



Mobile Application Development

Online News Reader

Final report

Group 3 Members:

Lê Việt Anh BI9-035
Đỗ Thành Đạt BI9-065
Nguyễn Tự Tùng BI10 -187
Vũ Tuấn Phương Nam BI9-022
Trần Bảo Huy BI10-079

I. Abstract

II. Table of works contribution

III. Introduction

- 1. Problem Statement**
- 2. Summary**

IV. General Methodologies

V. Overall Functionalities

VI. Project Overview

- 1. UML diagram**
- 2. Sequence diagram**
- 3. Library**
- 4. Screenshot**

V. Conclusion

I. Abstract

- As the world's technology is swiftly expanding, we are so fortunate to have high-speed connections and networks to directly communicate to another person.
- Due to day-to-day use in mobile, tablet and laptop is increasing, most of the people have already acquired these facilities. Moreover, in this quick and information-oriented world, we also need to stay refreshed with every incident and news.
- Our team was inspired to develop a piece of simple news-reading application to transfer it to users in the best convenient way.

II. Table of works contribution

Name	ID	Role
Lê Việt Anh	BI9-035	Main coder
Đỗ Thành Đạt	BI9-065	PPT designer, Presentator, Report Writer
Nguyễn Tự Tùng	BI10-187	Report Writer
Trần Bảo Huy	BI10-079	Code Debugger / Reviewer
Vũ Tuấn Phương Nam	BI9-022	

III. Introduction

1. Problem Statement

- The traditional newspaper was always the first and most crucial news source, but the arrival of the Internet era has brought a new competitor - online news.
- First and foremost, the essential role of the traditional newspaper is undeniable. It remains an important source of news for the majority of the population, as the elderly simply do not adopt changes but rather stick to the old habit. They have been buying and reading traditional news for decades - obviously, there should be no obligation for them to switch on Internet news now.
- On the other hand, Online News is likely to dominate, since the age of modern technological advances. One of the key benefits is that the Internet can provide the audiences with free and real-time updated information, which newspapers can never achieve. Another primarily key is the modern news support environment - by finding information in the online world instead of purchasing a newspaper, humanity can save an immense amount of paper and carbon footprint which is the major cause of environmental pollution.

2. Summary

To suit the needs of customers and to make it easier to read newspapers at any time and from any location, this application will automatically update new content from <https://newsapi.org/> in real-time. More, it must have a pleasant and easy-to-use interface for users. Since we were testing on the Nexus 5 running on Android 7 Nougat, our project might not suitable for more former Android devices.

IV. General Methodologies

- **Retrofit:** Powerful framework for authenticating and interacting with APIs and sending network requests with OkHttp.
- **News API:** A simple, easy-to-use REST API that returns JSON search results for current and historic news articles published by over 80,000 worldwide sources.
- **Java:** Popular programming language that can be used on a variety of platforms.
- **Firebase:** A cloud-hosted NoSQL database that lets you store and sync data between your users in real-time.

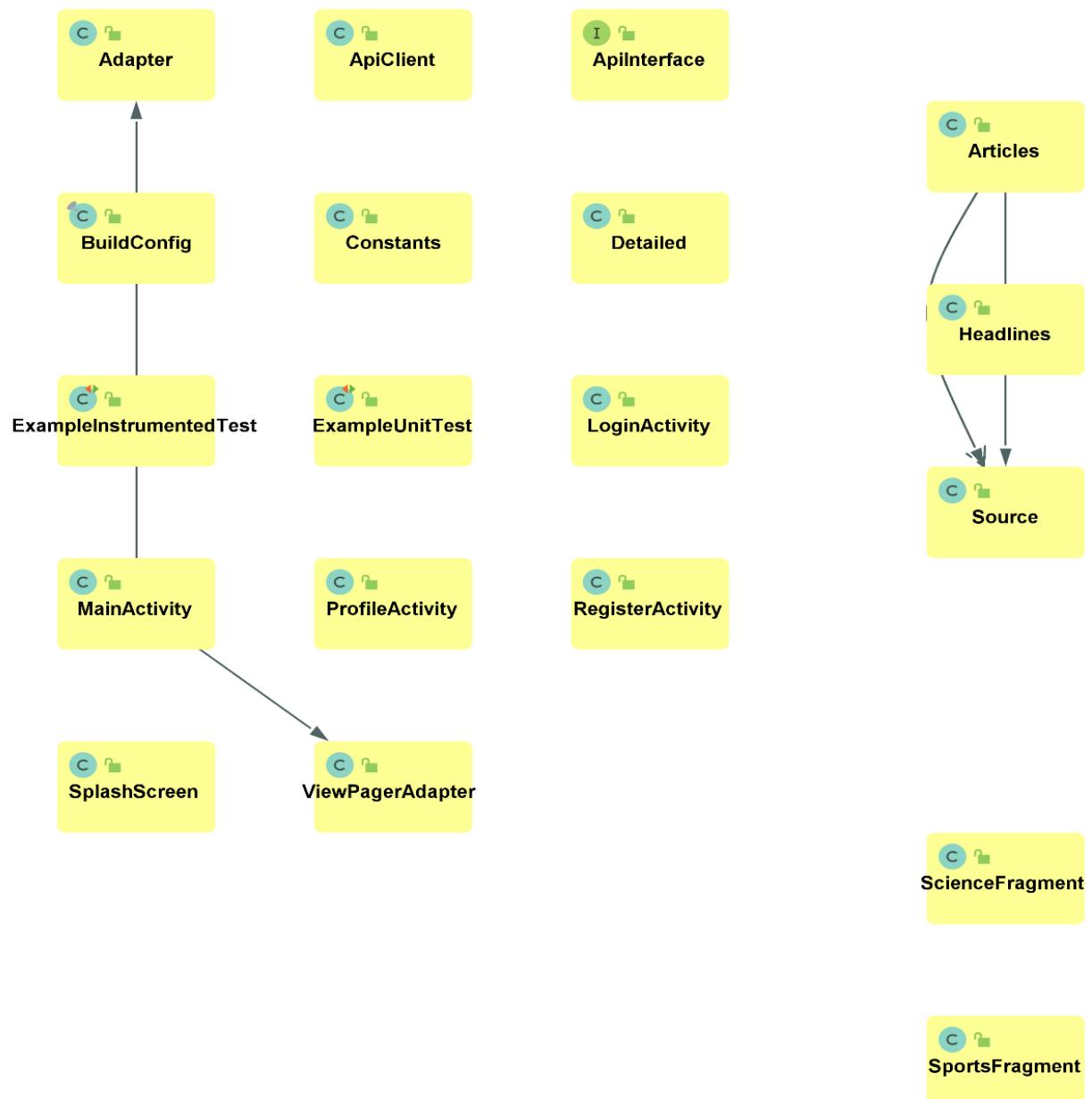
V. Overall Functionalities

- **Authentication:** Register, using the username and password of an existing account via Google accounts.
- **Search function:** Look for an article with keywords.
- **Fetch articles from News API using Retrofit.**
- **Swipe to refresh.**
- **The detailed page corresponds to each news headline.**
- **Splash screen:** display page while waiting for the app to boot

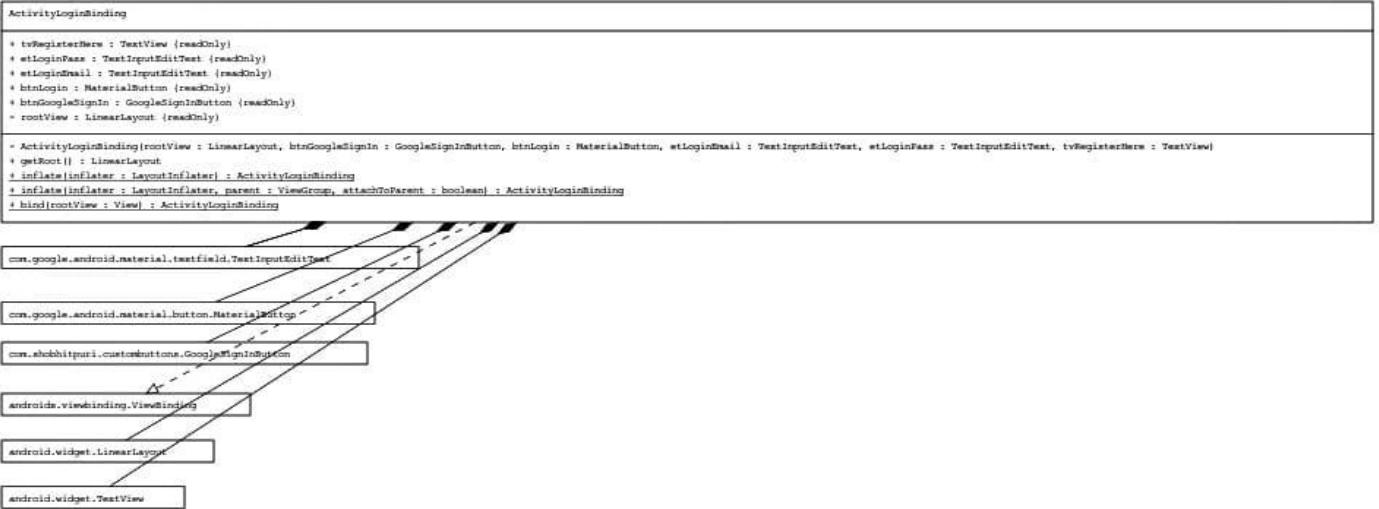
VI. Project Overview

1. UML diagrams:

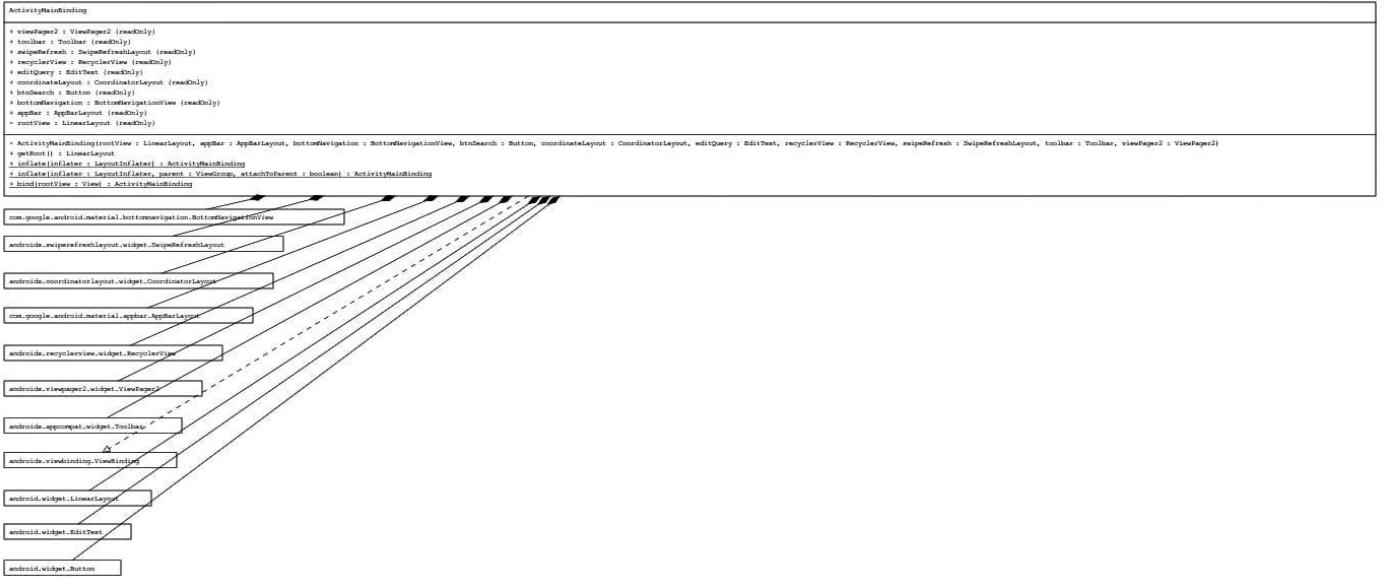
Class diagram:



Activity Login Binding



Activity Main Binding

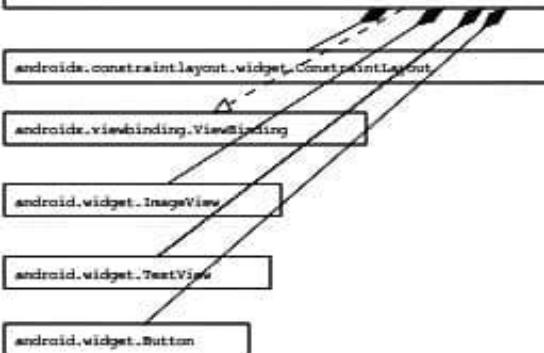


Activity Profile Binding

```
ActivityProfileBinding

+ userImage : ImageView {readOnly}
+ userEmail : TextView {readOnly}
+ userPhone : TextView {readOnly}
+ btnLogout : Button {readOnly}
- rootView : ConstraintLayout {readOnly}

- ActivityProfileBinding(rootView : ConstraintLayout, btnLogout : Button, userEmail : TextView, userImage : ImageView, userPhone : TextView)
+ getRoot() : ConstraintLayout
+ inflate(inflater : LayoutInflater) : ActivityProfileBinding
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : ActivityProfileBinding
+ bind(rootView : View) : ActivityProfileBinding
```

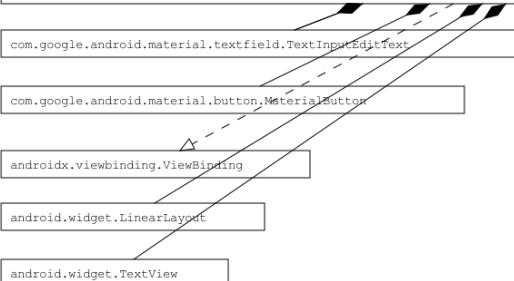


Activity Register Binding

```
ActivityRegisterBinding

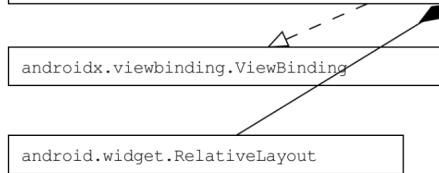
+ tvLoginHere : TextView {readOnly}
+ etRegPass : TextInputEditText {readOnly}
+ etRegEmail : TextInputEditText {readOnly}
+ btnRegister : MaterialButton {readOnly}
- rootView : LinearLayout {readOnly}

- ActivityRegisterBinding(rootView : LinearLayout, btnRegister : MaterialButton, etRegEmail : TextInputEditText, etRegPass : TextInputEditText, tvLoginHere : TextView)
+ getRoot() : LinearLayout
+ inflate(inflater : LayoutInflater) : ActivityRegisterBinding
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : ActivityRegisterBinding
+ bind(rootView : View) : ActivityRegisterBinding
```



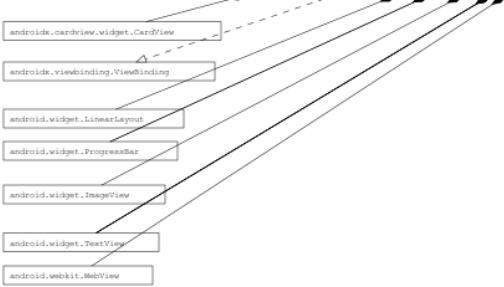
Activity Splash Screen Binding

```
ActivitySplashScreenBinding  
- rootView : RelativeLayout {readOnly}  
  
- ActivitySplashScreenBinding(rootView : RelativeLayout)  
+ getRoot() : RelativeLayout  
+ inflate(inflater : LayoutInflater) : ActivitySplashScreenBinding  
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : ActivitySplashScreenBinding  
+ bind(rootView : View) : ActivitySplashScreenBinding
```

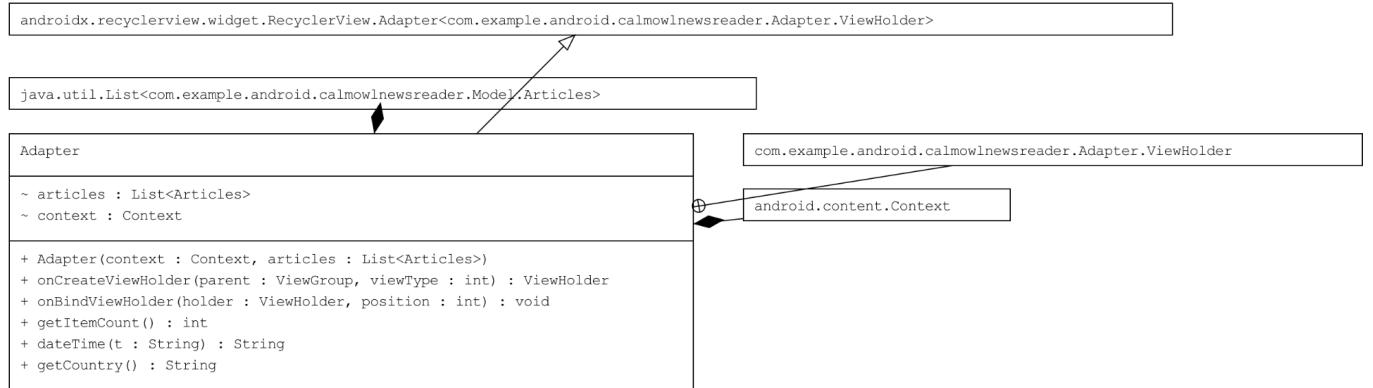


Activity Detailed Binding

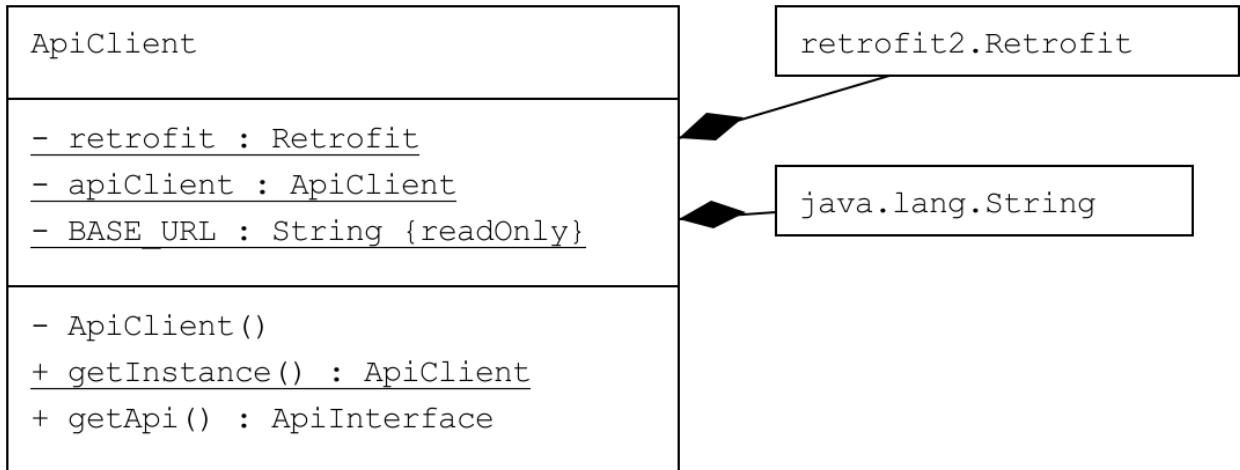
```
ActivityDetailedBinding  
+ webViewLoader : ProgressBar {readOnly}  
+ webView : WebView {readOnly}  
+ tvTitle : TextView {readOnly}  
+ tvSource : TextView {readOnly}  
+ tvDesc : TextView {readOnly}  
+ tvDate : TextView {readOnly}  
+ loader : ProgressBar {readOnly}  
+ imageView : ImageView {readOnly}  
+ cardView : CardView {readOnly}  
- rootView : LinearLayout {readOnly}  
  
- ActivityDetailedBinding(rootView : LinearLayout, cardView : CardView, imageView : ImageView, loader : ProgressBar, tvDate : TextView, tvDesc : TextView, tvSource : TextView, tvTitle : TextView, webView : WebView, webViewLoader : ProgressBar)  
+ getRoot() : LinearLayout  
+ inflate(inflater : LayoutInflater) : ActivityDetailedBinding  
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : ActivityDetailedBinding  
+ bind rootView : View = ActivityDetailedBinding
```



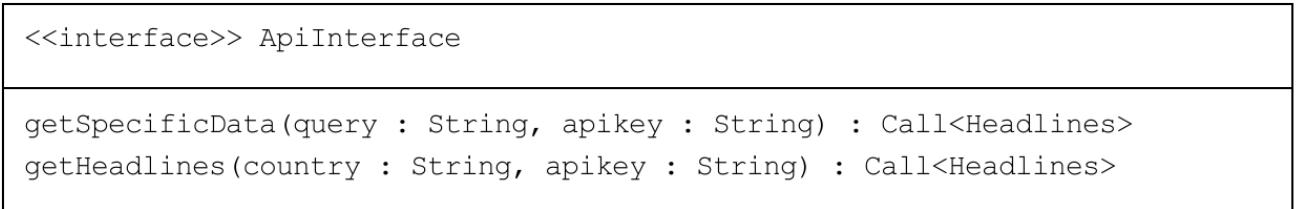
Adapter



ApiClient



ApiInterface



Articles

```
com.example.android.calmowlnewsreader.Model.Source
```

```
Articles
```

```
- publishedAt : String  
- urlToImage : String  
- description : String  
- title : String  
- author : String  
- source : Source  
- url : String  
  
+ getUrl() : String  
+ setUrl(url : String) : void  
+ getSource() : Source  
+ setSource(source : Source) : void  
+ getAuthor() : String  
+ setAuthor(author : String) : void  
+ getTitle() : String  
+ setTitle(title : String) : void  
+ getDescription() : String  
+ setDescription(description : String) : void  
+ getUrlToImage() : String  
+ setUrlToImage(urlToImage : String) : void  
+ getPublishedAt() : String  
+ setPublishedAt(publishedAt : String) : void
```

```
java.lang.String
```

BuildConfig

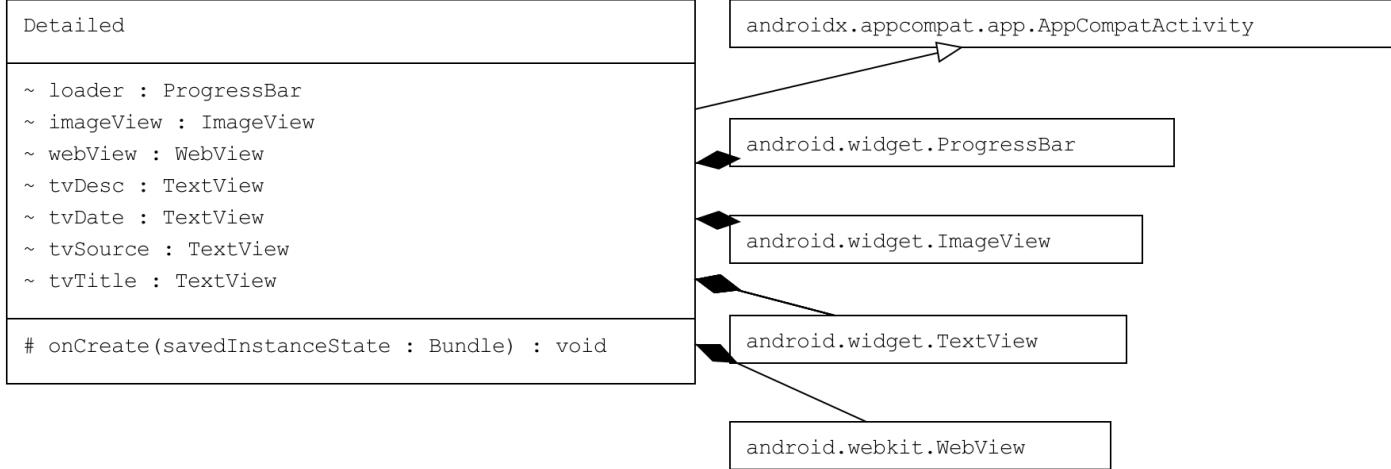
```
<<utility>> BuildConfig  
  
+ VERSION NAME : String {readOnly}  
+ VERSION CODE : int {readOnly}  
+ BUILD TYPE : String {readOnly}  
+ APPLICATION ID : String {readOnly}  
+ DEBUG : boolean {readOnly}
```

Constants

```
<<utility>> Constants
```

```
+ NEWS LOADER ID : int {readOnly}  
+ FRAGMENT SCIENCE : int {readOnly}  
+ FRAGMENT SPORTS : int {readOnly}  
+ FRAGMENT HOME : int {readOnly}  
+ FRAGMENT DASHBOARD : int {readOnly}  
+ API KEY : String {readOnly}
```

Detailed



Example Instrumented Test

```
ExampleInstrumentedTest
```

```
+ useAppContext() : void
```

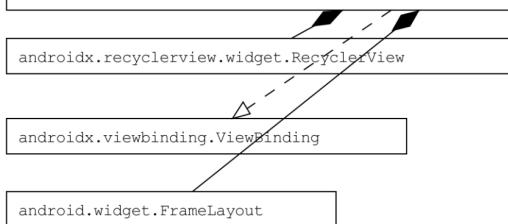
Example Unit Test

```
ExampleUnitTest
```

```
+ addition_isCorrect() : void
```

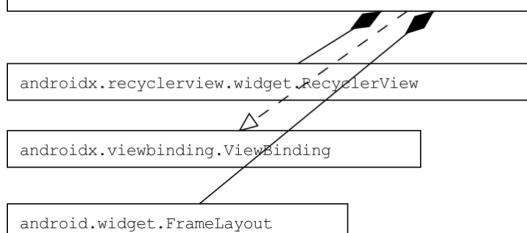
Fragment Science Binding

```
FragmentScienceBinding  
+ fragmentScience : RecyclerView {readOnly}  
- rootView : FrameLayout {readOnly}  
  
- FragmentScienceBinding(rootView : FrameLayout, fragmentScience : RecyclerView)  
+ getRoot() : FrameLayout  
+ inflate(inflater : LayoutInflater) : FragmentScienceBinding  
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : FragmentScienceBinding  
+ bind(rootView : View) : FragmentScienceBinding
```



Fragment Sports Binding

```
FragmentSportsBinding  
+ fragmentSports : RecyclerView {readOnly}  
- rootView : FrameLayout {readOnly}  
  
- FragmentSportsBinding(rootView : FrameLayout, fragmentSports : RecyclerView)  
+ getRoot() : FrameLayout  
+ inflate(inflater : LayoutInflater) : FragmentSportsBinding  
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : FragmentSportsBinding  
+ bind(rootView : View) : FragmentSportsBinding
```



Headlines

```
java.util.List<com.example.android.calmowlnewsreader.Model.Articles>
```

Headlines

```
- articles : List<Articles>
- totalResults : String
- status : String

+ getStatus() : String
+ setStatus(status : String) : void
+ getTotalResults() : String
+ setTotalResults(totalResults : String) : void
+ getArticles() : List<Articles>
+ setArticles(articles : List<Articles>) : void
```

java.lang.String

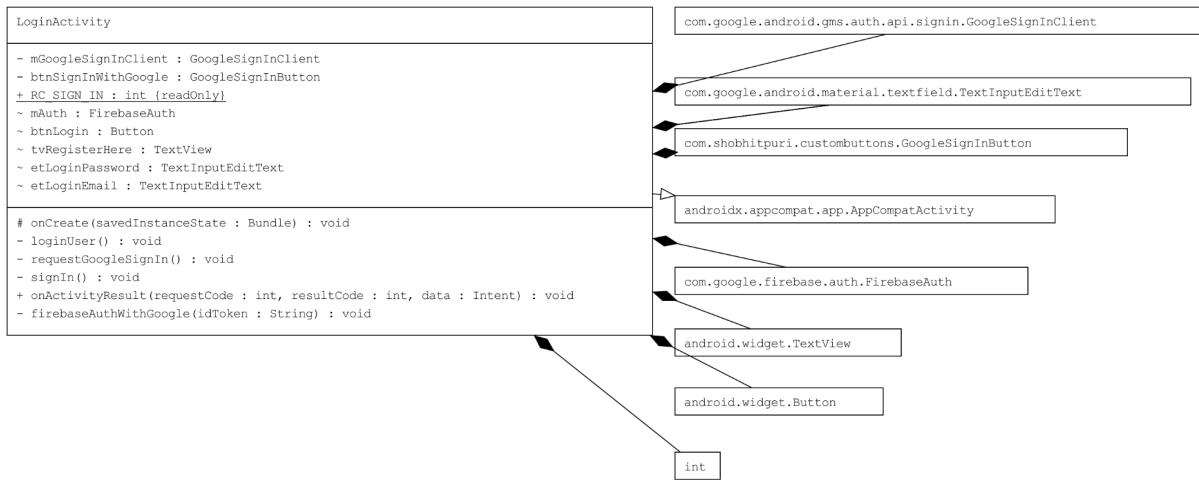
ItemBinding

```
ItemsBinding
+ tvTitle : TextView (readOnly)
+ tvSource : TextView (readOnly)
+ tvDate : TextView (readOnly)
+ loader : ProgressBar (readOnly)
+ image : ImageView (readOnly)
+ cardView : CardView (readOnly)
- rootView : LinearLayout (readOnly)

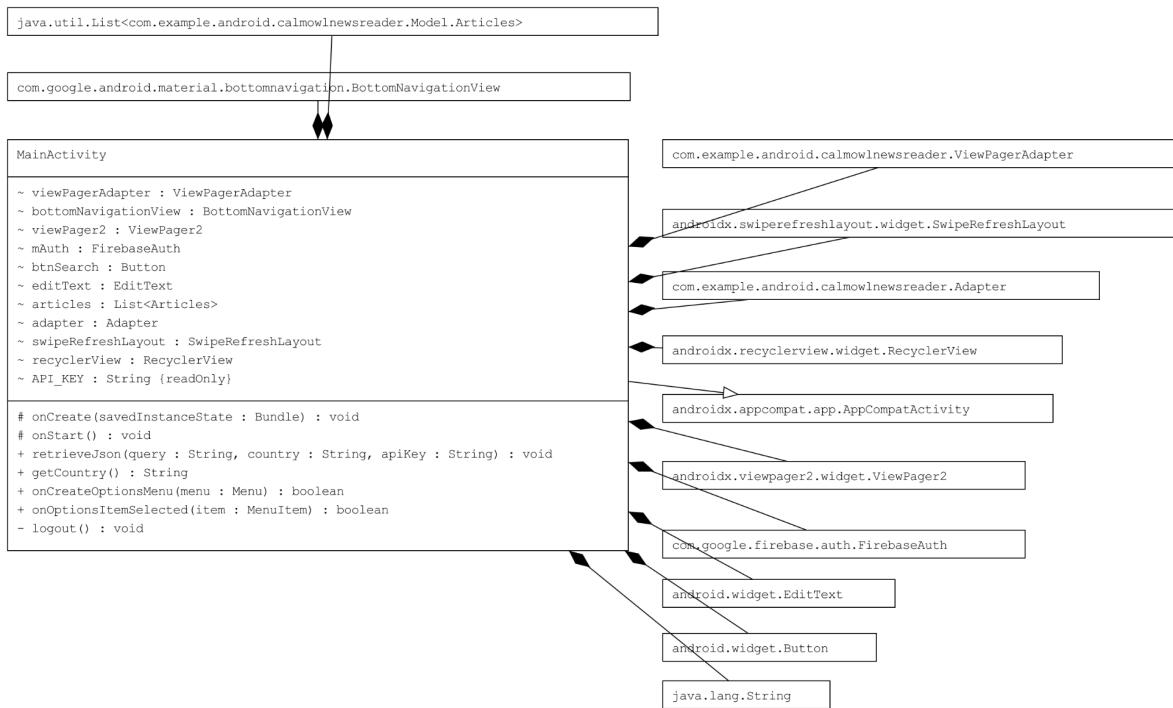
- ItemsBinding(rootView : LinearLayout, cardView : CardView, image : ImageView, loader : ProgressBar, tvDate : TextView, tvSource : TextView, tvTitle : TextView)
+ getRoot() : LinearLayout
+ inflate(inflater : LayoutInflater) : ItemsBinding
+ inflate(inflater : LayoutInflater, parent : ViewGroup, attachToParent : boolean) : ItemsBinding
+ bind(rootView : View) : ItemsBinding
```

```
androidx.cardview.widget.CardView
  ↳
androidx.viewbinding.ViewBinding
  ↳
android.widget.LinearLayout
  ↳
android.widget.ProgressBar
  ↳
android.widget.ImageView
  ↳
android.widget.TextView
```

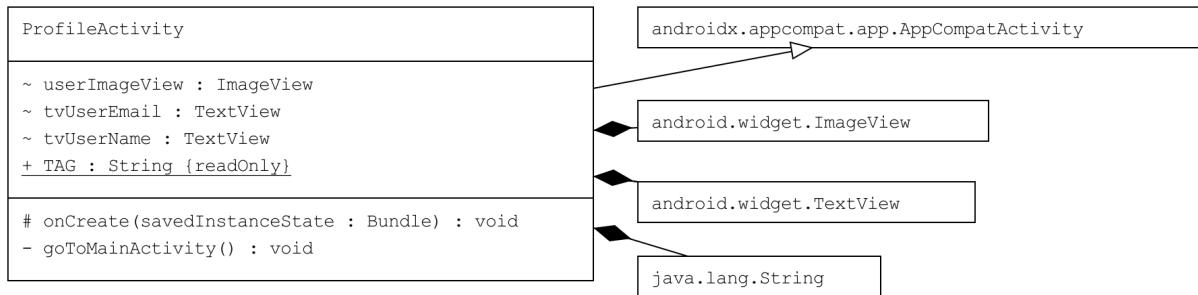
Login Activity



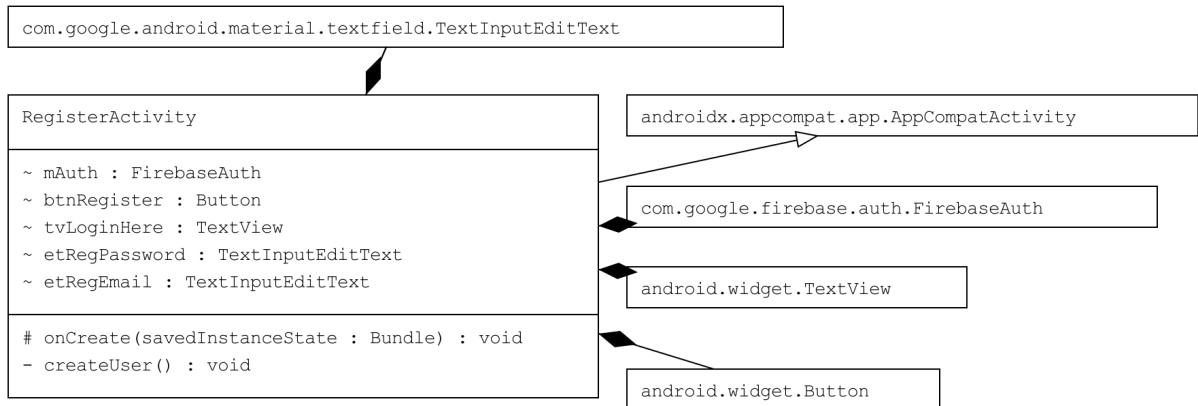
Main Activity



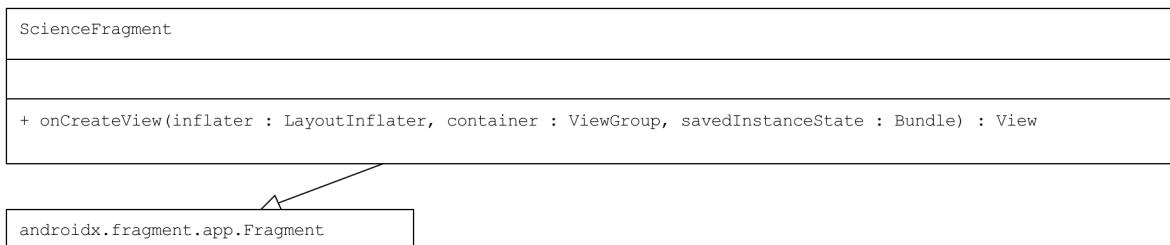
Profile Activity



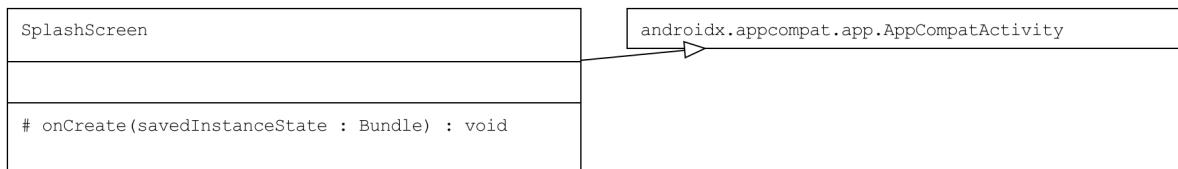
Register Activity



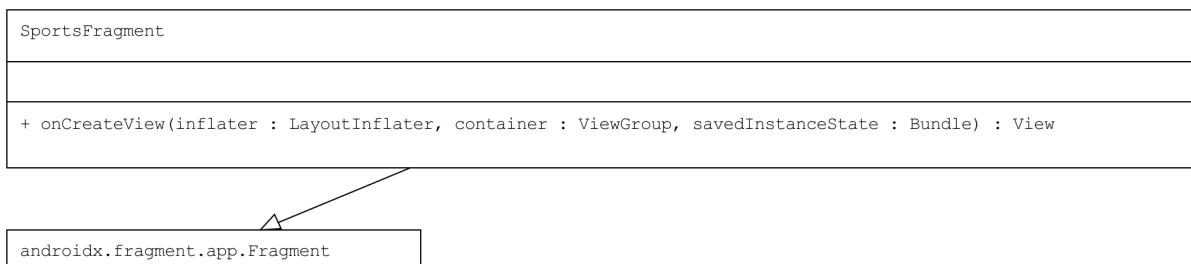
Science Fragment



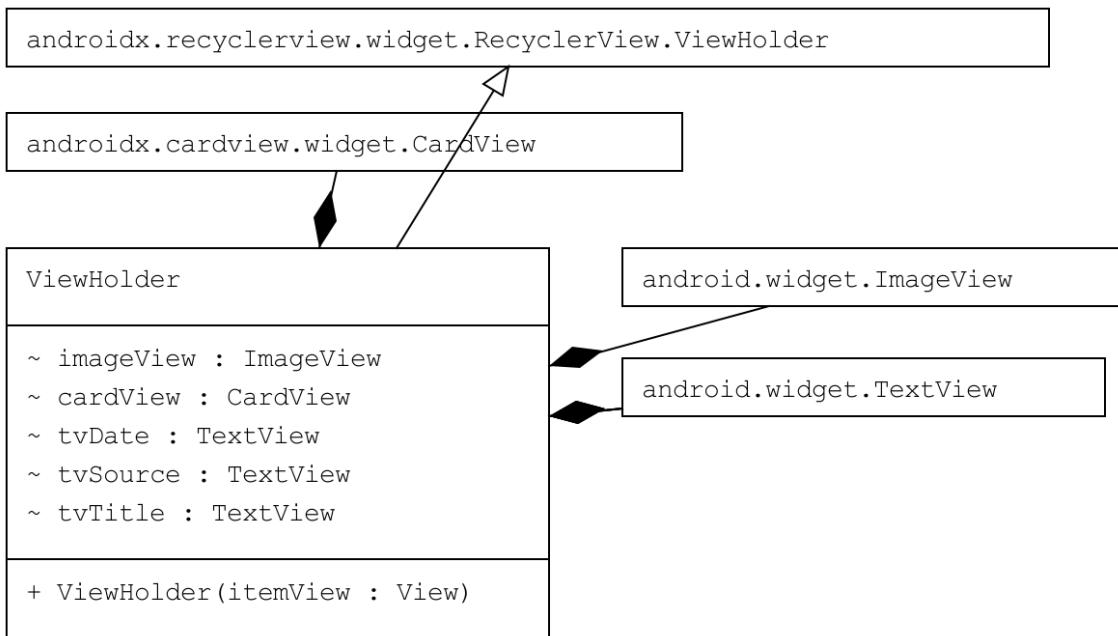
Splash screen



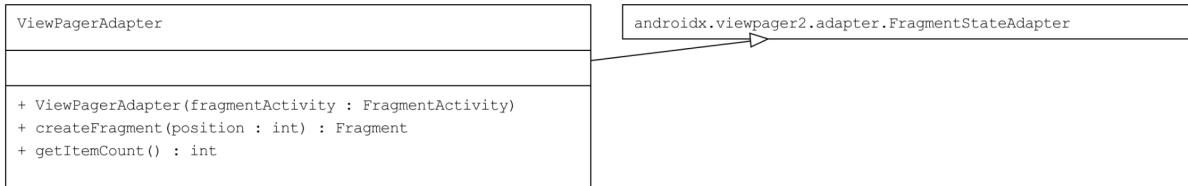
Sports Fragment



ViewHolder

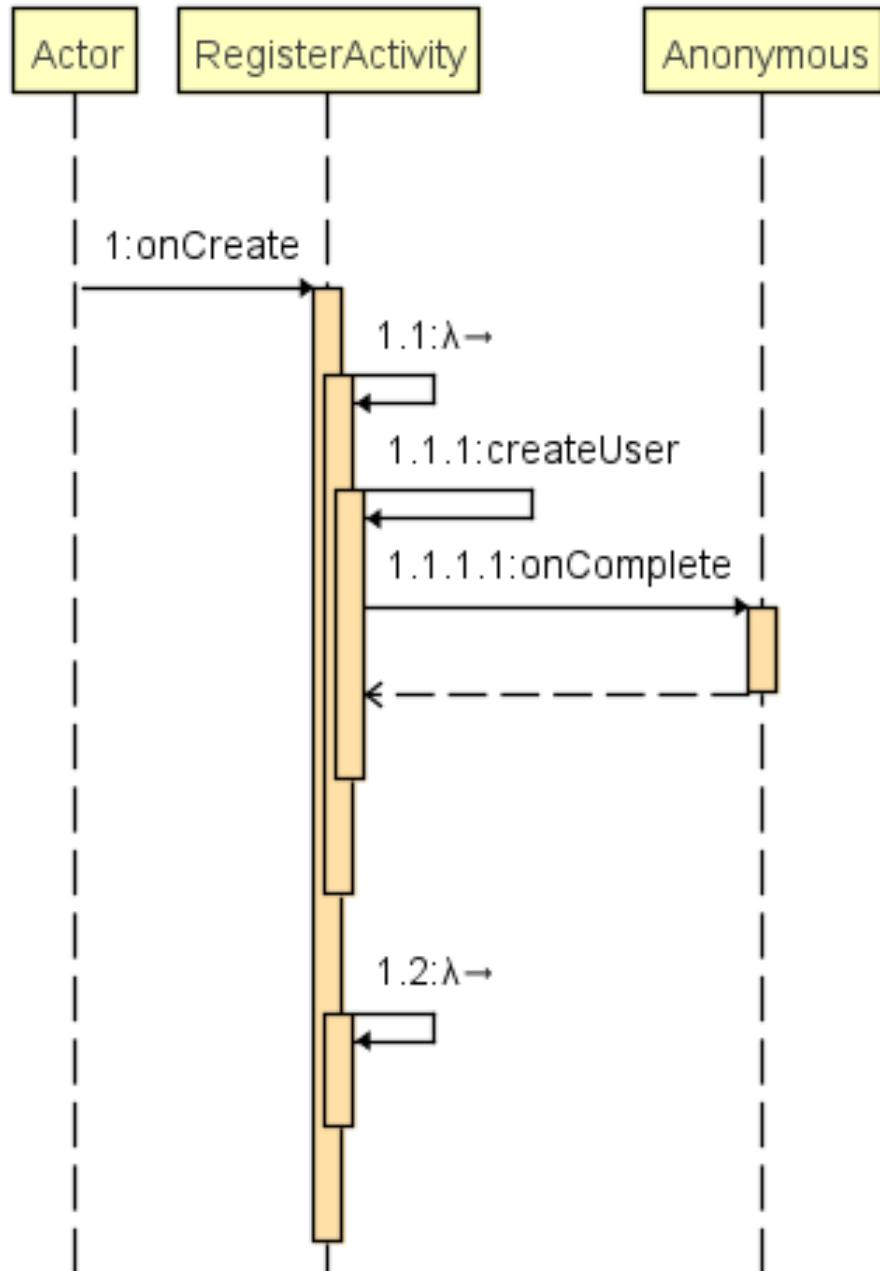


ViewPager Adapter

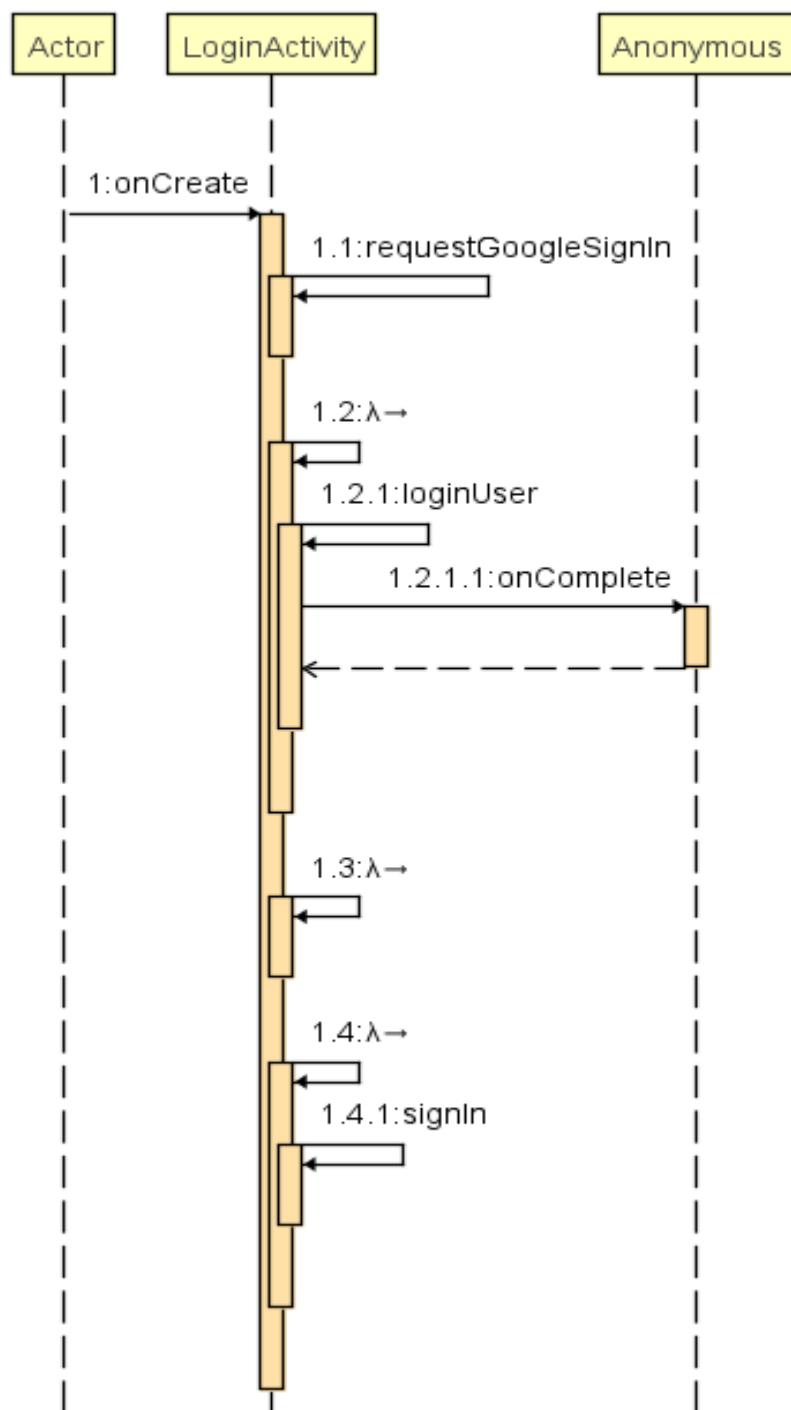


2. Sequence diagram

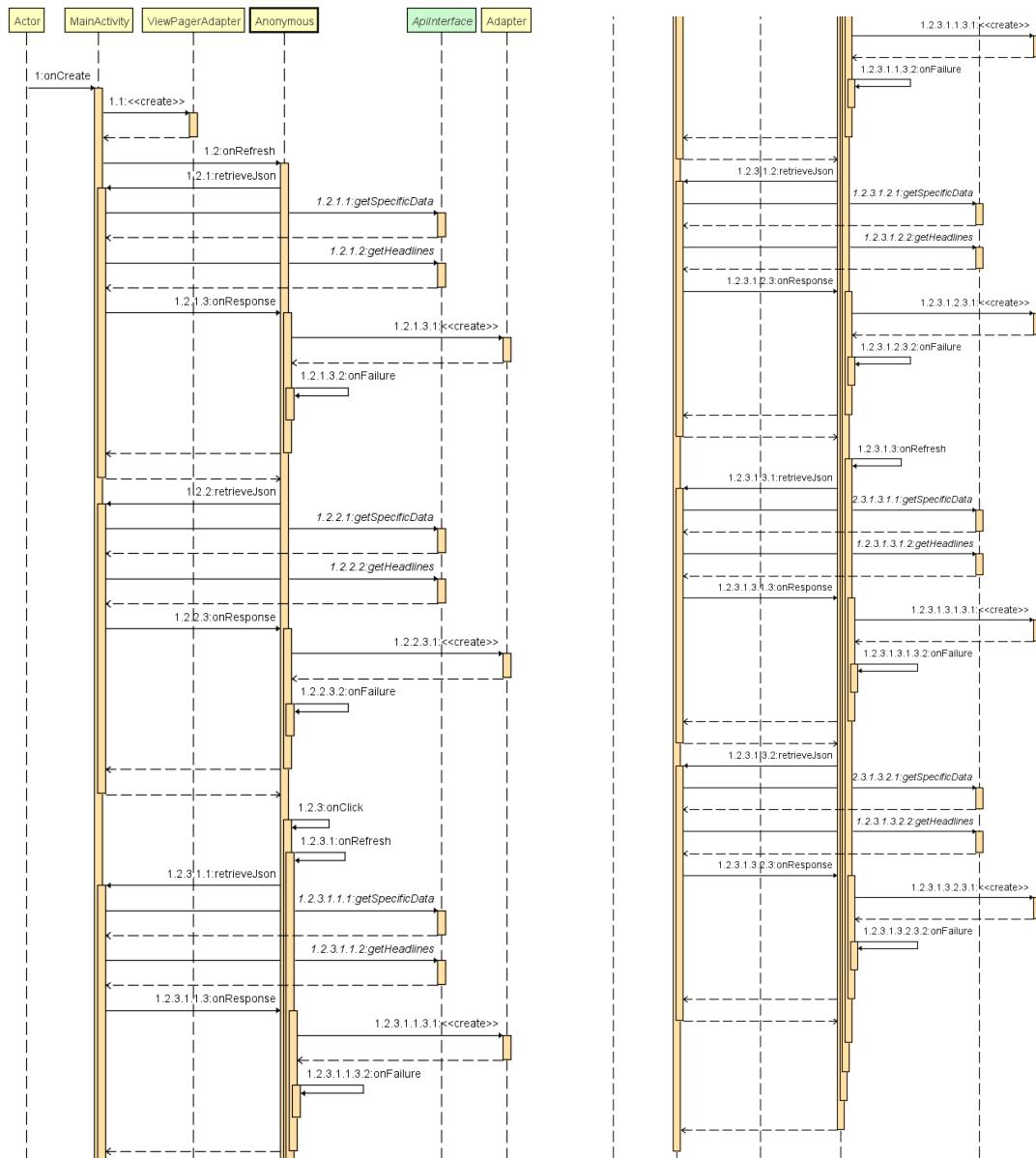
RegisterActivity_OnCreate Sequence diagram



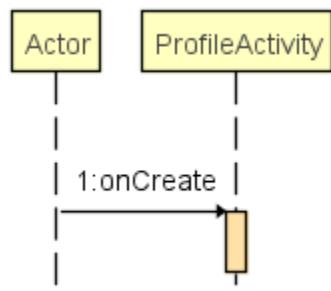
LoginActivity_OnCreate sequence diagram



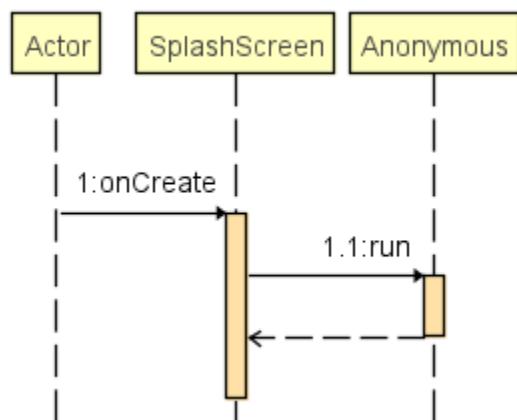
Mainactivity_OnCreate sequence diagram



ProfileActivity_OnCreate Sequence diagram

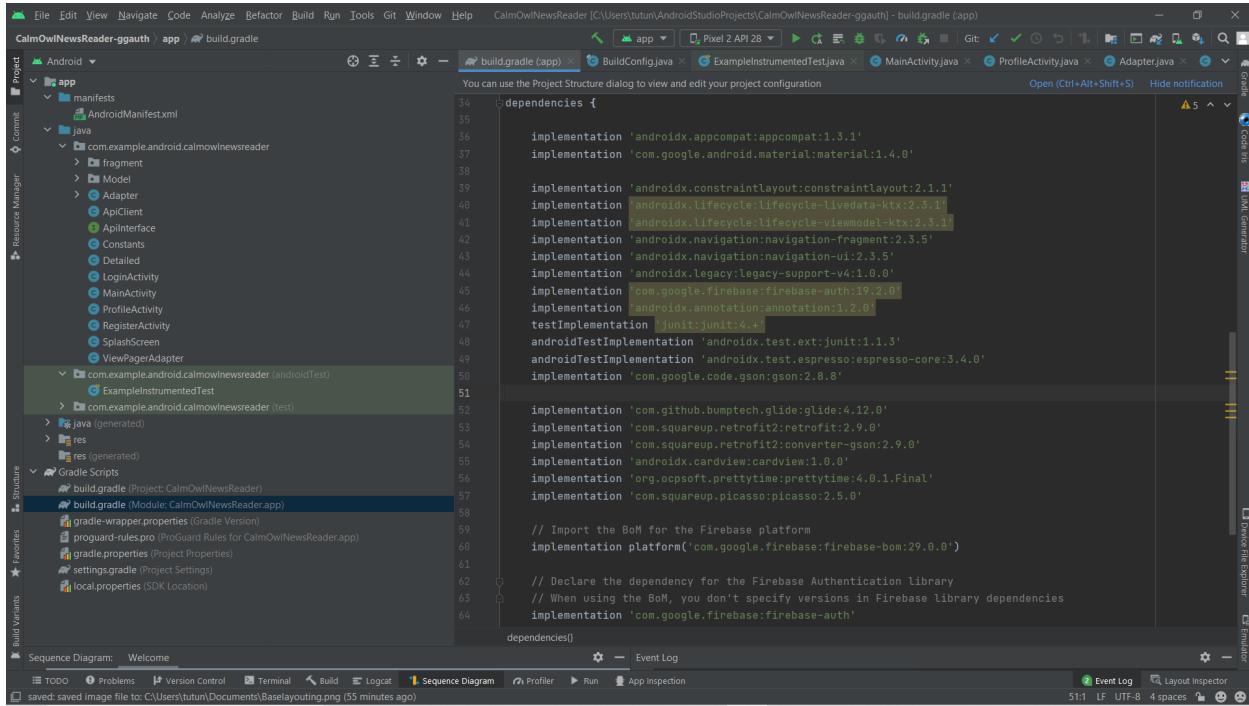


SplashScreen_oncreate Sequence diagram



3. Library

Here are our libraries used for this application:



The screenshot shows the Android Studio interface with the project structure on the left and the build.gradle file open in the main editor. The build.gradle file lists various dependencies, including androidx.appcompat:appcompat, androidx.constraintlayout:constraintlayout, androidx.lifecycle:lifecycle-livedata-ktx, androidx.lifecycle:lifecycle-viewmodel-ktx, androidx.navigation:navigation-fragment, androidx.navigation:navigation-ui, androidx.legacy:legacy-support-v4, com.google.firebase:firebase-auth, com.google.firebase:firebase-database, com.google.firebase:firebase-storage, com.google.firebase:firebase-messaging, com.github.bumptech.glide:glide, com.squareup.retrofit2:retrofit, com.squareup.retrofit2:converter-gson, androidx.cardview:cardview, org.ocpsoft.prettytime:prettytime, com.squareup.picasso:picasso, com.google.firebase:firebase-auth, com.google.firebase:firebase-bom, and com.google.android.gms:play-services-auth.

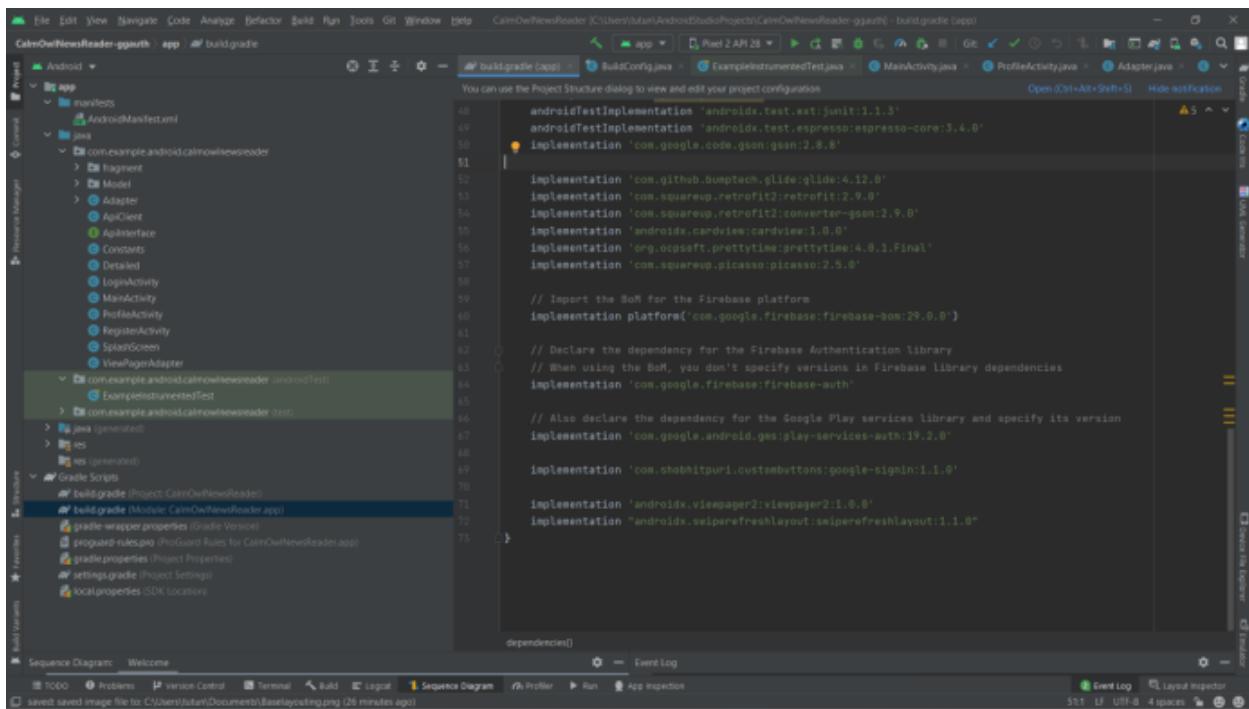
```
dependencies {
    implementation 'androidx.appcompat:appcompat:1.3.1'
    implementation 'com.google.android.material:material:1.4.0'

    implementation 'androidx.constraintlayout:constraintlayout:2.1.1'
    implementation 'androidx.lifecycle:lifecycle-livedata-ktx:2.3.1'
    implementation 'androidx.lifecycle:lifecycle-viewmodel-ktx:2.3.1'
    implementation 'androidx.navigation:navigation-fragment:2.3.5'
    implementation 'androidx.navigation:navigation-ui:2.3.5'
    implementation 'androidx.legacy:legacy-support-v4:1.0.0'
    implementation 'com.google.firebase:firebase-auth:19.2.0'
    implementation 'androidx.annotation:annotation:1.2.0'
    testImplementation 'junit:junit:4.13'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.google.code.gson:gson:2.8.8'

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'org.ocpsoft.prettytime:prettytime:4.0.1.Final'
    implementation 'com.squareup.picasso:picasso:2.5.0'

    // Import the BOM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:29.0.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BOM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebaseio:firebase-auth'
}
```



The screenshot shows the Android Studio interface with the project structure on the left and the build.gradle file open in the main editor. The build.gradle file lists various dependencies, including androidx.test.ext:junit, androidx.test.espresso:espresso-core, com.google.code.gson:gson, com.github.bumptech.glide:glide, com.squareup.retrofit2:retrofit, com.squareup.retrofit2:converter-gson, androidx.cardview:cardview, org.ocpsoft.prettytime:prettytime, com.squareup.picasso:picasso, com.google.firebase:firebase-bom, com.google.firebase:firebase-auth, com.google.android.gms:play-services-auth, com.shubhitpuri.custombuttons:google-signin, androidx.viewpager2:viewpager2, and androidx.swiperefreshlayout:swiperefreshlayout.

```
dependencies {
    androidTestImplementation 'androidx.test.ext:junit:1.1.3'
    androidTestImplementation 'androidx.test.espresso:espresso-core:3.4.0'
    implementation 'com.google.code.gson:gson:2.8.8'

    implementation 'com.github.bumptech.glide:glide:4.12.0'
    implementation 'com.squareup.retrofit2:retrofit:2.9.0'
    implementation 'com.squareup.retrofit2:converter-gson:2.9.0'
    implementation 'androidx.cardview:cardview:1.0.0'
    implementation 'org.ocpsoft.prettytime:prettytime:4.0.1.Final'
    implementation 'com.squareup.picasso:picasso:2.5.0'

    // Import the BOM for the Firebase platform
    implementation platform('com.google.firebase:firebase-bom:29.0.0')

    // Declare the dependency for the Firebase Authentication library
    // When using the BOM, you don't specify versions in Firebase library dependencies
    implementation 'com.google.firebaseio:firebase-auth'

    // Also declare the dependency for the Google Play services library and specify its version
    implementation 'com.google.android.gms:play-services-auth:19.2.0'

    implementation 'com.shubhitpuri.custombuttons:google-signin:1.1.0'

    implementation 'androidx.viewpager2:viewpager2:1.0.0'
    implementation "androidx.swiperefreshlayout:swiperefreshlayout:1.1.0"
}
```

This is the code that our team thinks it's important. This code calls to News API > get information as JSON type and display the retrieved content.

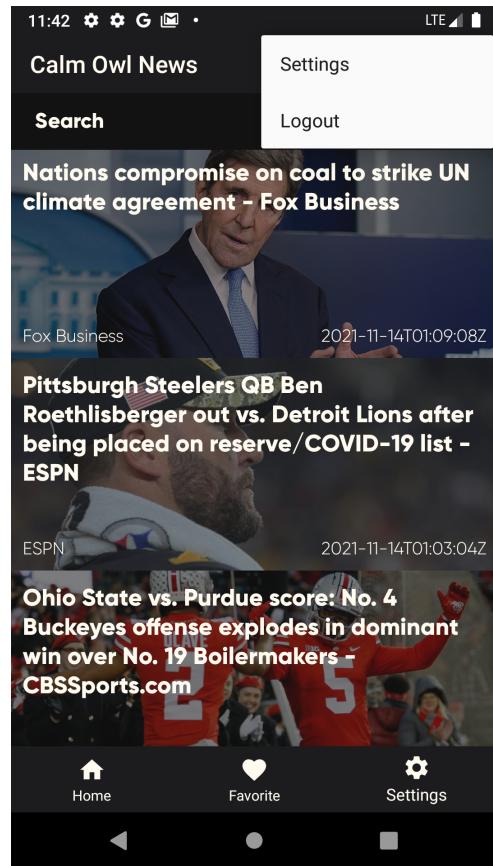
```
public void retrieveJson(String query, String country, String apiKey) {
    swipeRefreshLayout.setRefreshing(true);
    Call<Headlines> call;
    if (!editText.getText().toString().equals("")) {
        call = ApiClient.getInstance().getApi().getSpecificData(query, apiKey);
    }
    else {
        call = ApiClient.getInstance().getApi().getHeadlines(country, apiKey);
    }
    call.enqueue(new Callback<Headlines>() {
        @Override
        public void onResponse(Call<Headlines> call, Response<Headlines> response) {
            if(response.isSuccessful() && response.body().getArticles() != null) {
                swipeRefreshLayout.setRefreshing(false);
                articles.clear();
                articles = response.body().getArticles();
                // fill in the contents from articles
                adapter = new Adapter(MainActivity.this, articles);
                recyclerView.setAdapter(adapter);
            }
        }
        @Override
        public void onFailure(Call<Headlines> call, Throwable t) {
            swipeRefreshLayout.setRefreshing(false);
            Toast.makeText(MainActivity.this, t.getLocalizedMessage(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

4. Screenshot of the apps:

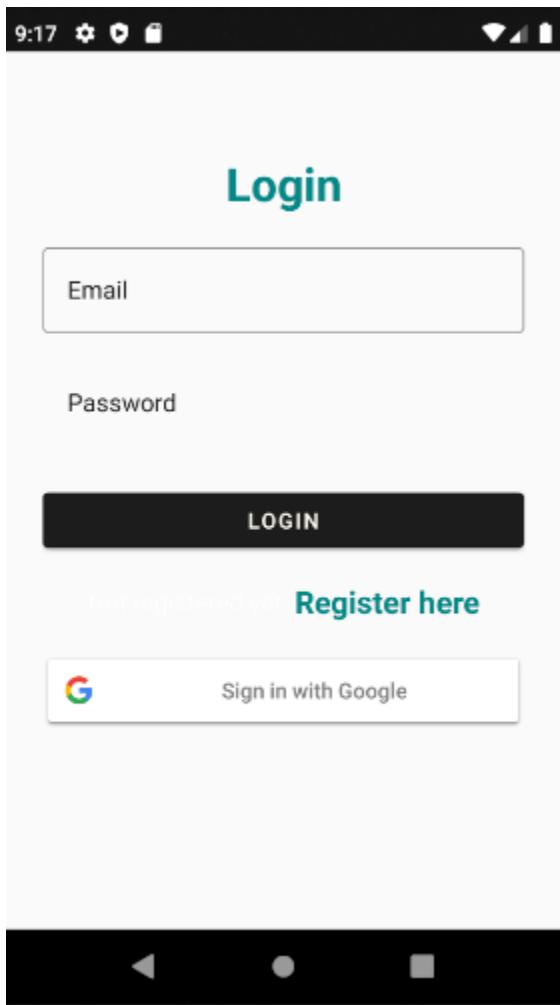
Splash screen



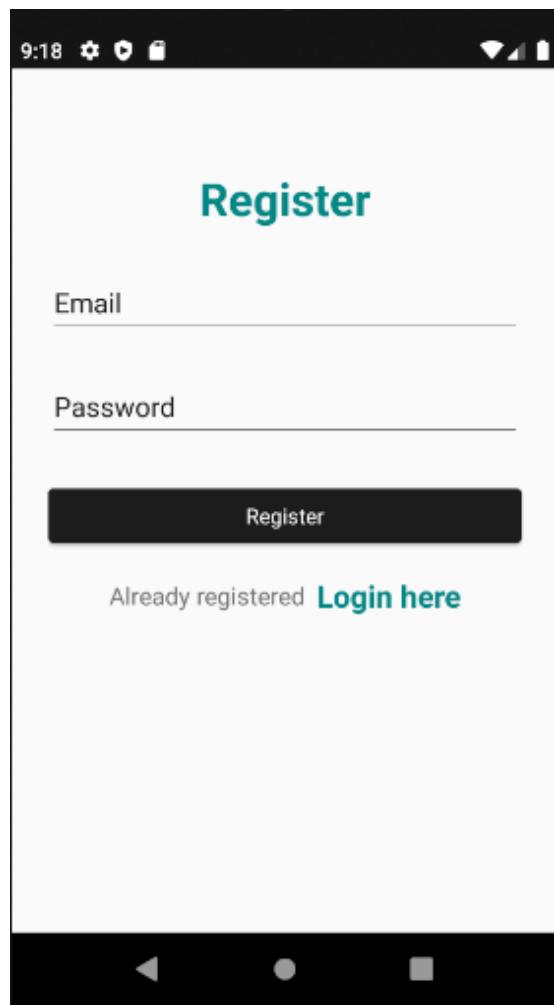
Setting bar



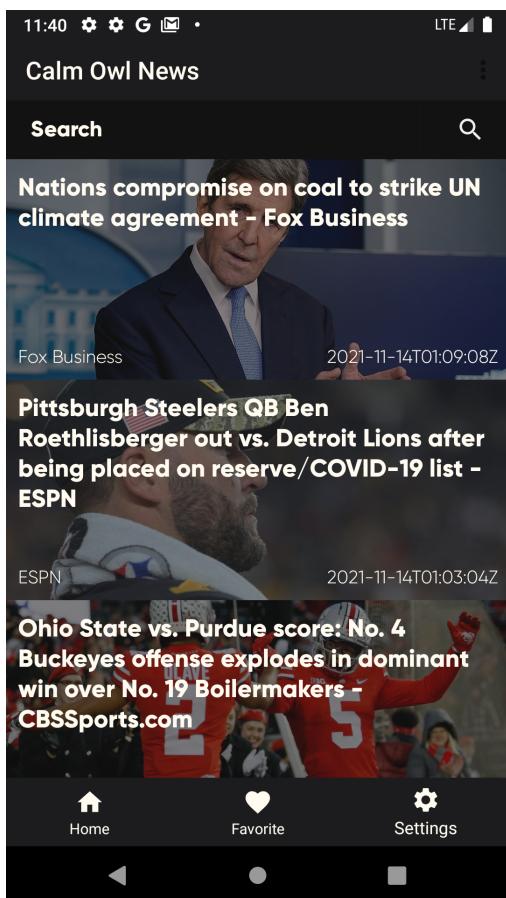
Login



Register



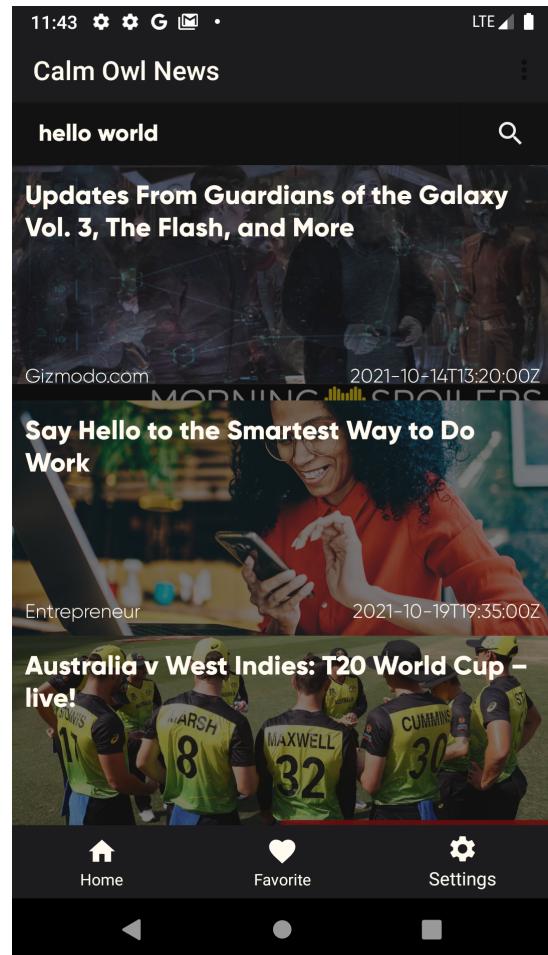
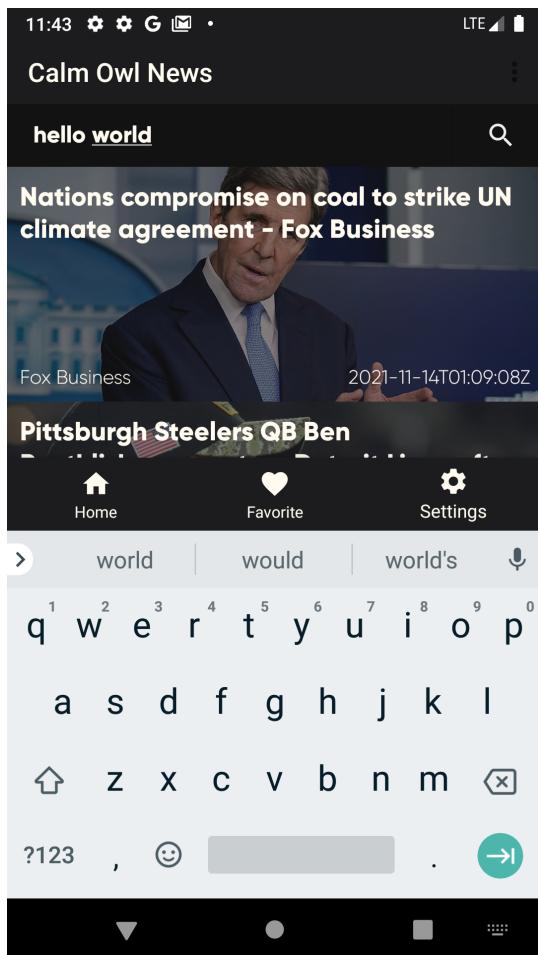
Main page:



Detailed page



Searching fragment



VII. Conclusion

Even though this application is working properly, it requires a certain need for improvement. In the future, we will further explore the design of adaptive interfaces, in order to be in a position to demonstrate a complete adaptive mobile news framework providing mechanical personalization of news app. Moreover, we could implement some sort of machine learning algorithm to automatically synchronize users' experiences. Last but not least, it is a must to update the functionality of the bottom navigation bar with different categories, along with the interface of users' profiles.