
II.3510 – Mobile Application Development for Android

Accompany Document - Group 5 - MovieMan

Group Composition

:

Jun MA 61549

Xianqi LIU 61551

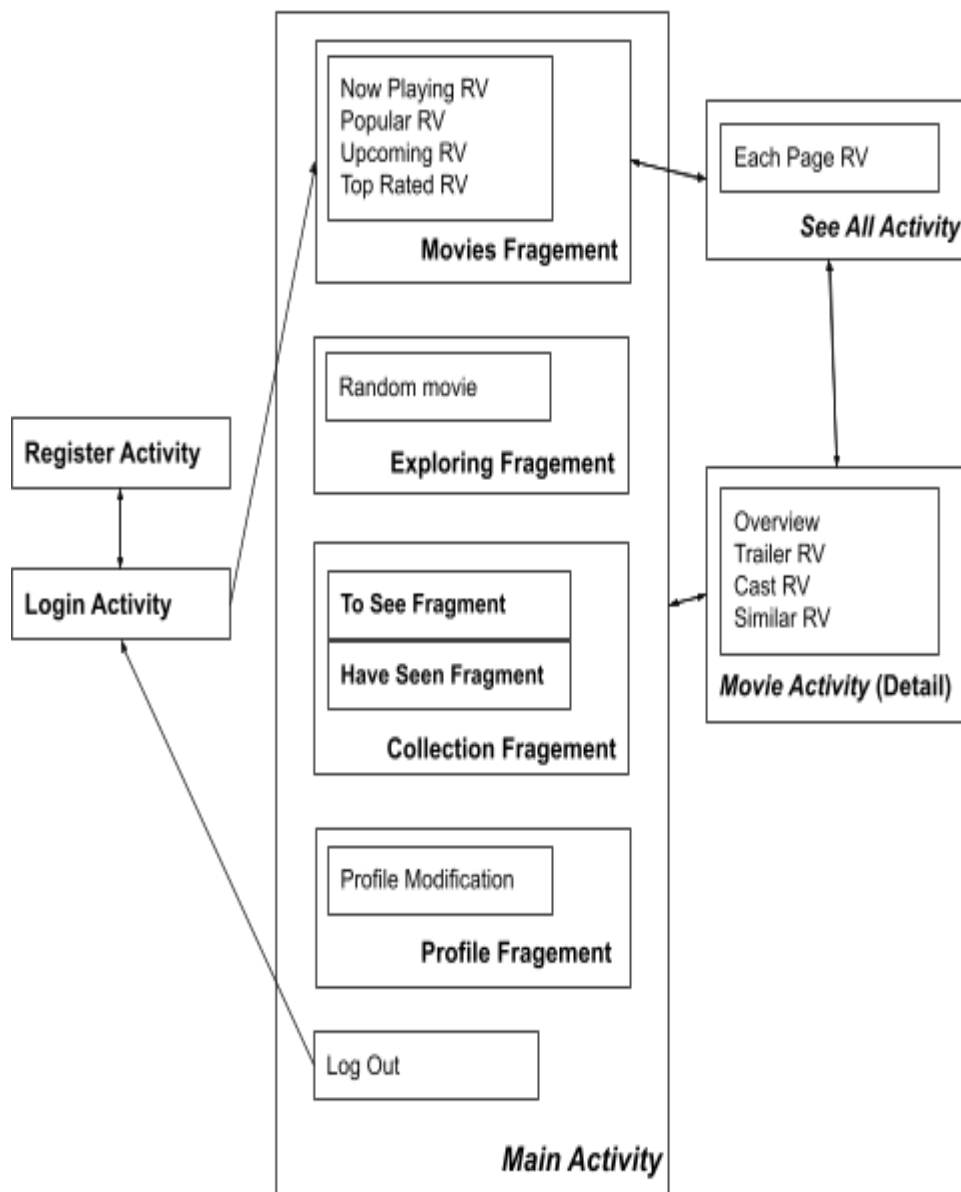
23 Jan. 2022

Github Link	2
Overview (Activity - Fragment - View Relationship)	2
Functional Point Listing	3
Account Register/Login/Logout/Modify	3
Navigation Menu	3
Movies quick-view and page-view based on Now playing / Popular / Upcoming / Top rated movies	4
Display movie's details by Name / Poster / Released date / Run time / Types / Rating / Overview / Trailer / Cast	5
Random movies recommendation	6
Collection: To see / Have seen movies listing	6
Technical Point Listing	7
Firebase	7
Retrofit2	8
Glide	8
Lombok	9
View Binding	9
DrawerLayout NavigationView	10
One Activity to Many Fragments	10
Swipe view - TabLayout + ViewPager	11

Github Link

- <https://github.com/kyleliuxianqi/MovieMan>

Overview (Activity - Fragment - View Relationship)



*The 'RV' here means RecyclerView.

Functional Point Listing

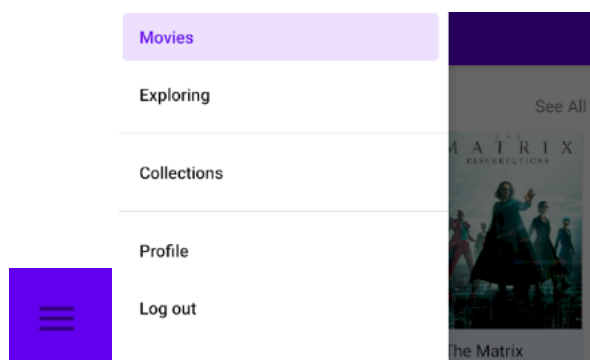
1. Account Register/Login/Logout/Modify

The top screenshot shows a registration form with three input fields: Email, Password, and Display Name. Below the fields are two buttons: SIGN IN and REGISTER. Below the buttons are two links: 'Don't have an account? Register now' and 'Already have an account? Sign in now'.

The bottom screenshot shows the same form with validation errors. The email field contains 'isepuser@gmail' and has a red exclamation mark icon with a tooltip that says 'Not a valid username'. The password field contains 'isepuser@gmail.com' and has a red exclamation mark icon with a tooltip that says 'Password must be >5 characters'.

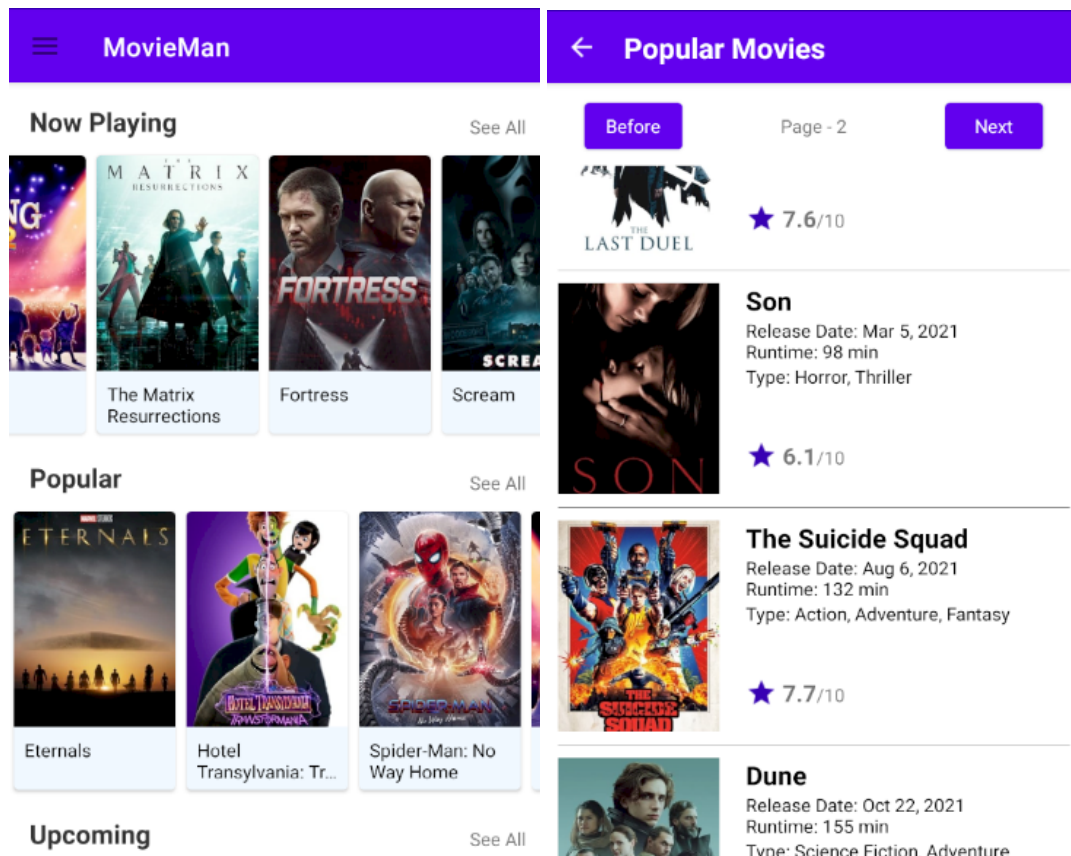
- Only need email, password and displayname can the user register an accountant for this application then log in. Besides, user can also change his password and modify his display name.
- The user name must be a valid email address, and the password must be greater than 5 characters.

2. Navigation Menu



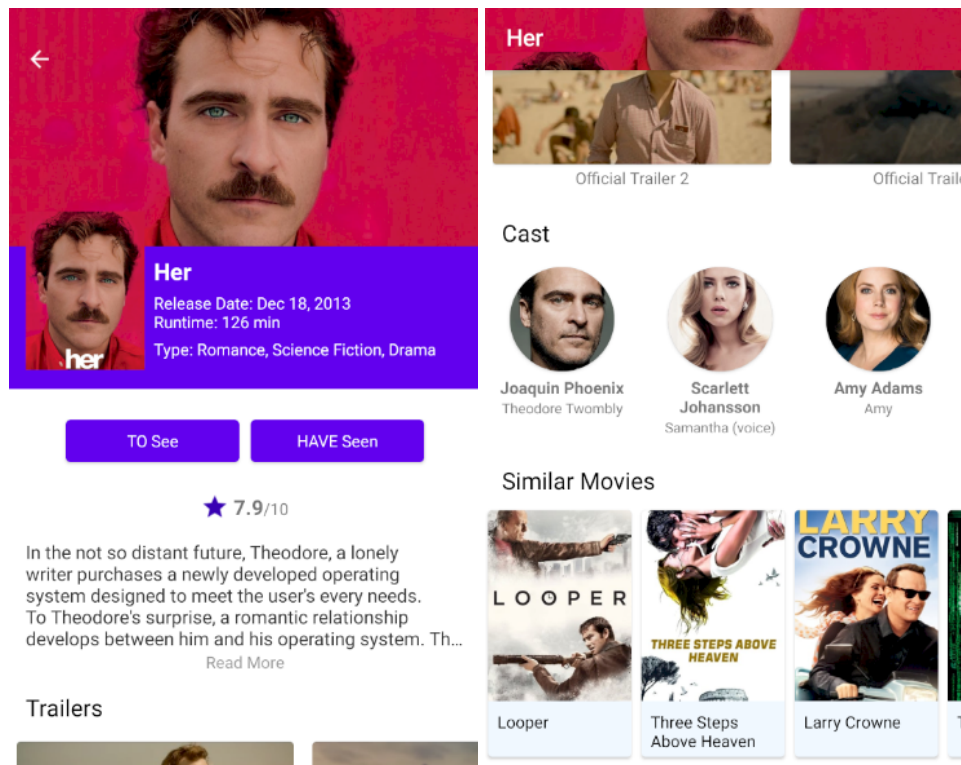
- A leftside drawer navigation menu including different clickable item helps the user go to different fragments.

3. Movies quick-view and page-view based on Now playing / Popular / Upcoming / Top rated movies



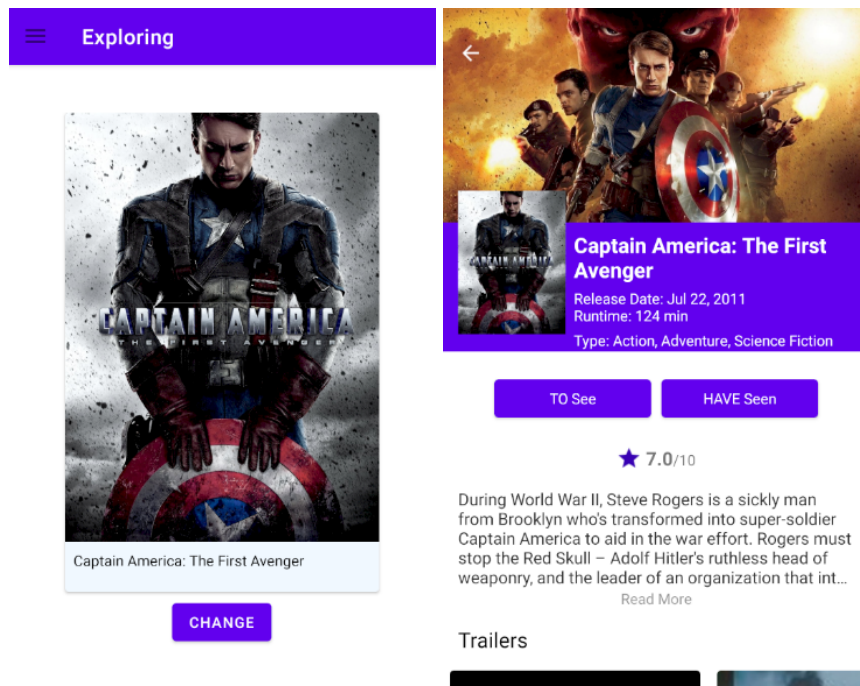
- User can have a quick view of Now playing/Popular/Upcoming/Top rated movies, each part of them can be horizontally scrolled
- The clickable text “See All” helps the user go to the activity which displays all the movies belonging to the chosen category on the previous fragment. Besides, it supports the user view these movies page by page, and the text “page - x” is also clickable, which can help the user to automatically scroll the recycler view back to the first item.

4. Display movie's details by Name / Poster / Released date / Run time / Types / Rating / Overview / Trailer / Cast



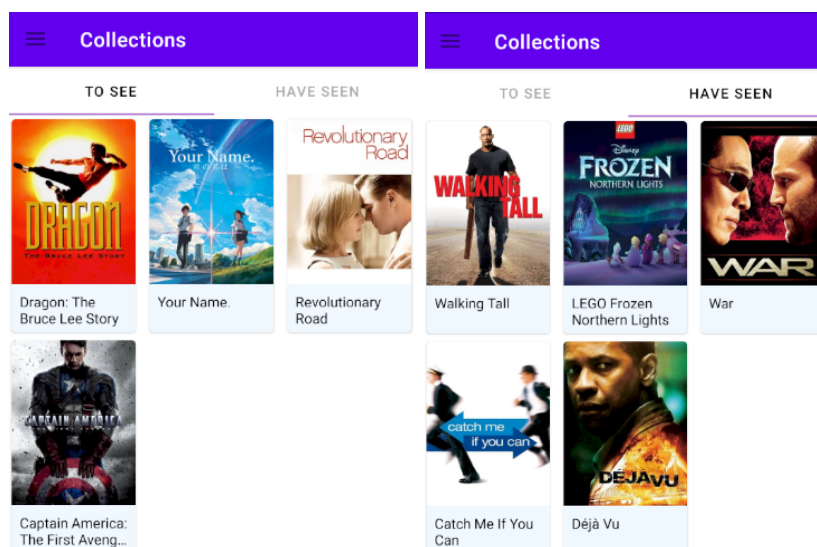
- A top toolbar displays the movie's name, poster, backdrop, released date, run time and type. Additionally, the toolbar is flexible, which will reduce its height and turn to a bar only including movies' name when user scrolls down the view.
- Providing the user with two buttons for adding the movie of the current view to their "TO See" or/and "Have Seen" collection list, and vice versa for removing.
- A rating view for displaying how much the movie is rated from IMDb.
- A text view for displaying the movie's synopsis, and the clickable text "Read more" for unfolding the text view.
- A recycler view for displaying the movie's trailers, each item of them can be clicked then go to the browser or Youtube app.
- A recycler view for displaying the casts' pictures, real name and character names in this movie.
- A recycler view for displaying the similar movies recommendation, each item of them can be clicked then go to the details page.

5. Random movies recommendation



- Providing user with a button that everytime after click “CHANGE”, there will be a new random movie recommadation. And if user clicks the movie card, the page will go to this movie’s detail page.

6. Collection: To see / Have seen movies listing



- A tab toolbar for switching the page for “TO SEE” fragment and “HAVE SEEN” fragment.
- Each item of the collection can be clicked to go to the movie’s detail page.

Technical Point Listing

1. Firebase

- Firebase is a powerful platform for your mobile and web application. Firebase can power your app's backend, including data storage, user authentication, static hosting, and more. With Firebase, you can easily build mobile and web apps that scale from one user to one million.

More details and tutorials for Android on:

<https://firebase.google.com/docs/android/setup>

- In this project, we use Firebase as a database to store our user's information.

For implementing the function of Account Register/Login/Logout/Modification, we store user's email and password.

ID	提供方	创建日期 ↓	上次登录日期	用户的 UID
isepuser@gmail.com	✉	2022年1月2...	2022年1月2...	OAU14K7v0HYvS3N0e3Y2D9nFH...
123456789@gamil.com	✉	2022年1月2...	2022年1月2...	cROJboWoBxQa2L4RqyyF4CW3U...
testuser@gmail.com	✉	2022年1月2...	2022年1月2...	MZxcvq1mnfeqsgSESRM8kuoja1...
test123@gmail.com	✉	2022年1月2...	2022年1月2...	7HD0Y5wjeqXUKhD3GwaYmKJvQ...
2@2.com	✉	2022年1月1...	2022年1月2...	CpwDAaZDwpcCzqBTcxmufBdof...
1@1.com	✉	2022年1月1...	2022年1月2...	6RlbelgpFR5sLINWE4DZeriGAD3

For implmenting the function of user's Collection Addition/Removal/Display, we store Movie's Id after user clicking the To See/Have Seen button in Movie's details page.

movieman-b3b65	OAU14K7v0HYvS3N0e3Y2D9nF...	haveSeen
+ 开始收集	+ 添加文档	+ 开始收集
6RlbelgpFR5sLINWE4DZeriGA...	haveSeen	+ 添加字段
7HD0Y5wjeqXUKhD3GwaYmKJvQI...	toSee	2: 7551
CpwDAaZDwpcCzqBTcxmufBdof...		3: 640
MZxcvq1mnfeqsgSESRM8kuoja1...		5: 10431
OAU14K7v0HYvS3N0e3Y2D9nFHK...		6: 431562
cROJboWoBxQa2L4RqyyF4CW3UJ...		7: 11358
		num: 7

The first column is each user's unique ID, the second column is the user's "toSee" and "haveSeen" collections, the third column is the stored data (id: MovieId) of user's "toSee" and "haveSeen" collections.

2. Retrofit2

- A type-safe HTTP client for Android and Java.
More details on <https://square.github.io/retrofit/>

- Retrofit turns your HTTP API into a Java interface.

```
public interface ApiService {  
  
    @GET("movie/{id}")  
    Call<Movie> getMovieDetails(@Path("id") Integer movieId, @Query("api_key") String apiKey);  
}
```

- The Retrofit class generates an implementation of the ApiService interface.

```
public class ApiClient {  
  
    public static final String BASE_URL = "https://api.themoviedb.org/3/";  
  
    private static Retrofit retrofit = null;  
  
    public static Retrofit getClient() {  
        if (retrofit == null) {  
            retrofit = new Retrofit.Builder()  
                .baseUrl(BASE_URL)  
                .addConverterFactory(GsonConverterFactory.create())  
                .build();  
        }  
        return retrofit;  
    }  
}
```

- Each Call from the created ApiService can make a synchronous or asynchronous HTTP request to the remote webserver.

```
ApiService apiService = ApiClient.getClient().create(ApiService.class);  
  
mMovieCall = apiService.getMovieDetails(mMovieId, "ee535b33eb9a9a4898620076656bea76");  
mMovieCall.enqueue(new Callback<Movie>() {  
    @Override  
    public void onResponse(@NonNull Call<Movie> call, @NonNull final Response<Movie> response) {
```

3. Glide

- A fast and efficient open-source media management and image loading framework for Android. More details on: <https://github.com/bumptech/glide>

- **Use case:**

Glide

```
.with(myFragment)
.load(url)
.centerCrop()
.placeholder(R.drawable.loading_spinner)
.into(myImageView);
```

```
Glide.with(requireContext())
    .load(Constants.URL_IMG_LOAD_1280 + movieRandom.getPoster_path())
    .centerCrop()
    .diskCacheStrategy(DiskCacheStrategy.ALL)
    .into(mBinding.itemRandom.ivMovieItemImg);
```

4. Lombok

- A java library that automatically plugs into your editor and build tools, spicing up your java. Never write another getter or equals method again, with one annotation your class has a fully featured builder, Automate your logging variables, and much more.
- More details on: <https://projectlombok.org/setup/android>

- In this project, we mostly use

[@Getter/@Setter](#)

Then we don't need to write the method like `public int getId() {return id;}` again.

5. View Binding

- A feature that allows you to more easily write code that interacts with views. Once view binding is enabled in a module, it generates a binding class for each XML layout file present in that module. An instance of a binding class contains direct references to all views that have an ID in the corresponding layout.

More details on: <https://developer.android.com/topic/libraries/view-binding>

- In this project, we use view binding to replace the traditional `findViewById`.


```
private ActivityMainBinding binding;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);

    binding = ActivityMainBinding.inflate(getLayoutInflater());
    View view = binding.getRoot();

    setContentView(view);
    setSupportActionBar(binding.mainBar.toolbar)
}
```

6. DrawerLayout NavigationView

- The navigation drawer is a UI panel that shows your app's main navigation menu. The drawer appears when the user touches the drawer icon  in the app bar or when the user swipes a finger from the left edge of the screen. More details on: <https://developer.android.com/guide/navigation/navigation-ui>.

7. One Activity to Many Fragments

- In this project, we create a Navigational Drawer and a container layout for MainActivity, the Drawer contains different items which can be clicked to selected fragments, then we use the FragmentManager to replace the container layout with different fragments' layout.

```
@Override
public boolean onNavigationItemSelected(MenuItem item) {
    int id = item.getItemId();
    binding.drawerLayout.closeDrawer(GravityCompat.START);

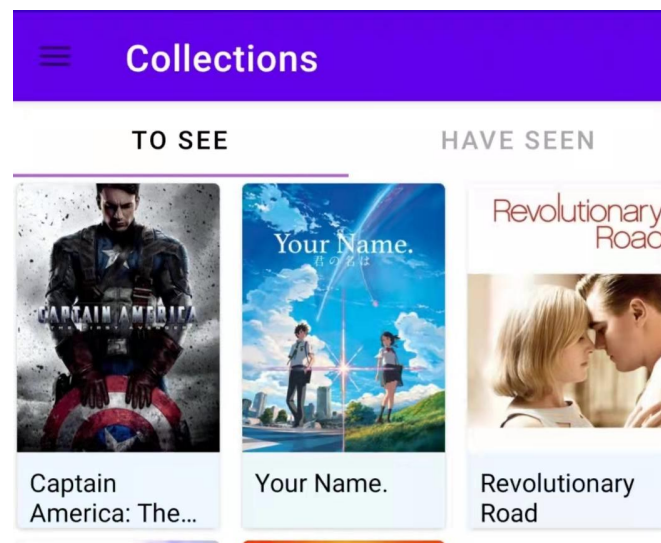
    if(id == R.id.item_movies){
        binding.mainBar.toolbar.setTitle("Movies");
        setFragment(new MoviesFragment());
        return true;
    }else if(id == R.id.item_explore){

```

```
private void setFragment(Fragment fragment) {
    FragmentManager fragmentManager = getSupportFragmentManager();
    FragmentTransaction fragmentTransaction = fragmentManager.beginTransaction();
    fragmentTransaction.replace(R.id.activity_main_fragment_container, fragment);
    fragmentTransaction.commit();
}
```

8. Swipe view - TabLayout + ViewPager

- Swipe views allow you to navigate between sibling screens, such as tabs, with a horizontal finger gesture, or swipe. This navigation pattern is also referred to as horizontal paging.
- More details on:
<https://developer.android.com/guide/navigation/navigation-swipe-view>
- In this project's Collection Fragment, we make a swipe view by using TabLayout and ViewPager Adapter, which help us to switch from the "TO SEE" fragment and the "HAVE SEEN" fragment.



```
public void getTabs(){
    final CollectViewPagerAdapter collectViewPagerAdapter = new CollectViewPagerAdapter(requireActivity().getSupportFragmentManager());

    new Handler().post(() -> {
        collectViewPagerAdapter.addFragment(ToSeeFragment.getInstance(), "TO See");
        collectViewPagerAdapter.addFragment(HaveSeenFragment.getInstance(), "HAVE Seen");

        mBinding.viewPager.setAdapter(collectViewPagerAdapter);
        mBinding.tabLayout.setupWithViewPager(mBinding.viewPager);
    });
}
```