context, attrs: AttributeSet?, defStyleAttr: Int) :
22         super(context, attrs, defStyleAttr) {
23         initView(context, attrs)
24     }
25
26     private fun initView(context: Context, attrs: AttributeSet?) {
27         inflate(context, R.layout.searchview_custom, this)
28
29         searchView = findViewById(R.id.searchBar)
```

```
CustomSearchBar.kt
```

```
31     val attributes = context.obtainStyledAttributes(attrs, R.styleable.CustomButton)
32
33     searchView.apply { this: SearchView
34         queryHint = attributes.getString(R.styleable.CustomButton_queryHint)
35         queryHint = attributes.getString(R.styleable.CustomButton_android_queryHint)
36     }
37     removeClearTextAnimation()
38
39     val isUseAdditionalButton = attributes.getBoolean(R.styleable.CustomButton_useAdditionalButton, defaultValue: false)
40     val btnSrc = attributes.getDrawable(R.styleable.CustomButton_additionalButtonSrc)
41     if (isUseAdditionalButton) {
42         imageView.visibility = View.VISIBLE
43         setAdditionalButtonImage(btnSrc)
44     } else {
45         imageView.visibility = View.GONE
46     }
47
48     attributes.recycle()
49 }
50
51 private fun removeClearTextAnimation() {
52     val closeBtn = searchView.findViewById<ImageView>(androidx.appcompat.R.id.search_close_btn)
53     closeBtn.background = null
54 }
55
56 private fun removeSearchIcon() {
57     val icSearch = searchView.findViewById<ImageView>(androidx.appcompat.R.id.search_mag_icon)
58     icSearch.setLayoutParams(LayoutParams(0, 0))
```

```
CustomSearchBar.kt
58     icSearch.layoutParams = LayoutParams(width: 0, height: 0)
59 }
60
61     fun setOnSearchViewClickListener(listener: () -> Unit) {
62         searchView.setOnQueryTextFocusChangeListener { v, hasFocus ->
63             if (hasFocus) {
64                 listener()
65                 v.clearFocus()
66             }
67         }
68         (this as LinearLayout).setOnClickListener { listener() }
69     }
70
71     fun setOnQueryChangeListener(listener: SearchView.OnQueryTextListener) {
72         searchView.setOnQueryTextListener(listener)
73     }
74
75     fun setOnAdditionalButtonListener(onClick: () -> Unit) {
76         imageView.setOnClickListener { onClick() }
77     }
78
79     fun setQuery(query: String){
80         searchView.setQuery(query, submit: false)
81     }
82
83     fun setQueryHint(hint: String) {
84         searchView.queryHint = hint
85     }
```

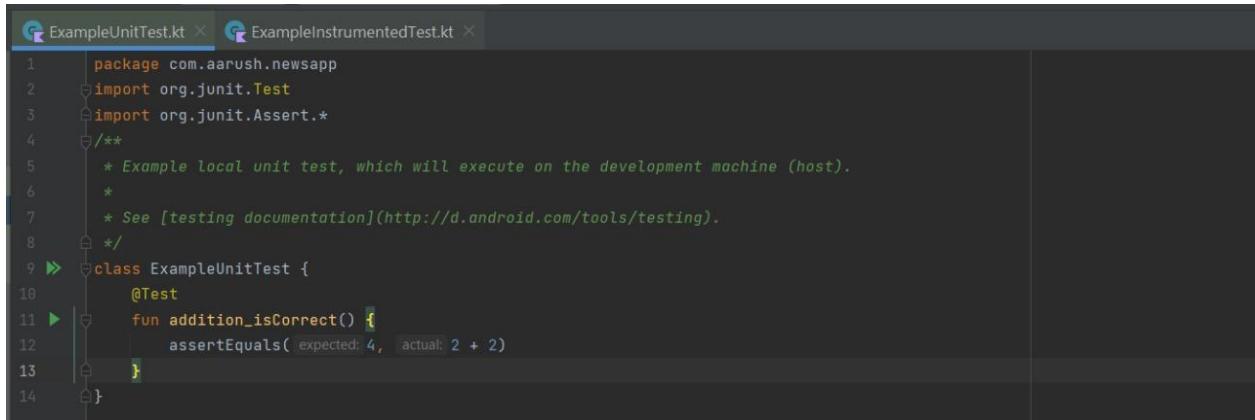
```
CustomSearchBar.kt
85     }
86
87     fun getQuery(): String {
88         return searchView.query.toString()
89     }
90
91     fun isUseSearchIcon(isUse: Boolean) {
92         if (!isUse) {
93             removeSearchIcon()
94         }
95     }
96
97     fun isUsingAdditionalButton(isUsing: Boolean) {
98         imageView.visibility = if (isUsing) {
99             View.VISIBLE
100        } else {
101            View.GONE
102        }
103    }
104
105    fun setAdditionalButtonImage(source: Any?) {
106        Glide.with(context) RequestManager
107            .load(source) RequestBuilder<Drawable>
108            .into(imageView)
109    }
110 }
```

CODE 16: CustomTabLayout.kt

```
1 package com.aarush.newsapp.widget
2
3 import android.content.Context
4 import android.util.AttributeSet
5 import android.view.ViewGroup
6 import androidx.viewpager2.widget.ViewPager2
7 import com.google.android.material.tabs.TabLayout
8 import com.google.android.material.tabs.TabLayoutMediator
9
10 class CustomTabLayout : TabLayout {
11     constructor(context: Context) : super(context)
12     constructor(context: Context, attrs: AttributeSet?) : super(context, attrs)
13     constructor(context: Context, attrs: AttributeSet?, defStyleAttr: Int) :
14         super(context, attrs, defStyleAttr)
15
16     fun addTitleOnlyTabs(titles: List<String>) {
17         titles.forEach { title ->
18             addTab(newTab().setText(title.replaceFirstChar { Char::uppercase }))
19         }
20     }
21
22     fun setTabsMargin(left: Int, top: Int, right: Int, bottom: Int) {
23         for (i in 0 .. until tabCount) {
24             val tab = getChildAt( index = 0 ) as ViewGroup.getChildAt(i)
25             val params = tab.layoutParams as ViewGroup.MarginLayoutParams
26             params.setMargins(left, top, right, bottom)
27             tab.requestLayout()
28         }
29     }
30 }
```

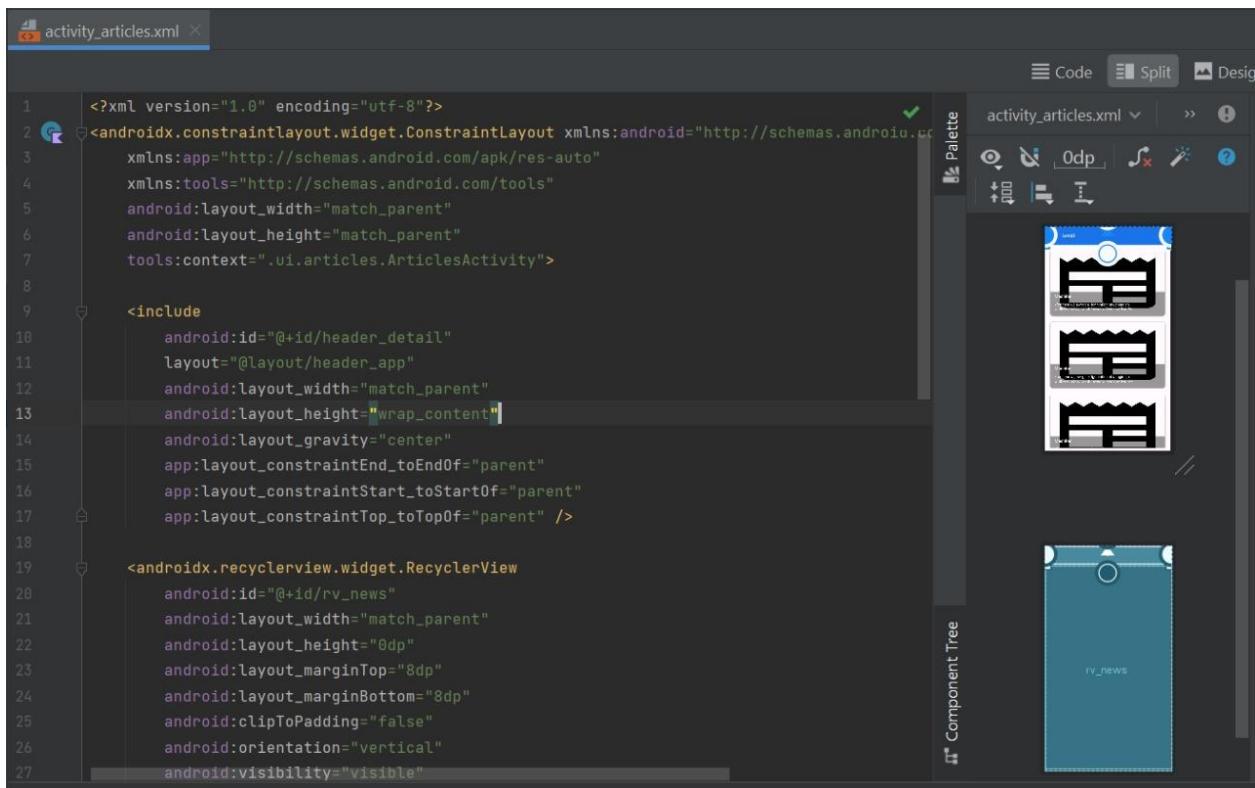
```
16     fun addTitleOnlyTabs(titles: List<String>) {
17         titles.forEach { title ->
18             addTab(newTab().setText(title.replaceFirstChar { Char::uppercase }))
19         }
20     }
21
22     fun setTabsMargin(left: Int, top: Int, right: Int, bottom: Int) {
23         for (i in 0 .. until tabCount) {
24             val tab = getChildAt( index = 0 ) as ViewGroup.getChildAt(i)
25             val params = tab.layoutParams as ViewGroup.MarginLayoutParams
26             params.setMargins(left, top, right, bottom)
27             tab.requestLayout()
28         }
29     }
30
31     fun setIndicatorForViewPager(pager: ViewPager2) {
32         TabLayoutMediator( tabLayout: this, pager) { tab, _ ->
33             with(tab.view) { this.TabView
34                 .isFocusable = false
35                 .isClickable = false
36                 .isEnabled = false
37             }
38             tab.select()
39         }.attach()
40     }
41 }
```

CODE 17: ExampleUnitTest.kt

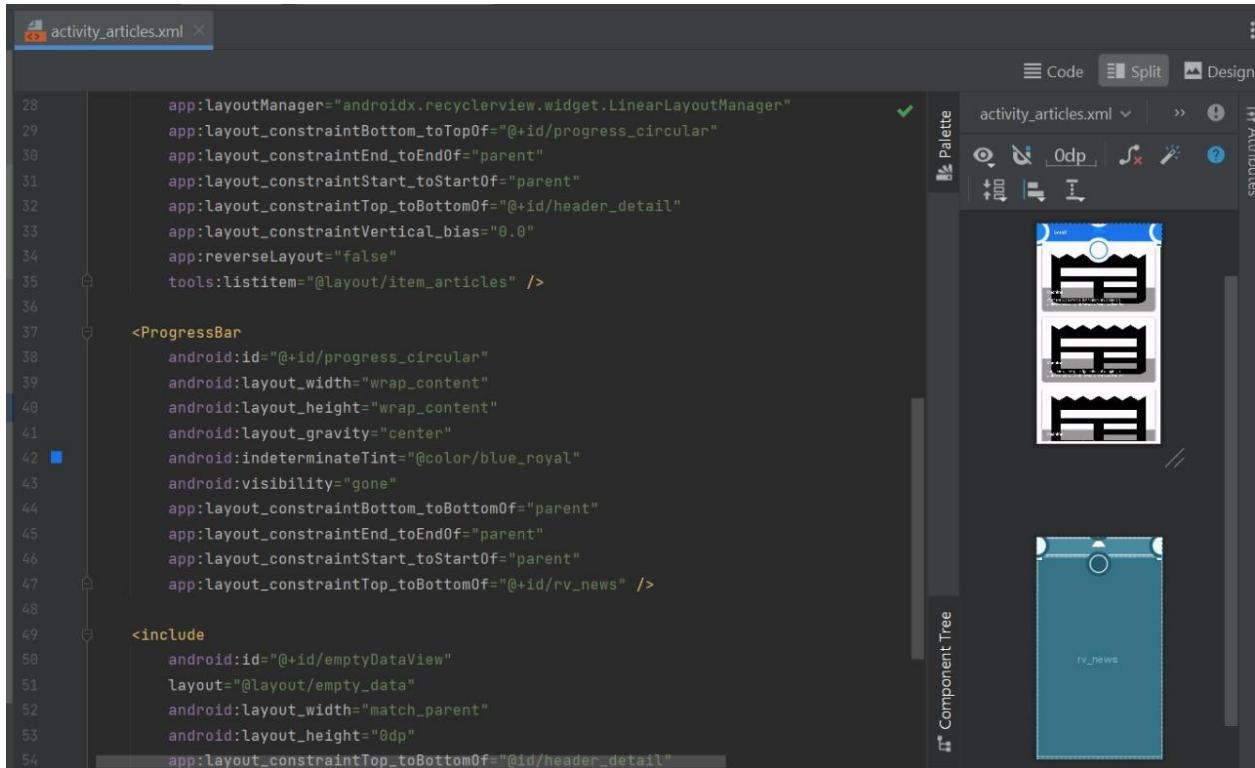


```
1 package com.aarush.newsapp
2 import org.junit.Test
3 import org.junit.Assert.*
4 /**
5  * Example local unit test, which will execute on the development machine (host).
6  *
7  * See [testing documentation](http://d.android.com/tools/testing).
8 */
9 class ExampleUnitTest {
10     @Test
11     fun addition_isCorrect() {
12         assertEquals(4, 2 + 2)
13     }
14 }
```

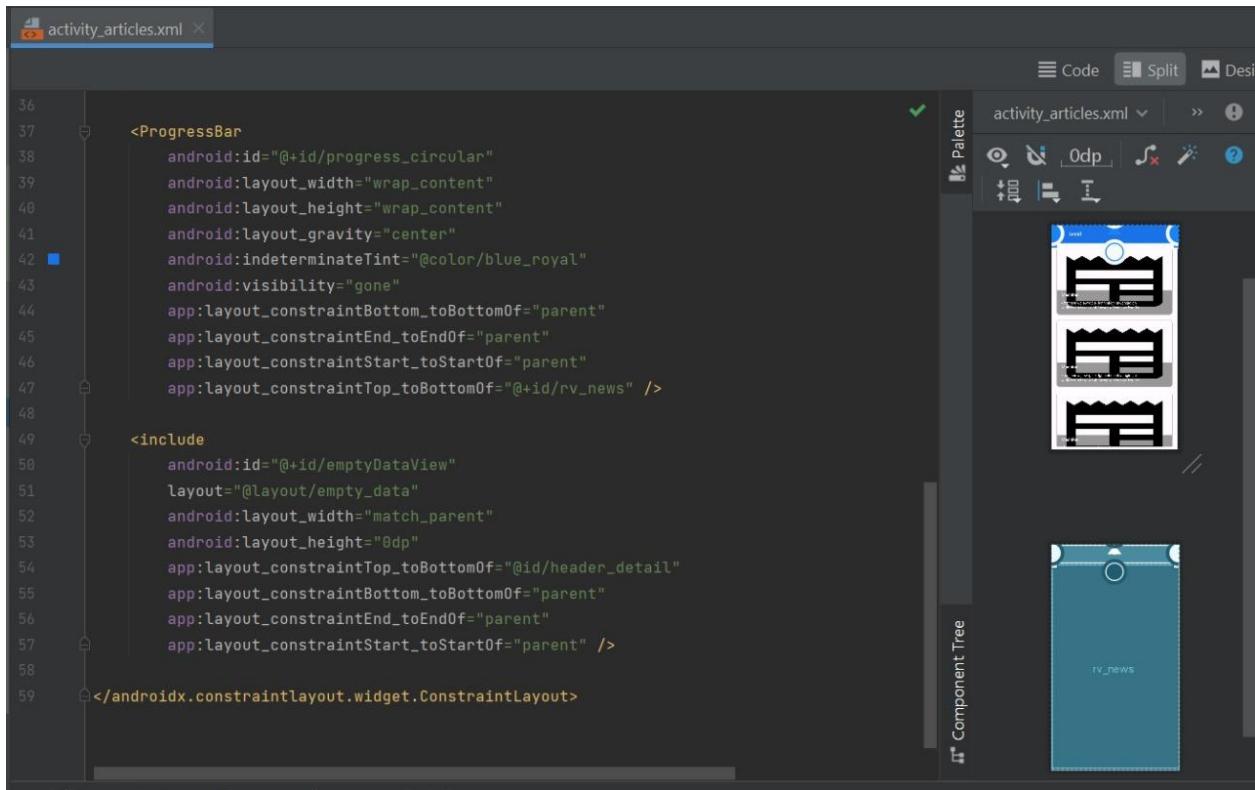
CODE 18: activity_articles.xml



```
1 <?xml version="1.0" encoding="utf-8"?>
2 <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3     xmlns:app="http://schemas.android.com/apk/res-auto"
4     xmlns:tools="http://schemas.android.com/tools"
5     android:layout_width="match_parent"
6     android:layout_height="match_parent"
7     tools:context=".ui.articles.ArticlesActivity">
8
9     <include
10         android:id="@+id/header_detail"
11         layout="@layout/header_app"
12         android:layout_width="match_parent"
13         android:layout_height="wrap_content"
14         android:layout_gravity="center"
15         app:layout_constraintEnd_toEndOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <androidx.recyclerview.widget.RecyclerView
20         android:id="@+id/rv_news"
21         android:layout_width="match_parent"
22         android:layout_height="0dp"
23         android:layout_marginTop="8dp"
24         android:layout_marginBottom="8dp"
25         android:clipToPadding="false"
26         android:orientation="vertical"
27         android:visibility="visible" />
```



```
activity_articles.xml
28     app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
29     app:layout_constraintBottom_toTopOf="@+id/progress_circular"
30     app:layout_constraintEnd_toEndOf="parent"
31     app:layout_constraintStart_toStartOf="parent"
32     app:layout_constraintTop_toBottomOf="@+id/header_detail"
33     app:layout_constraintVertical_bias="0.0"
34     app:reverseLayout="false"
35     tools:listItem="@layout/item_articles" />
36
37 <ProgressBar
38     android:id="@+id/progress_circular"
39     android:layout_width="wrap_content"
40     android:layout_height="wrap_content"
41     android:layout_gravity="center"
42     android:indeterminateTint="@color/blue_royal"
43     android:visibility="gone"
44     app:layout_constraintBottom_toBottomOf="parent"
45     app:layout_constraintEnd_toEndOf="parent"
46     app:layout_constraintStart_toStartOf="parent"
47     app:layout_constraintTop_toBottomOf="@+id/rv_news" />
48
49 <include
50     android:id="@+id/emptyDataView"
51     layout="@layout/empty_data"
52     android:layout_width="match_parent"
53     android:layout_height="0dp"
54     app:layout_constraintTop_toBottomOf="@+id/header_detail"
```



```
activity_detail_news.xml
36
37 <ProgressBar
38     android:id="@+id/progress_circular"
39     android:layout_width="wrap_content"
40     android:layout_height="wrap_content"
41     android:layout_gravity="center"
42     android:indeterminateTint="@color/blue_royal"
43     android:visibility="gone"
44     app:layout_constraintBottom_toBottomOf="parent"
45     app:layout_constraintEnd_toEndOf="parent"
46     app:layout_constraintStart_toStartOf="parent"
47     app:layout_constraintTop_toBottomOf="@+id/rv_news" />
48
49 <include
50     android:id="@+id/emptyDataView"
51     layout="@layout/empty_data"
52     android:layout_width="match_parent"
53     android:layout_height="0dp"
54     app:layout_constraintTop_toBottomOf="@+id/header_detail"
55     app:layout_constraintBottom_toBottomOf="parent"
56     app:layout_constraintEnd_toEndOf="parent"
57     app:layout_constraintStart_toStartOf="parent" />
58
59 </androidx.constraintlayout.widget.ConstraintLayout>
```

CODE 19: *activity_detail_news.xml*

The screenshot shows the Android Studio interface with the XML code for `activity_detail_news.xml`. The code defines a `ConstraintLayout` with a header detail and a `WebView`, and includes a `ProgressBar`.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.detail.DetailNewsActivity">

    <include
        android:id="@+id/header_detail"
        layout="@layout/header_app"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/header_detail" />

    <ProgressBar
        android:id="@+id/progress_circular"
        android:layout_gravity="center"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent" />

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

The screenshot shows the Android Studio interface with the XML code for `activity_login_form.xml`. The code defines a `ConstraintLayout` with a `WebView` and a `ProgressBar`.

```
        android:layout_gravity="center"
        android:layout_constraintEnd_toEndOf="parent"
        android:layout_constraintStart_toStartOf="parent"
        android:layout_constraintTop_toTopOf="parent" />

    <WebView
        android:id="@+id/webview"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/header_detail" />

    <ProgressBar
        android:id="@+id/progress_circular"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:indeterminateTint="@color/blue_royal"
        android:visibility="visible"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/header_detail" />

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

CODE 20: `activity_login_form.xml`

The screenshot shows the Android Studio interface with the XML code editor and design preview side-by-side.

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    tools:context=".login_form">
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/main_header"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
<ImageView
    android:id="@+id/image_view2"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:src="@drawable/shape"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintBaseline_toTopOf="parent"/>
<TextView
    android:id="@+id/text1"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="LOGIN"
    android:layout_marginTop="180dp"/>
```

Design View:

The design view shows a login screen with the following components:

- An ImageView at the top with a rounded shape.
- A TextView below it with the text "LOGIN".
- Two EditText fields: "Enter username" and "Enter password".
- A "Forgot Password?" button.
- A large teal "Login" button.
- A "Don't have account? Sign Up" link at the bottom.

This screenshot shows the same Android Studio interface, but the XML code editor has been scrolled down to focus on the two EditText fields.

XML Code:

```
        android:textColor="@color/black"
        android:textStyle="bold"
        android:textSize="30sp"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.497"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
    />
<EditText
    android:id="@+id/logtxt"
    android:layout_width="320dp"
    android:layout_height="50dp"
    android:layout_marginTop="35dp"
    android:background="@drawable/input_style"
    android:hint="Enter username"
    android:paddingLeft="20dp"
    android:textColorHint="@color/black"
    android:textColor="@color/black"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text1"
    />
<EditText
    android:id="@+id/ed3"
    android:layout_width="320dp"
    android:layout_height="50dp"
    android:layout_marginTop="35dp"
```

Design View:

The design view remains the same as in the first screenshot, showing the login screen with its components.

activity_login_form.xml

```
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
```

Code Split Design

Palette Attributes

Component Tree

```
    android:background="@drawable/input_style"
    android:hint="Enter password"
    android:inputType="textPassword"
    android:textColorHint="@color/black"
    android:textColor="@color/black"
    android:drawableRight="@drawable/ic_baseline_visibility_24"
    android:paddingRight="20dp"
    android:paddingLeft="20dp"
    android:longClickable="false"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/logtxt"
    />
<Button
    android:id="@+id/passwordVisibilityButton"
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    android:text="Show Password"
    android:textSize="14sp"
    android:textColor="@color/teal_200"
    app:layout_constraintTop_toBottomOf="@+id/ed3"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    android:layout_marginTop="10dp"
    />
<Button
```

activity_login_form.xml

```
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
```

Code Split Design

Palette Attributes

Component Tree

```
    android:id="@+id/btnlogin"
    android:layout_width="320dp"
    android:layout_height="60dp"
    android:layout_marginTop="64dp"
    android:backgroundTint="@color/teal_200"
    android:gravity="center"
    android:text="Login"
    android:textSize="28dp"
    android:textStyle="bold"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.505"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ed3" />
<LinearLayout
    android:id="@+id/lin2"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:layout_marginTop="20dp"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/btnlogin"
    android:gravity="center"
    android:orientation="horizontal"
    >
```

The screenshot shows the Android Studio interface with the XML code for `activity_login_form.xml` on the left and the design preview on the right. The code includes nested `LinearLayout` and `ConstraintLayout` elements, with various `TextView`, `EditText`, and `Button` components defined. The design preview shows a light blue header with rounded corners, followed by a white content area containing an `ImageView`, two `EditText` fields, a `Button` labeled "Login", and a link at the bottom.

```
105     android:orientation="horizontal"
106     >
107
108     <TextView
109         android:id="@+id/txt1"
110         android:layout_width="wrap_content"
111         android:layout_height="wrap_content"
112         android:text="Not Registered?"
113         android:textColor="@color/teal_200"
114         android:textSize="15dp"
115         tools:ignore="TextSizeCheck" />
116     <TextView
117         android:id="@+id/regis_link"
118         android:layout_width="wrap_content"
119         android:layout_height="wrap_content"
120         android:text="Sign Up"
121         android:textStyle="bold"
122         android:textColor="@color/teal_200"
123         android:textSize="16dp" />
124     </LinearLayout>
125
126
127     </androidx.constraintlayout.widget.ConstraintLayout>
128
129 </ScrollView>
```

CODE 21: `activity_main.xml`

The screenshot shows the Android Studio interface with the XML code for `activity_main.xml` on the left and the design preview on the right. The code defines a `ConstraintLayout` with a `CustomSearchBar` and a `CustomTabLayout`. The design preview shows a top navigation bar with tabs and a main content area containing a list view labeled `rv_news`.

```
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="match_parent"
7      tools:context=".ui.main.MainActivity">
8
9      <com.aarush.newsapp.widget.CustomSearchBar
10         android:id="@+id/svNews"
11         android:layout_width="match_parent"
12         android:layout_height="wrap_content"
13         android:layout_gravity="center"
14         android:layout_margin="16dp"
15         android:background="@drawable/bg_searchbar"
16         android:queryHint="Find Sources and Articles"
17         app:layout_constraintEnd_toEndOf="parent"
18         app:layout_constraintStart_toStartOf="parent"
19         app:layout_constraintTop_toTopOf="parent"
20         app:additionalButtonSrc="@drawable/ic_filter"
21         app:useAdditionalButton="true" />
22
23      <com.aarush.newsapp.widget.CustomTabLayout
24          android:id="@+id/tab_layout_category"
25          style="@style/Theme.NewsApp.ChooserTabLayout"
26          android:layout_width="0dp"
27          android:layout_height="wrap_content" />
```

The screenshot shows the Android Studio interface with the XML code for `activity_main.xml` on the left and the design view on the right.

XML Code:

```
28     android:layout_marginHorizontal="8dp"           ⓘ 2 ⚠ 1 1 ^ v
29     app:layout_constraintEnd_toEndOf="parent"
30     app:layout_constraintStart_toStartOf="parent"
31     app:layout_constraintTop_toBottomOf="@+id/svNews" />
32
33     <androidx.recyclerview.widget.RecyclerView
34         android:id="@+id/rv_news"
35         android:layout_width="match_parent"
36         android:layout_height="0dp"
37         android:layout_marginTop="8dp"
38         android:layout_marginBottom="8dp"
39         android:clipToPadding="false"
40         android:orientation="vertical"
41         android:visibility="visible"
42         app:layoutManager="androidx.recyclerview.widget.LinearLayoutManager"
43         app:layout_constraintBottom_toTopOf="@+id/progress_circular"
44         app:layout_constraintEnd_toEndOf="parent"
45         app:layout_constraintStart_toStartOf="parent"
46         app:layout_constraintTop_toBottomOf="@+id/tab_layout_category"
47         app:layout_constraintVertical_bias="0.0"
48         app:reverseLayout="false"
49         tools:listItem="@layout/item_source" />
50
51     <ProgressBar
52         android:id="@+id/progress_circular"
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
```

Design View:

The design view shows a RecyclerView component with the ID `rv_news`. The RecyclerView is positioned above a ProgressBar component with the ID `progress_circular`.

The screenshot shows the Android Studio interface with the XML code for `activity_main.xml` on the left and the design view on the right.

XML Code:

```
49     tools:listItem="@layout/item_source" />
50
51     <ProgressBar
52         android:id="@+id/progress_circular"
53         android:layout_width="wrap_content"
54         android:layout_height="wrap_content"
55         android:layout_gravity="center"
56         android:indeterminateTint="@color/blue_royal"
57         android:visibility="gone"
58         app:layout_constraintBottom_toBottomOf="parent"
59         app:layout_constraintEnd_toEndOf="parent"
60         app:layout_constraintStart_toStartOf="parent"
61         app:layout_constraintTop_toBottomOf="@+id/rv_news" />
62
63     <include
64         android:id="@+id/emptyDataView"
65         layout="@layout/empty_data"
66         android:layout_width="match_parent"
67         android:layout_height="0dp"
68         app:layout_constraintTop_toBottomOf="@+id/tab_layout_category"
69         app:layout_constraintBottom_toBottomOf="parent"
70         app:layout_constraintEnd_toEndOf="parent"
71         app:layout_constraintStart_toStartOf="parent" />
72
73     <androidx.constraintlayout.widget.ConstraintLayout>
```

Design View:

The design view shows the same layout structure as the previous screenshot, but the RecyclerView component (`rv_news`) is now positioned below the ProgressBar component (`progress_circular`).

CODE 22: `activity_search.xml`

The screenshot shows the Android Studio interface with the XML code for `activity_search.xml` in the left pane and the Design tab in the right pane.

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    tools:context=".ui.search.SearchActivity">

    <ImageView
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginTop="32dp"
        android:layout_marginStart="16dp"
        android:src="@drawable/ic_close"
        app:tint="@color/blue_royal"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <com.aarush.newsapp.widget.CustomSearchBar
        android:id="@+id/svNews"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_gravity="center"
        android:layout_margin="16dp"
        android:background="@drawable/bg_searchbar"
        android:queryHint="Find Sources and Articles" />
```

Design Tab:

The Design tab displays the layout structure. It includes a `svNews` `EditText` at the top and a `rv_news` `RecyclerView` below it. A `Component Tree` panel on the left lists the views in the layout.

The screenshot shows the Android Studio interface with the XML code for `activity_search.xml` in the left pane and the Design tab in the right pane.

XML Code:

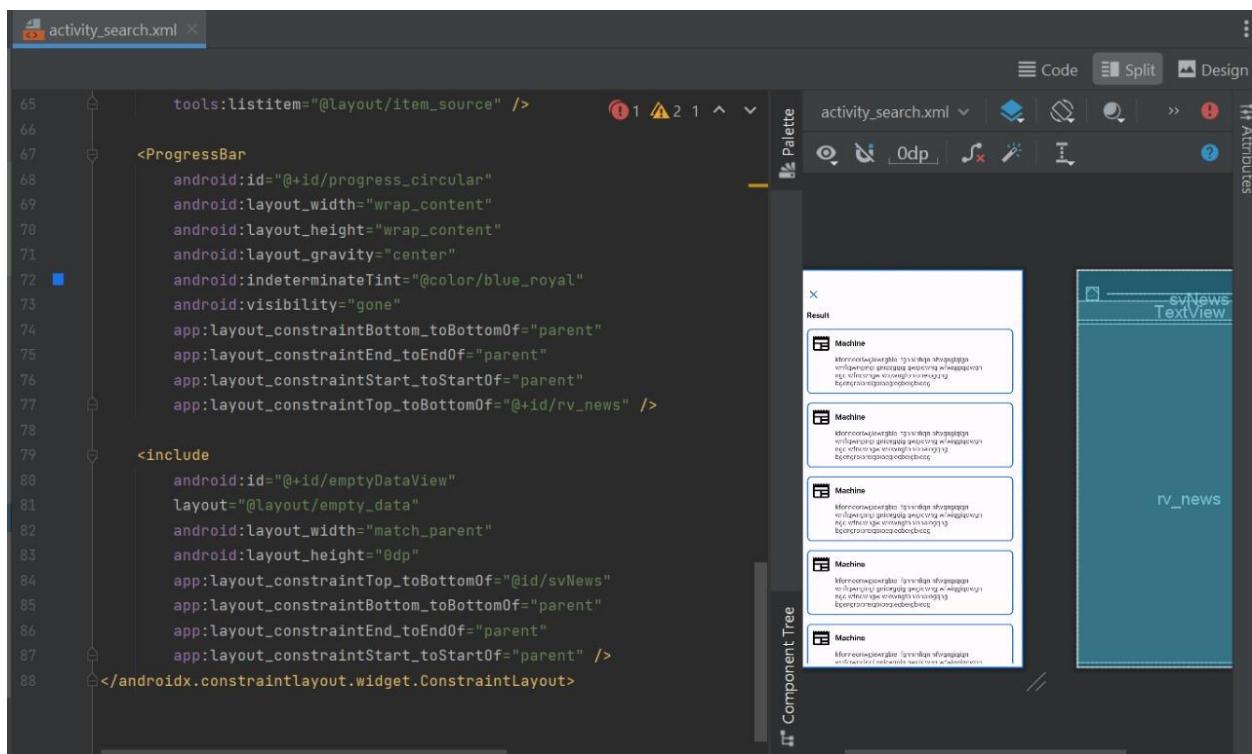
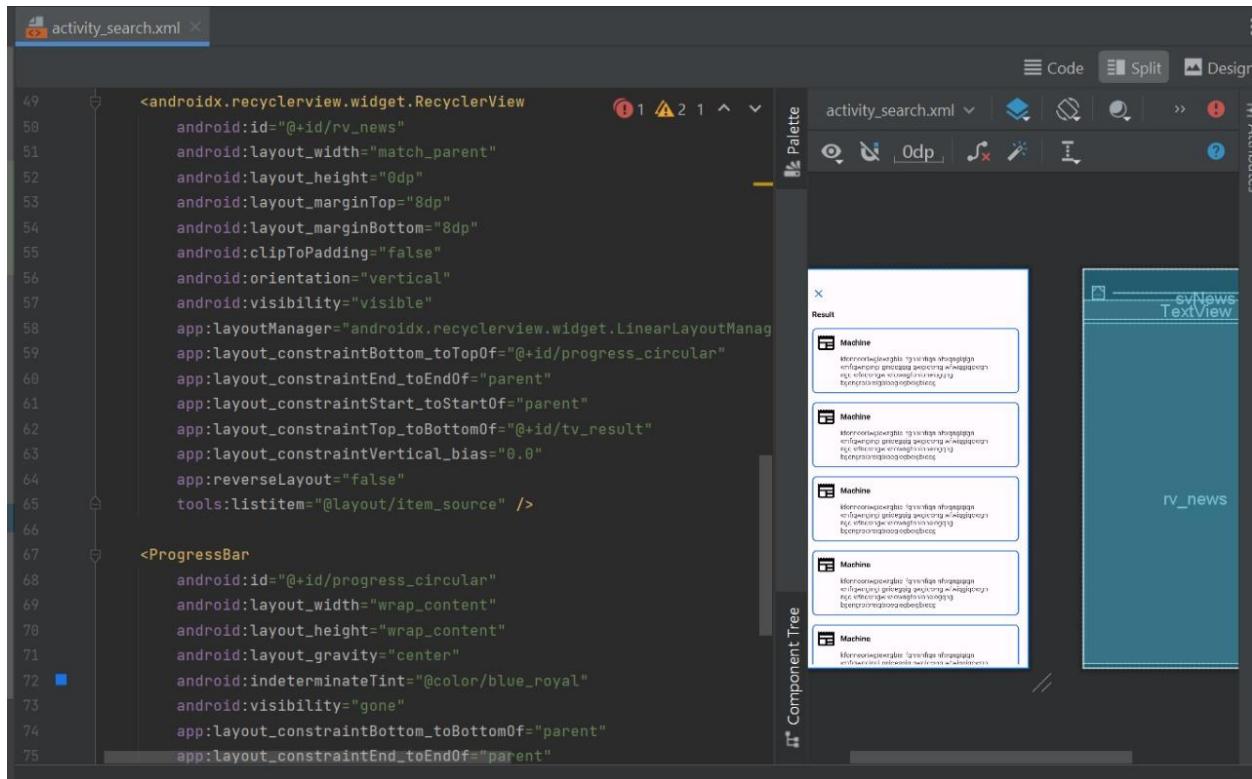
```
    app:additionalButtonSrc="@drawable/ic_filter" ① ⚠ 2 1 ⌂
    app:layout_constraintBottom_toBottomOf="@+id/btnBack"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toEndOf="@+id/btnBack"
    app:layout_constraintTop_toTopOf="@+id/btnBack"
    app:useAdditionalButton="true" />

    <TextView
        android:id="@+id/tv_result"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:paddingHorizontal="16dp"
        android:paddingTop="16dp"
        android:textColor="@color/black"
        android:textSize="14sp"
        android:textStyle="bold"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/btnBack"
        tools:text="Result" />

    <androidx.recyclerview.widget.RecyclerView
        android:id="@+id/rv_news"
        android:layout_width="match_parent"
        android:layout_height="0dp"
        android:layout_marginTop="8dp"
        android:layout_marginBottom="8dp" />
```

Design Tab:

The Design tab displays the layout structure. It includes a `svNews` `EditText` at the top, a `rv_news` `RecyclerView` below it, and a `Component Tree` panel on the left listing the views.



CODE 23: `activity_signup_form.xml`

The screenshot shows the Android Studio interface with the XML code for `activity_signup_form.xml` on the left and a design preview on the right.

XML Code:

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:background="@color/background"
    tools:context=".signupForm">
<androidx.constraintlayout.widget.ConstraintLayout
    android:id="@+id/main_header"
    android:layout_width="match_parent"
    android:layout_height="match_parent">
    <ImageView
        android:id="@+id/image_view"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/shape"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintBaseline_toTopOf="parent"
        />
    <TextView
        android:id="@+id/text1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="REGISTRATION"
        android:layout_marginTop="150dp"/>

```

Design Preview:

The design preview shows a registration screen with the following components:

- An **ImageView** at the top.
- A **Text View** with the text "REGISTRATION" centered below the image view.
- Three **EditText** fields labeled "ed1", "ed2", and "ed3".
- A **Button**.
- A **Text View** at the bottom with the text "Already Registered? Login".

The screenshot shows the Android Studio interface with the XML code for `activity_signup_form.xml` on the left and a design preview on the right.

XML Code:

```
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
<EditText
    android:id="@+id/ed1"
    android:layout_width="320dp"
    android:layout_height="50dp"
    android:layout_marginTop="35dp"
    android:background="@drawable/input_style"
    android:hint="Enter Full Name"
    android:textColorHint="@color/black"
    android:textColor="@color/black"
    android:paddingLeft="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/text1"
    />
<EditText
    android:id="@+id/ed2"
    android:layout_width="320dp"
    android:layout_height="50dp"
    android:layout_marginTop="35dp"
    android:background="@drawable/input_style"
    android:hint="Enter Username"
    android:textColorHint="@color/black"
    android:textColor="@color/black"
    android:paddingLeft="20dp"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/ed1"
    />
```

Design Preview:

The design preview shows the same registration screen as the previous screenshot, but with the first **EditText** field (`ed1`) highlighted in yellow, indicating it is the current selected element in the XML code.

The screenshot shows the Android Studio interface with the XML code editor on the left and the design preview on the right. The XML code defines a registration form with three text input fields and a register button:

```
55     android:background="@drawable/input_style"
56     android:textColor="@color/black"
57     android:hint="Enter Username"
58     android:textColorHint="@color/black"
59     android:paddingLeft="20dp"
60     app:layout_constraintEnd_toEndOf="parent"
61     app:layout_constraintStart_toStartOf="parent"
62     app:layout_constraintTop_toBottomOf="@+id/ed1"
63   />
64   <EditText
65     android:id="@+id/ed3"
66     android:layout_width="320dp"
67     android:layout_height="50dp"
68     android:layout_marginTop="35dp"
69     android:background="@drawable/input_style"
70     android:hint="Enter password"
71     android:textColorHint="@color/black"
72     android:textColor="@color/black"
73     android:inputType="textPassword"
74     android:drawableRight="@drawable/ic_baseline_visibility_24"
75     android paddingRight="20dp"
76     android paddingLeft="20dp"
77     android:longClickable="false"
78     app:layout_constraintEnd_toEndOf="parent"
79     app:layout_constraintStart_toStartOf="parent"
80     app:layout_constraintTop_toBottomOf="@+id/ed2"
81   />
```

The design preview shows a light blue background with a white card containing the registration form. The card has a title "REGISTRATION" and three text input fields labeled "Enter Full Name", "Enter Username", and "Enter password". Below the inputs is a teal-colored "Register" button. At the bottom of the card, there is a link "Already Registered? Login".

This screenshot shows the same XML code as the first one, but with a red square marker highlighting the closing tag of the first text input field at line 95. The design preview remains the same, displaying the registration form with its components and styling.

```
82   <Button
83     android:id="@+id/btnrgs"
84     android:layout_width="320dp"
85     android:layout_height="60dp"
86     android:gravity="center"
87     android:text="Register"
88     android:textSize="20dp"
89     android:textStyle="bold"
90     android:layout_marginTop="35dp"
91     android:backgroundTint="@color/teal_200"
92     app:layout_constraintEnd_toEndOf="parent"
93     app:layout_constraintStart_toStartOf="parent"
94     app:layout_constraintTop_toBottomOf="@+id/ed3"
95   />
96   <TextView
97     android:id="@+id/t1"
98     android:layout_width="wrap_content"
99     android:layout_height="wrap_content"
100    tools:ignore="MissingConstraints" />
101   <LinearLayout
102     android:id="@+id/lin1"
103     android:layout_width="match_parent"
104     android:layout_height="wrap_content"
105     android:layout_marginTop="20dp"
106     app:layout_constraintStart_toStartOf="parent"
107     app:layout_constraintEnd_toEndOf="parent"
108     app:layout_constraintTop_toBottomOf="@+id/btnrgs"
```

The screenshot shows the Android Studio interface with the code editor and design preview side-by-side. The code editor displays the XML layout for a registration screen, and the design preview shows the visual representation of the screen with various UI components like text fields and buttons.

```
activity_signup_form.xml
106     app:layout_constraintStart_toStartOf="parent"    ▲28 ^ v
107     app:layout_constraintEnd_toEndOf="parent"
108     app:layout_constraintTop_toBottomOf="@+id/btnrgs"
109     android:gravity="center"
110     android:orientation="horizontal"
111   >
112   <TextView
113     android:id="@+id/txt3"
114     android:layout_width="wrap_content"
115     android:layout_height="wrap_content"
116     android:text="Already Registered?"
117     android:textColor="@color/teal_200"
118     android:textSize="15dp"
119     tools:ignore="TextSizeCheck" />
120   <TextView
121     android:id="@+id/login_link"
122     android:layout_width="wrap_content"
123     android:layout_height="wrap_content"
124     android:text="Login"
125     android:textStyle="bold"
126     android:textColor="@color/teal_200"
127     android:textSize="16dp"
128   />
129   </LinearLayout>
130 </androidx.constraintlayout.widget.ConstraintLayout>
131 </ScrollView>
```

CODE 24: `empty_data.xml`

The screenshot shows the Android Studio interface with the code editor displaying the XML layout for a screen that handles empty data. The layout includes an ImageView and a TextView.

```
empty_data.xml
1  <?xml version="1.0" encoding="utf-8"?>
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res/android"
3    xmlns:app="http://schemas.android.com/apk/res-auto"
4    android:layout_width="match_parent"
5    android:layout_height="match_parent"
6    android:id="@+id/parent"
7    android:visibility="gone"
8    android:background="@color/white">
9
10
11  <ImageView
12    android:id="@+id/imgEmpty"
13    android:layout_width="0dp"
14    android:layout_height="0dp"
15    android:src="@drawable/ic_not_found"
16    app:layout_constraintDimensionRatio="1:1"
17    app:layout_constraintBottom_toTopOf="@+id/guideline4"
18    app:layout_constraintEnd_toEndOf="parent"
19    app:layout_constraintStart_toStartOf="parent"
20    app:layout_constraintTop_toTopOf="@+id/guideline3"
21    app:tint="@color/blue_royal" />
22
23
24  <TextView
25    android:layout_width="wrap_content"
26    android:layout_height="wrap_content"
27    android:text="Not Found"
28    android:textColor="@color/black"
```

```
empty_data.xml
38     android:layout_height="wrap_content"
39     android:orientation="vertical"
40     app:layout_constraintGuide_percent="0.60" />
41
42     <androidx.constraintlayout.widget.Guideline
43         android:id="@+id/guideline2"
44         android:layout_width="wrap_content"
45         android:layout_height="wrap_content"
46         android:orientation="vertical"
47         app:layout_constraintGuide_percent="0.4" />
48
49     <androidx.constraintlayout.widget.Guideline
50         android:id="@+id/guideline3"
51         android:layout_width="wrap_content"
52         android:layout_height="wrap_content"
53         android:orientation="horizontal"
54         app:layout_constraintGuide_percent="0.35" />
55
56     <androidx.constraintlayout.widget.Guideline
57         android:id="@+id/guideline4"
58         android:layout_width="wrap_content"
59         android:layout_height="wrap_content"
60         android:orientation="horizontal"
61         app:layout_constraintGuide_percent="0.55" />
62 </androidx.constraintlayout.widget.ConstraintLayout>
```

CODE 25: header_app.xml

```
header_app.xml
1  <?xml version="1.0" encoding="utf-8"?>           ▲ 1 ^ v
2  <androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://
3      xmlns:app="http://schemas.android.com/apk/res-auto"
4      xmlns:tools="http://schemas.android.com/tools"
5      android:layout_width="match_parent"
6      android:layout_height="wrap_content"
7      android:background="@color/blue_royal"
8      android:padding="16dp">
9
10     <ImageView
11         android:id="@+id/btnBack"
12         android:layout_width="wrap_content"
13         android:layout_height="wrap_content"
14         android:src="@drawable/ic_close"
15         app:layout_constraintBottom_toBottomOf="parent"
16         app:layout_constraintStart_toStartOf="parent"
17         app:layout_constraintTop_toTopOf="parent" />
18
19     <TextView
20         android:id="@+id/tvHeader"
21         android:layout_width="wrap_content"
22         android:layout_height="wrap_content"
23         android:textColor="@color/white"
24         android:textSize="16sp"
25         tools:text="Detail"
26         android:textStyle="bold"
27         android:layout_marginStart="16dp" />
```

The screenshot shows the Android Studio interface with the XML file `header_app.xml` open. The left pane displays the XML code, and the right pane shows the visual representation of the layout. The layout consists of an `ImageView` at the top with a close button icon and a `TextView` below it displaying the header text.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginHorizontal="16dp"
    android:background="@drawable/bg_item_rv">

    <ImageView
        android:id="@+id/btnBack"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:src="@drawable/ic_close"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        android:id="@+id/tvHeader"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:textColor="@color/white"
        android:textSize="16sp"
        tools:text="Detail"
        android:textStyle="bold"
        android:layout_marginStart="16dp"
        app:layout_constraintBottom_toBottomOf="@+id/btnBack"
        app:layout_constraintStart_toEndOf="@+id/btnBack"
        app:layout_constraintTop_toTopOf="@+id/btnBack" />

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

CODE 26: item.xml

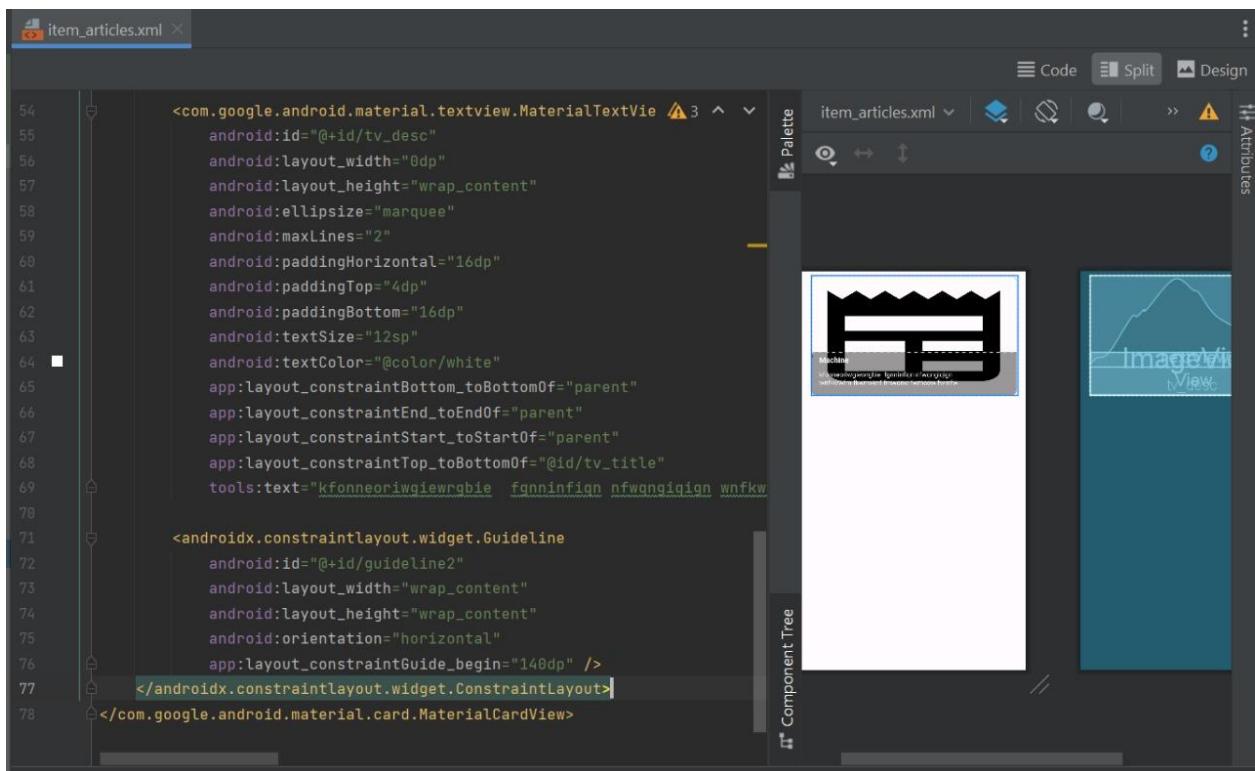
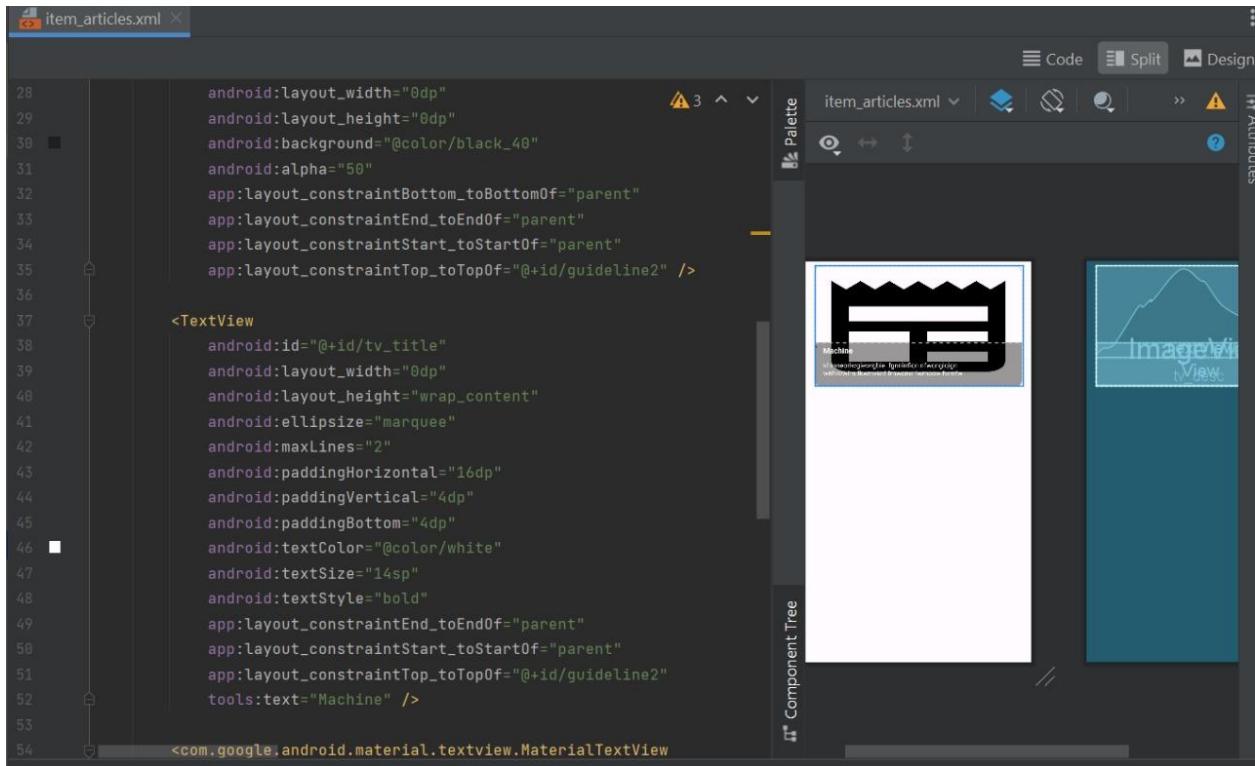
The screenshot shows the Android Studio interface with the XML file `item_source.xml` open. The left pane displays the XML code, and the right pane shows the visual representation of the layout. The layout includes an `ImageView` for a tool icon and a `TextView` for the source text.

```
<?xml version="1.0" encoding="utf-8"?>
<androidx.constraintlayout.widget.ConstraintLayout xmlns:android="http://schemas.android.com/apk/res-auto"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:padding="8dp"
    android:layout_marginTop="8dp"
    android:layout_marginBottom="8dp"
    android:layout_marginHorizontal="16dp"
    android:background="@drawable/bg_item_rv">

    <ImageView
        android:id="@+id/img_tool"
        android:layout_width="0dp"
        android:layout_height="0dp"
        app:layout_constraintDimensionRatio="1:1"
        app:layout_constraintEnd_toStartOf="@+id/guideline"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        android:src="@drawable/ic_news" />

    <TextView
        android:id="@+id/tv_source"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:ellipsize="marquee" />

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



CODE 27: searchview_custom.xml

searchview_custom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="16"
    android:paddingVertical="2dp"
    tools:parentTag="android.widget.LinearLayout">

    <androidx.appcompat.widget.SearchView
        android:id="@+id/searchBar"
        android:layout_weight="14"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false"
        android:layout_gravity="center_vertical"
        android:theme="@style/SearchViewText"
        app:iconifiedByDefault="false"
        app:queryBackground="@android:color/transparent"
        app:searchIcon="@drawable/ic_search"/>

    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/btnSearchUtil"
        android:layout_weight="2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerInside"
        android:visibility="gone"
        android:layout_marginEnd="16dp"
        android:layout_gravity="center_vertical"
        android:foreground="@android:color/transparent"
        android:background="@android:color/transparent"/>

```

searchview_custom.xml

```
<?xml version="1.0" encoding="utf-8"?>
<merge xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="wrap_content"
    android:orientation="horizontal"
    android:weightSum="16"
    android:paddingVertical="2dp"
    tools:parentTag="android.widget.LinearLayout">

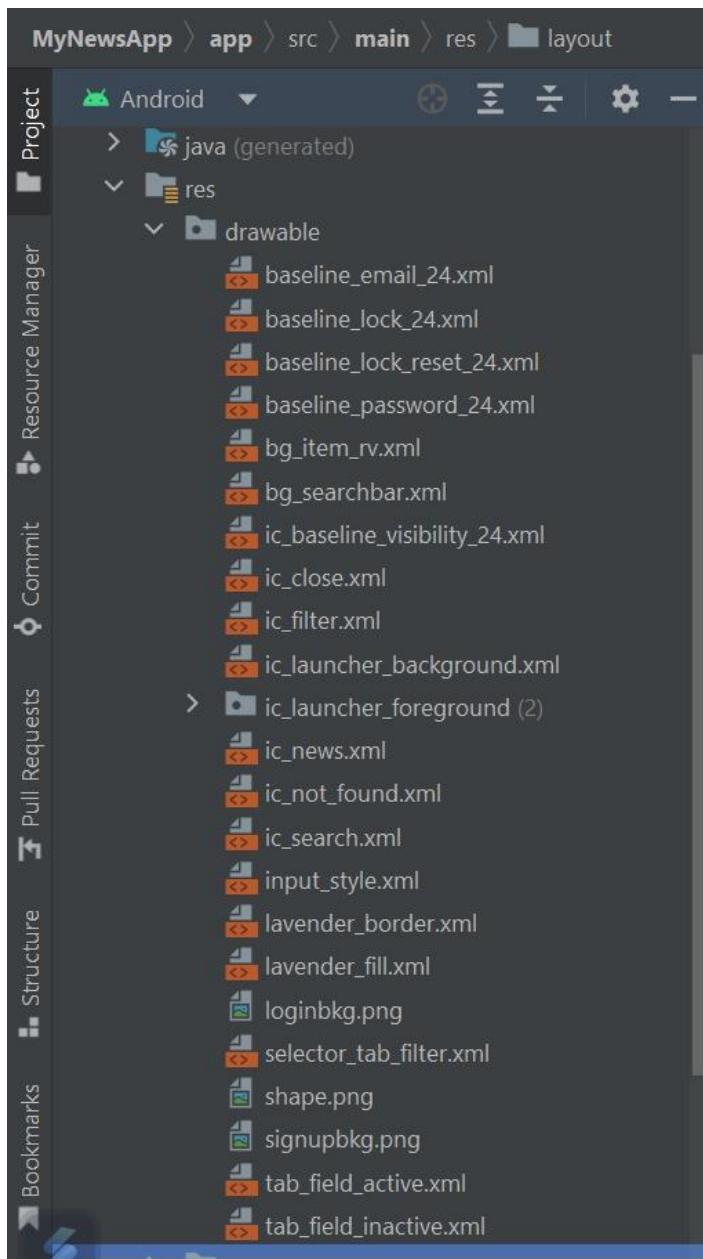
    <androidx.appcompat.widget.SearchView
        android:id="@+id/searchBar"
        android:layout_weight="14"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:iconifiedByDefault="false"
        android:layout_gravity="center_vertical"
        android:theme="@style/SearchViewText"
        app:iconifiedByDefault="false"
        app:queryBackground="@android:color/transparent"
        app:searchIcon="@drawable/ic_search"/>

    <com.google.android.material.imageview.ShapeableImageView
        android:id="@+id/btnSearchUtil"
        android:layout_weight="2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:scaleType="centerInside"
        android:visibility="gone"
        android:layout_marginEnd="16dp"
        android:layout_gravity="center_vertical"
        android:foreground="@android:color/transparent"
        android:background="@android:color/transparent"/>

```

DRAWABLE RESOURCES

Take a screenshot of an image from your web browser, and drop it into the drawable folder.
Refactor the image.

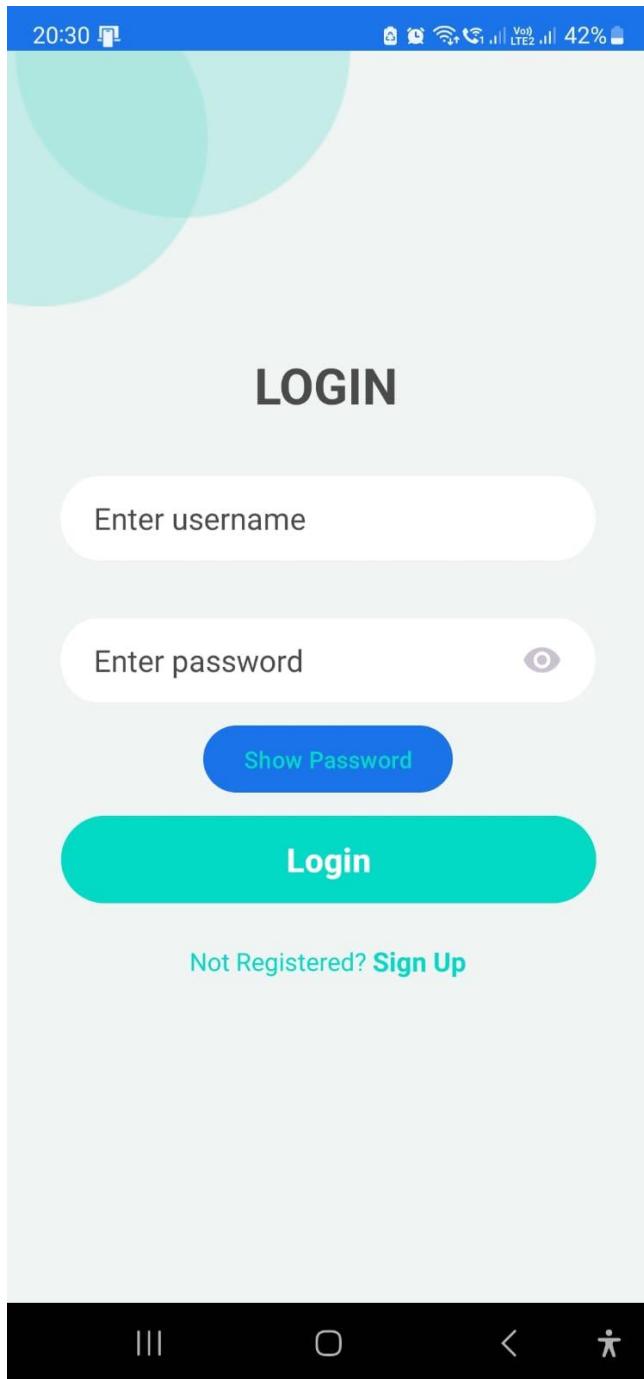


OUTPUT:

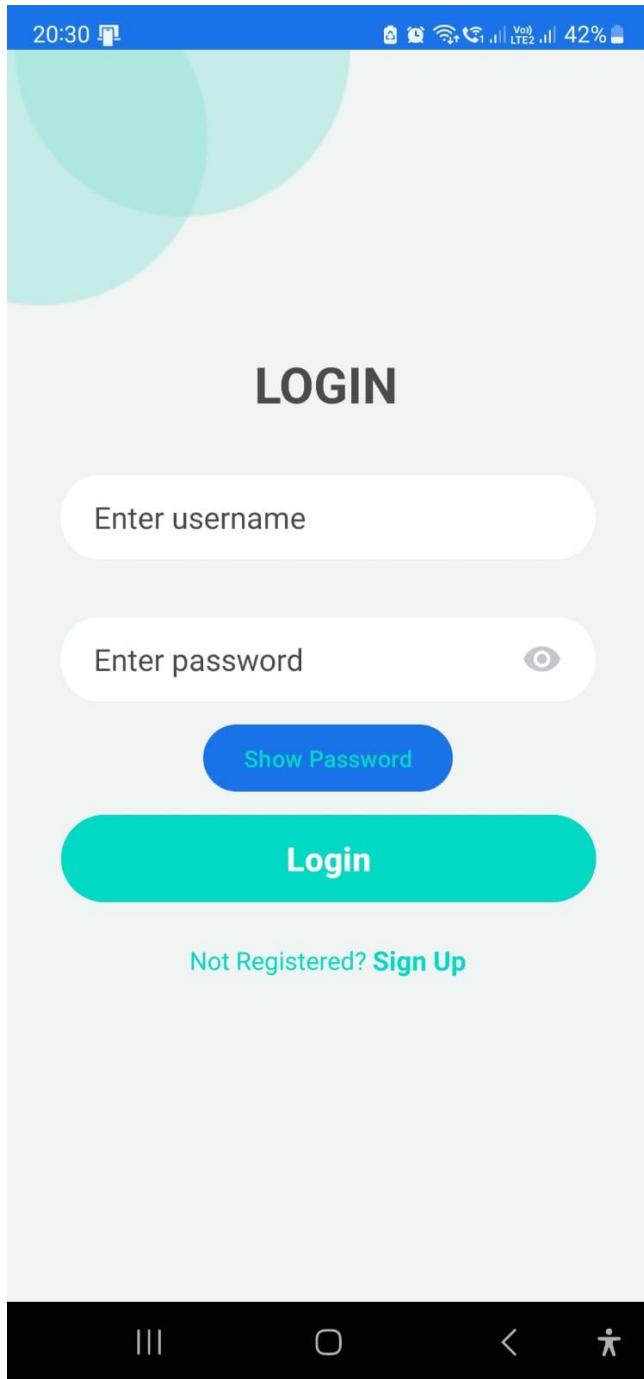
- 1. Log-in Screen**
- 2. Sign-up Screen**
- 3. Home Page**
- 4. News Page**
- 5. Specific News Page**

OUTPUT 1: Log-in Screen

LIGHT MODE:

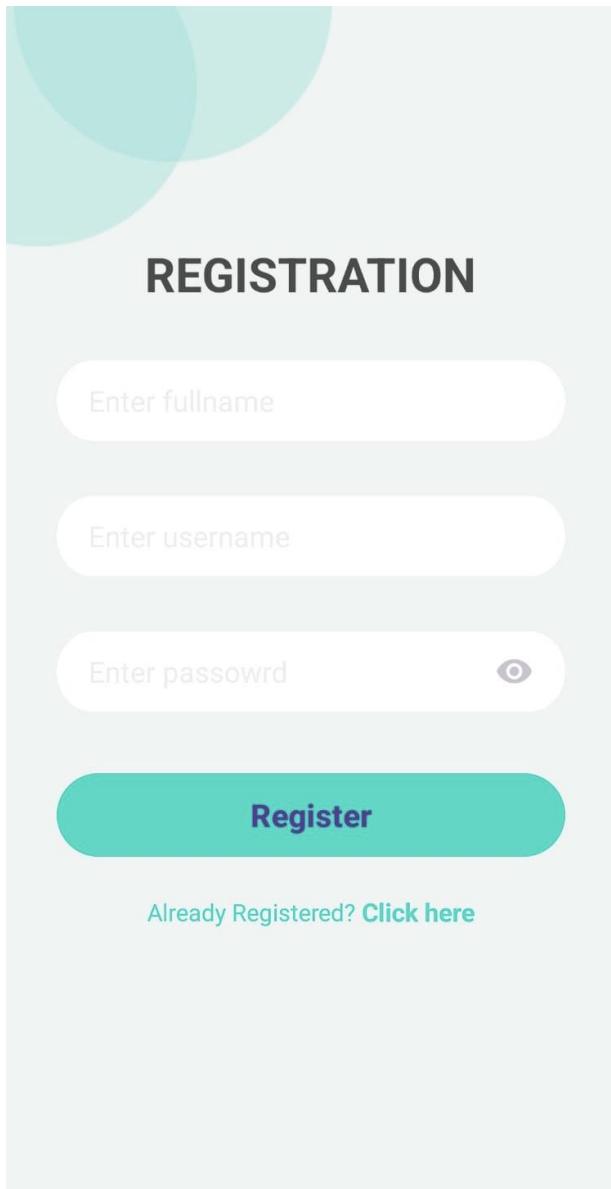


DARK MODE:

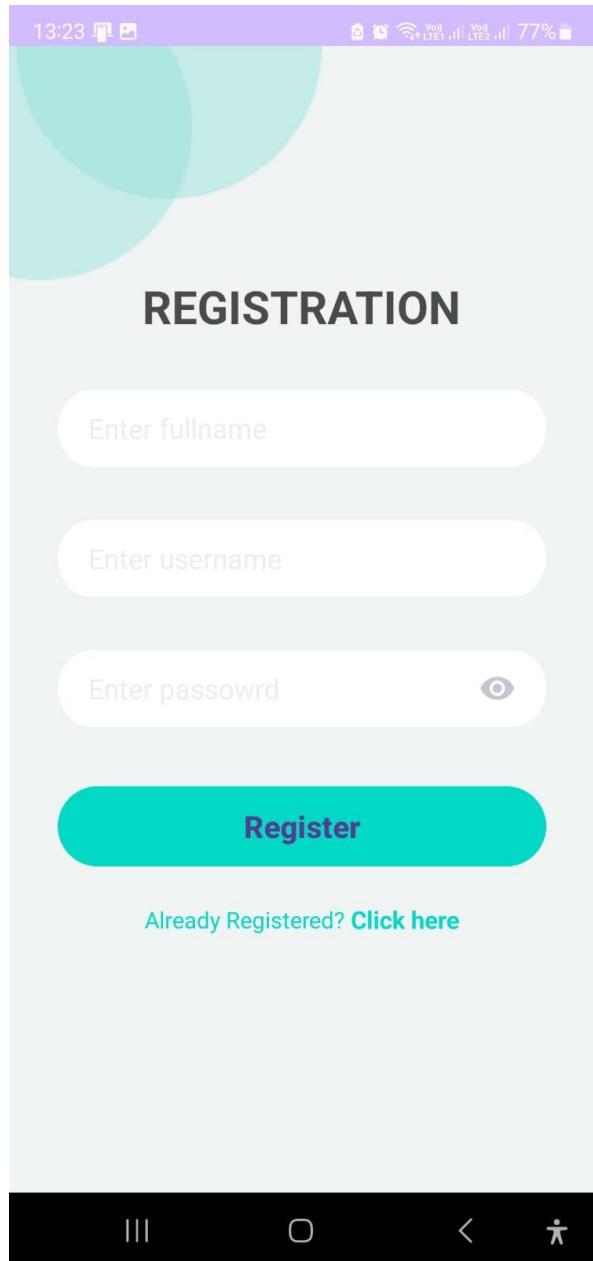


OUTPUT 2: Sign-up Screen

LIGHT MODE:



DARK MODE:



OUTPUT 3: Home Page

LIGHT MODE:

The image shows a smartphone home screen with a blue header bar at the top displaying the time (13:30), signal strength, battery level (76%), and other icons. Below the header is a search bar with the placeholder "Find Sources and Articles". Underneath the search bar are four category buttons: "Health", "Science", "Sports", and "Technology". The main content area displays five news source cards, each with a logo icon and the source's name. The first card is for "ABC News", which describes itself as "Your trusted source for breaking news, analysis, exclusive interviews, headlines, and videos at ABCNews.com.". The second card is for "ABC News (AU)", described as "Australia's most trusted source of local, national and world news. Comprehensive, independent, in-depth analysis, the latest business, sport, weather and more.". The third card is for "Aftenposten", stating it is "Norges ledende nettavis med alltid oppdaterte nyheter innenfor innenriks, utenriks, sport og kultur.". The fourth card is for "Al Jazeera English", which offers "News, analysis from the Middle East and worldwide, multimedia and interactives, opinions, documentaries, podcasts, long reads and broadcast schedule.". The fifth card is for "ANSA.it", which provides "Agenzia ANSA: ultime notizie, foto, video e approfondimenti su: cronaca, politica, economia, regioni, mondo, sport, calcio, cultura e tecnologia.". At the bottom of the screen is a navigation bar with icons for recent apps, home, back, and search.

- ABC News**
Your trusted source for breaking news, analysis, exclusive interviews, headlines, and videos at ABCNews.com.
- ABC News (AU)**
Australia's most trusted source of local, national and world news. Comprehensive, independent, in-depth analysis, the latest business, sport, weather and more.
- Aftenposten**
Norges ledende nettavis med alltid oppdaterte nyheter innenfor innenriks, utenriks, sport og kultur.
- Al Jazeera English**
News, analysis from the Middle East and worldwide, multimedia and interactives, opinions, documentaries, podcasts, long reads and broadcast schedule.
- ANSA.it**
Agenzia ANSA: ultime notizie, foto, video e approfondimenti su: cronaca, politica, economia, regioni, mondo, sport, calcio, cultura e tecnologia.

DARK MODE:

The screenshot shows a mobile application interface for a news aggregator or search engine. At the top, there is a purple status bar displaying the time (13:28), signal strength, battery level (77%), and other icons. Below the status bar is a white header bar with a magnifying glass icon and the placeholder text "Find Sources and Articles". To the right of the search bar is a menu icon consisting of three horizontal lines. Below the header are four blue tabs labeled "General", "Business", "Entertainment", and "Health".

The main content area displays five news sources in cards:

- ABC News**: Your trusted source for breaking news, analysis, exclusive interviews, headlines, and videos at ABCNews.com.
- ABC News (AU)**: Australia's most trusted source of local, national and world news. Comprehensive, independent, in-depth analysis, the latest business, sport, weather and more.
- Aftenposten**: Norges ledende nettavis med alltid oppdaterte nyheter innenfor innenriks, utenriks, sport og kultur.
- Al Jazeera English**: News, analysis from the Middle East and worldwide, multimedia and interactives, opinions, documentaries, podcasts, long reads and broadcast schedule.
- ANSA.it**: Agenzia ANSA: ultime notizie, foto, video e approfondimenti su: cronaca, politica, economia, regioni, mondo, sport, calcio, cultura e tecnologia.

At the bottom of the screen are standard Android navigation icons: three vertical bars (Home), a circle (Recent Apps), a triangle (Back), and a person icon (Profile).

13:29



77%



Find Sources and Articles



Health

Science

Sports

Technology



ABC News

Your trusted source for breaking news, analysis, exclusive interviews, headlines, and videos at ABCNews.com.



ABC News (AU)

Australia's most trusted source of local, national and world news. Comprehensive, independent, in-depth analysis, the latest business, sport, weather and more.



Aftenposten

Norges ledende nettavis med alltid oppdaterte nyheter innenfor innenriks, utenriks, sport og kultur.



Al Jazeera English

News, analysis from the Middle East and worldwide, multimedia and interactives, opinions, documentaries, podcasts, long reads and broadcast schedule.



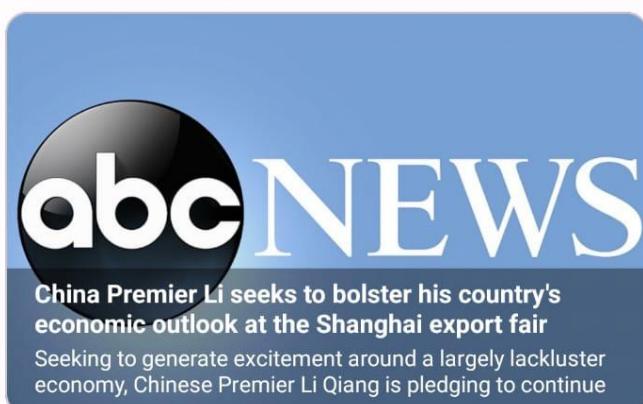
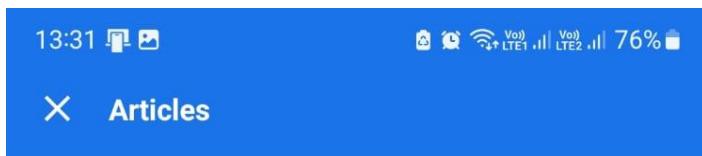
ANSA.it

Agenzia ANSA: ultime notizie, foto, video e approfondimenti su: cronaca, politica, economia, regioni, mondo, sport, calcio, cultura e tecnologia.



OUTPUT 4: News Page

LIGHT MODE:



DARK MODE:

13:29

Articles

China Premier Li seeks to bolster his country's economic outlook at the Shanghai export fair

Seeking to generate excitement around a largely lackluster economy, Chinese Premier Li Qiang is pledging to continue

abc NEWS

China Premier Li seeks to bolster his country's economic outlook at the Shanghai export fair

Seeking to generate excitement around a largely lackluster economy, Chinese Premier Li Qiang is pledging to continue

abc NEWS

Turkey's main opposition party elects Ozgur Ozel as new leader

Turkey's main opposition party has voted for fresh

OUTPUT 5: Specific News Page

Light:

The screenshot shows a mobile news application interface. At the top, there is a blue header bar with the time "13:31", signal strength, battery level at 76%, and the "ABC News" logo. Below the header is the ABC News logo with a bell icon. A navigation menu icon (three horizontal lines) is on the left. On the right, there is a "Rising SEA Chemical Markets" section with an "OPEN >" button. The main content features a large, bold headline: "China Premier Li seeks to bolster his country's economic outlook at Shanghai trade fair". Below the headline is a summary text: "Seeking to generate excitement around a largely lackluster economy, Chinese Premier Li Qiang is pledging to continue deepening reforms, expand free trade zones and relax market access for foreign investment". The author information "By The Associated Press" and the date "November 5, 2023, 1:13 PM" are at the bottom of the article. Below the article is a video player interface showing a man in a suit speaking, with control buttons for volume, brightness, and navigation.

13:31 76%

X ABC News

≡ abcNEWS

Rising SEA Chemical Markets OPEN >

China Premier Li seeks to bolster his country's economic outlook at Shanghai trade fair

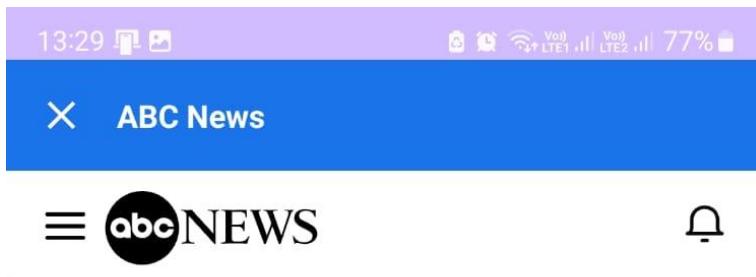
Seeking to generate excitement around a largely lackluster economy, Chinese Premier Li Qiang is pledging to continue deepening reforms, expand free trade zones and relax market access for foreign investment

By The Associated Press
November 5, 2023, 1:13 PM

f t e ↴

||| □ < ⌂

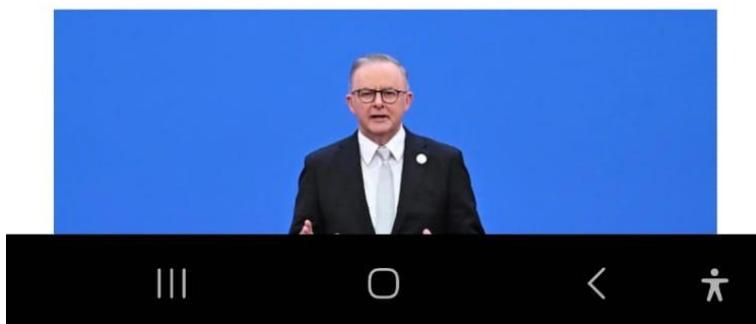
DARK MODE:



China Premier Li seeks to bolster his country's economic outlook at Shanghai trade fair

Seeking to generate excitement around a largely lackluster economy, Chinese Premier Li Qiang is pledging to continue deepening reforms, expand free trade zones and relax market access for foreign investment

By The Associated Press
November 5, 2023, 1:13 PM



GITHUB LINK: <https://github.com/smartinternz02/SI-GuidedProject-587104-1696656485>