# CS 371M: Homework 5: Reddit

For this assignment, you will build an app that fetches content from reddit and displays it to the user. The user can refresh the contents, see the details on an individual post, and change to a different subreddit. You will also implement a favorites list for the user.

Up to this point your flipped classrooms and homeworks have been "programming in the small." What I mean by that expression is that the task you have to do is simple, and the tools at your disposal are sufficient for the task. This assignment (and I hope your project) are intended to expose you to "programming in the large," even though by absolute line count this homework is pretty small. But the point is that compromises are necessary: there might be multiple ways of doing something and no clear reasons to prefer one over the other. There is the occasional mysterious parameter which isn't terribly logical, but useful for the implementation.

Let me highlight one quick example before we get mired in homework specifics. The model for this homework is a `RedditPost`, which is a structure with 12 fields. Worse, some of these fields are useful when the `RedditPost` is a post to a subreddit, and other fields are useful when the "`RedditPost` is actually information about a subreddit. The reddit API (`https://www.reddit.com/dev/api/`) specifies that a request to list subreddits returns what it calls a list of posts, which is identical to what is returned when you ask for the posts in a subreddit. Is that strictly logical? I would argue that it is not. Even worse, some fields are used when the "post" is a subreddit post and different fields are used when the "post" is information about a subreddit. This is the world we live in and to make progress we must accept reality as it is.

You might say (as I did) that what we need is some sort of inheritance or interface. We have a generic something, `AbstractPost`, that has two subtypes (or interface implementations), `RedditPost` and `SubredditListing`. Well, I started down that road, and Donny, I entered a world of pain. Gson acts like a quiet and omnipotent god when you ask it to decode a class with all primitive types. It just does the right thing. But once you start asking it to create different types of objects, it turns into an angry and vengeful god hell-bent on smiting those who would dare to subtype.

On the other side of the coin, we have been training for this homework all semester. This is a moment to look back (or down?) and see how far you have come. We have coded a lot of these behaviors in isolation, now we get to see them in a single app.

The code as given to you does not build properly. You need to add some code before it will even compile.

**Note**: This homework requires more programming and is a bit more conceptually difficult than your previous homeworks. It will be worth more points. The code I gave you has about 500 lines cut from my implementation.

**Note** that there is STILL no collaboration for homeworks. This homework is challenging, but every individual should be able to do it for themselves (and you will learn a lot).

A note on **how to get json from reddit**. Looking at the API documentation, it is incredibly poorly described how to get json returned from your reddit api request. I actually found the answer in a reddit post, which is some form of irony, or irony in reverse. Here is an example for getting json that describes a particular subreddit. You never need to make this particular call in your app, but if you look at the example, it should make it clear how to get a json response from reddit when you do need it (which you do when getting posts and getting the list of subreddits). `https://www.reddit.com/r/aww/about.json`

By the way, the reddit documentation claims that appending `raw_json=1` to a request will not only return json, it will not embed those annoying HTML encoded characters. Well, if you can get that to work I'm impressed, because it always fails for me.

Let's start with specifying behaviors. You might get all this information from watching the video demo, but I wanted to give it to you in easy-to-digest written form.

- **Startup.** When your app starts, it should fetch the **hot** 100 items in the `aww` subreddit, displaying them in a list.
  - The title bar has three parts, the subreddit name (also known as a title are), the search area, and a solid heart. Clicking the subreddit name brings up a list of subreddits to choose from. Clicking the search area lets you type in a search term. Clicking the heart brings you to the favorites list.
  - Display an empty heart if the post is not a favorite, and turn it into a solid heart if the user clicks the empty heart to add the post to their favorite list. Each click toggles the state, and you should see the icon change.
  - When you refresh the posts for a subreddit, that should not change your favorites.
  - When you are in the favorites list or the subreddit list, clicking the favorite icon or the titleBar title should do nothing.
- **OnePost.** When the user clicks on the title of a post, they should see a new activity that just displays the single post. I will call this the OnePost view, and your activity should be named OnePost.
  - You can get back from the OnePost view to the list of posts using the system back button and by the "back" arrow in the upper left.
  - When you go back from the OnePost view, do NOT refresh the listing (so do NOT make a network request).
  - The OnePost view should display the text (`selfText`) in a scrollable environment, because it can be long.
  - The post's title should fit in the title bar. If it is too long, it should end with an ellipsis (...).
- **Favorites list.** When the user clicks the heart in the action bar, they are taken to the favorites list.
  - Change the title to "Favorites".
  - Favorites do NOT have to last between sessions, you never need to write them to persistent storage. But they do persist when you refetch over the network. They also stay the same if you change subreddits. Note how `equals` is defined for a RedditPost. That will make implementing favorites easier.
  - The user can click the heart of an item in the favorites list and then that item should disappear from the list. This behavior should happen even if the user is in the middle of searching the favorites list.
  - When you go back to the PostRow view using the system back button, do not refetch the list from the network, but do redraw the list because the favorite status of posts can change in the favorites list view.

- **Subreddit list.** When you click the current subreddit, you get a list of subreddits. These should be the **popular** subreddits `https://www.reddit.com/dev/api/#GET_subreddits_popular`. Get 100 of them.
  - The title should say, "Pick."
  - This list will look a bit different from the list of posts, but it is still displayed as a recyclerview of RedditPosts. It will have its own adapter (and its own fragment/view).
  - If you exit the subreddit list with the system back button, do NOT refresh your listing from the network. But please restore the current subreddit title.
  - When the user clicks a heading, this list disappears, we are back to the PostRow view, the title is updated with the new subreddit name. We fetch the new subreddit contents from the network, but it often takes a beat for the new contents to arrive. That delay is fine.
- **Search.** In both the PostRow view of a subreddit, the favorite's list, and in the subreddit listing view, the search box provides incremental, ignoreCase filtering for both the title and the subtitle text. The first match found is colored blue.
  - As you type characters, the list is restricted to the entries with text that matches. You need to highlight the matching text.
  - As you delete characters, the list expands, returning to its original state when the search box is empty. In this sense search is non-destructive. You always "have" the entire list that you fetched over the network.
  - When the search string is empty, you must dismiss the soft keyboard.
  - Search works in all recycler views.
  - Searching does not create new network requests. Reddit provides a search API that we are not using.
  - For a regular post search text in `title` and `selfText`.
  - For a subreddit listing search test in `displayName` and `publicDescription`.

Let's start by describing code given to you that you do not need to change.

- **MainActivity.kt** This file has the `hideKeyboard` method, which you will call for part of the search functionality. We set up the custom action bar so you don't have to. It also starts the HomeFragment and attaches it to the layout, without adding to the backstack.

- **glide/AppGlideModule.kt** This file lets you fetch images via glide. Check out `applyOptions` where you can set the log level if you want more output from Glide. I find Glide very chatty, so I want less output by default.

  Also note that Glide fails to load many images, especially when given a video. In general, reddit returns some sketchy URLs for images and it isn't "wrong" for Glide to fail to load them. We call glide with a regular URL and a backup thumbnail precisely because the main URL (`imageURL`) is often busted (it can be a video or a web page, neither of which Glide handles).

  When we are fetching the subreddit icon, you can use the icon as both the `urlString` and the `thumbnailURL`.

  Note that we find the size of the user's display and pass that information to Glide, so it can more efficiently manage memory. Large pictures take up a lot of memory and storage and we don't need images that have greater resolution than our display.

  Also note that I have to "adjust" the HTML escape characters that reddit puts in these URLs by calling `fromHtml`.

  Don't change this file because we are going to replace it when we test your code.

Let's talk about the networking code. Much of this is given to you, but just like the flipped classroom, you need to fill in the key parts. You should start here because it will be tough to do anything in your project without data from the network. All of these steps should be familiar from flipped classroom activities.

- **api/RedditApi.kt** Just like for CatNet and TriviaGame, you need to tell Retrofit how to fetch posts for a particular subreddit and how to fetch the list of subreddits. For posts, you can hardcode the limit of 100 posts, but you do have to pass a parameter that identifies the current subreddit. I talk about it a bit in the code comments, but don't be afraid to google around a bit.

  I have given you the types you need to "decode" Reddit's responses. Go and read this documentation `https://www.reddit.com/dev/api/#listings`, which is what reddit returns. Also read `https://www.reddit.com/dev/api/#GET_hot`

- **api/RedditPost.kt** This is your model. I talked about it in the intro. It has a bunch of fields, some of which you won't use, some of which are only relevant for the subreddit list, etc.

  We provide most of the search/text highlighting functionality, but leave out the implementation of `searchFor`. The pieces are in the file, you just need to put them together in this function.

- **api/RedditPostRepository.kt** This is pretty simple, it provides the bridge between your application code and the web API. We include debugging code that is important to us and you should not change. You only need to fill in the `// XXX Write me` lines in this file.

  The `unpackPosts` function takes a listing (what is returned from the Reddit web service) and does some work to turn it into a `List<RedditPost>`.

These files you have to write all or most of. Figure out the quickest way to start, were you connect to reddit and you can display something. Build it up from there.

I have some `// XXX` in the code, but there are no guarantees for these files.

- **AndroidManifest.xml** This does not have the right permissions and it does not list all activities. Read the documentation for `android:parentActivityName`.

- **ui/OnePost.kt** This is an activity that shows a single post. It should have a title bar, as shown in the video and described above. The back button should work (you might need Mr. Google and stack overflow to help with that one).

  We give you nothing. Create the file (called OnePost.kt) and write it! Ours is 53 lines.

  Even better, we don't give you the layout file. The layout file isn't totally trivial, but it doesn't have any FrameLayouts or anything like that. The title bar should be of type `androidx.appcompat.widget.Toolbar`. It should have a child that is a TextView with `android:textAppearance` equal to `?android:attr/textAppearanceMedium` and padding of 10dp. Include a scrollable section for selfText and an ImageView for an image.

  If the title less than or equal to 30 characters, then make it the title to the OnePost view, otherwise take the first 30 characters and add "..." for the title. You can do this by logic in your code, or you can use the `android:ellipsize` directive in your layout.

- **ui/HomeFragment.kt** Home base for your app. I left in a bunch of auxiliary function names that I used. You don't need to use them, but they are evocative of the kind of functionality you have to implement.

  Bear in mind that the callbacks you set on the title bar will remain in effect even if you swap in another fragment.

  Your HomeFragment will interact with your viewmodel, and I'm leaving that up to you. Conceptually, the interaction with the viewmodel and the views is the most difficult. Think about what livedata your viewmodel should manage and who should be observing it.

  I will say that I split out the title and the current subreddit. The action bar has to manage the current title, which is either "Pick," "Favorites," or a subreddit (that starts with "r/"). I split these up because when you change subreddits, you need to fetch a new list of posts over the network, and you need to update the title. But when you go from the favorites list back to the list of reddit posts, you just need to update the title.

  Just like in the flipped classroom, you need to manage the swiperefreshlayout.

  fragment_home.xml is the layout, which should be obvious.

  Your file has 68 lines, mine has 180 lines.

- **ui/PostRowAdapter.kt** A good old fashioned recycler list adapter. I use the **ListAdapter** base class, which exports the `submitList` function, which is easy to use when you fetch across the network. I left in the RedditDiff. You need to implement the standard adapter stuff, namely binding and a view holder.

  One caveat is contained in the comments. This adapter inherits from ListAdapter, which should mean that all we need to do is give it a new list and an old list and as clients we will never have to call `notifyDatasetChanged()`. Well, unfortunately, I can't implement equal for SpannableStrings correctly. So clients of this adapter are, under an unfortunately large set of circumstances, going to have to call `notifyDatasetChanged()`.

  One detail: in the XML for selfText, we specify elipsize, which will automatically truncate text that is too long and add an ellipsis.

  Yours 42, mine 113.

  - **row_post.xml** has the layout. The FrameLayout that contains both an ImageView and a TextView is a little wacky. Some posts have a big image, some have text. You just fill in the one you have information for and don't worry about it.
  - Load the `imageURL` and as a backup, the smaller `thumbnailURL` using glide (see below).

- **ui/Favorites.kt** This fragment should use a PostRowAdapter. You can figure out which layout to use.

  Yours 27, mine 57.

- **ui/subreddits/Subreddits.kt** This fragment is almost identical to Favorites.kt. It is more trouble than it is worth to merge them and use subtyping. You can figure out which layout to use.

- **ui/subreddits/SubredditListAdapter.kt** Use the iconURL for the `subRowPic`, the `displayName` for the `subRowHeading`, and the `publicDescription` for the `subRowDetails`.

  There is a comment in the code about how to quit the controlling fragment when a subreddit is chosen. I could have done this via a function passed in to the constructor, but it seemed cleaner to do what the code says. Tough call.

  row_subreddit.xml has the layout.

- **ui/MainViewModel.kt** Well, I saved the best for last. As you might be picking up, the viewmodels simplify your life in the fragment, but they are a bit tricky to design and write.

  Remember how to fetch over the network. Use `viewModelScope`.

  As we have seen, `MediatorLiveData` is your friend. You can use `Transform` or a few other tricks, but `MediatorLiveData` is sufficient.

  I left in a little bit about the title, by which I mean the title in the action bar. I left it in because I had to leak a little bit of view information into the viewmodel and I didn't want you to have to "read my mind" about how to handle this case.

  The favorites list persists across nework refreshes, so be sure to test that behavior in your app.

  Finally, you might wonder why I have a `doOnePost` function in my ViewModel companion object. This is pure convenience. When it comes time to launch a OnePost activity, I'm in some code that makes it difficult. Rather can pass in callbacks, I just call MainViewModel.doOnePost. Is it strictly logical? Probably not, but it makes the code easier to write.

  I'd like to talk about search, because implementing search is a bit tricky. I would advise getting pretty far into your implementation before adding it.

  Conceptually, the searchTerm live data is just another source for Mediator live data (whether that be subreddit posts, a list of subreddits or your favorites). That is how I implement it, but for reasons that I can't explain, I add the searchTerm source for regular posts when I make that Mediator live data. But for subreddits and favorites, I create the live data variable them with one source, then add the searchTerm source in the viewmodel constructor. I include a comment in the code that mentions this. I'm not completely sure why it is necessary, but I'm tipping you off that it is necessary.

  **Hint**. I have separate live data objects for e.g., the list of posts I received over the network (which I call `netPosts`) and the list of posts I display to the user after search filtering (which I call `searchPosts`).

  Yours 58, mine 195.

This assignment should be submitted through github according to standard proceedures. Please fill in the README.

**Possible plan.** This plan is really just a suggestion. If you have already blown past some of these tasks, good for you.

1. Start with retrieving 100 posts from aww. Even if you have to hardcode the subreddit, get this working first.

2. Once you have a listing, do `OnePost`. I hope that will be "easy."

3. Then tackle subreddit listings and choosing a new subreddit.

Only after you have completed these three tasks should you even start to think about favorites, search, and more advanced features like proper handling of the action bar. I think I would do favorites before search.

I am a big believer in trying to get something in your code working as soon as possible. Then build from that piece of functionality to the next. Don't get distracted by more complex tasks that wait for you in the future. Once you get experience with the basic functionality, it will be easier to understand how to implement the more complex functionality.