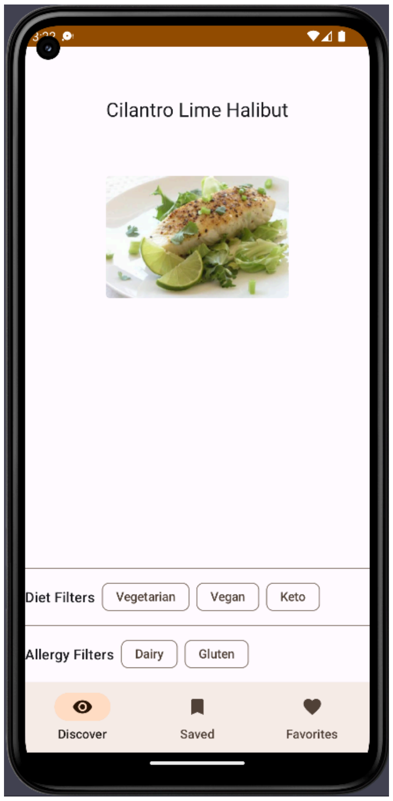
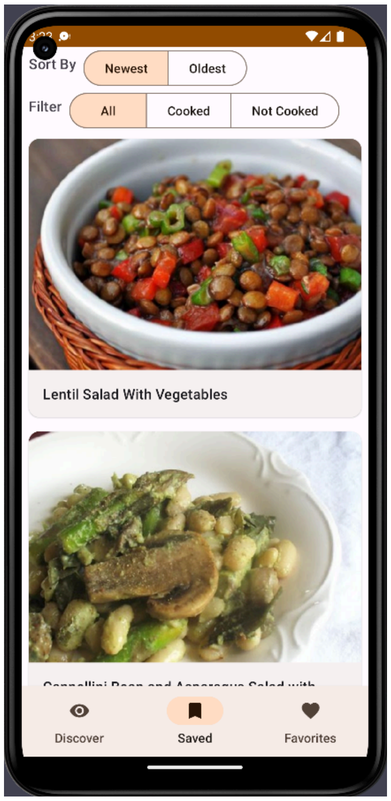
MyRecipe

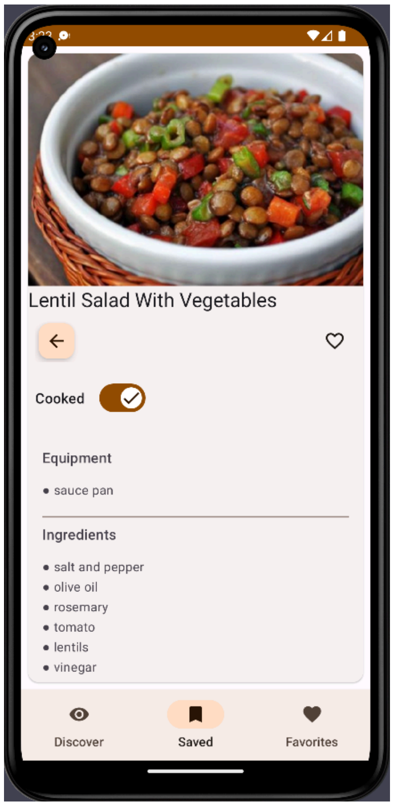
A food recipe discovery and management app

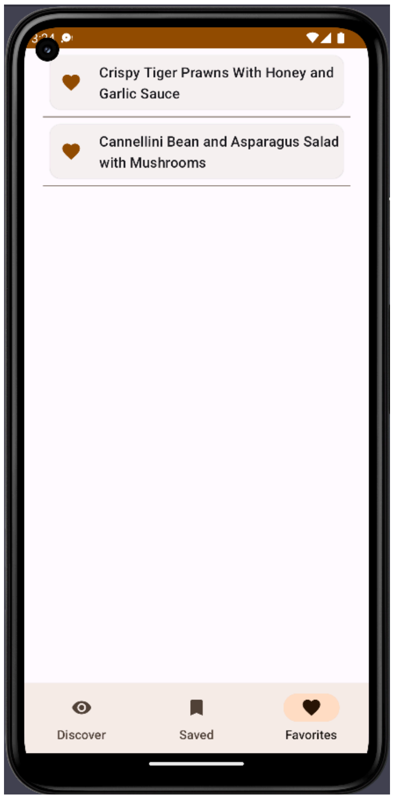
Matthew Hibshman
mah8693

Screenshots








Crispy Tiger Prawns With Honey and Garlic Sauce
Cannellini Bean and Asparagus Salad with Mushrooms
Discover
Saved
Favorites

Android APIs

- LiveData
- RecyclerViews
- Fragments
- GestureDetector

Third party Libraries

- Retrofit: HTTP/S utility used to get data over network. Connects Android application to backend hosted on Google Cloud functions. Easy to get working but error handling through a try-catch mechanism became fairly clunky and spread throughout the codebase. I refactored a good amount of it into a small "framework" used throughout the codebase (see MyRecipeApi.kt companion object), but with more time would like to clean this up through functional paradigms, perhaps using Reactive Streams which RetroFit and Android work with natively.
- Glide: fetches images over the network via HTTP/S and caches locally. Used for retrieving all images displayed in the application. Extremely easy to import and setup, I had zero issues using Glide.
- Lottie: lightweight JSON-based animations. Plays cute little animations when users are swiping while discovering new recipes. While the documentation was a little sparse and its presence on StackOverflow left more to be desired, overall Lottie's easy setup and simple JSON-based animations were enough to get what I needed out of it.

Third party Services

- Firebase Authentication: provides user management with login via email and password. Setup exactly as done previously in the course, very nice to provide user content and not have to worry about sensitive password management.
- Firestore: user data management in a NoSQL DB hosted on Firebase. Again, setup exactly as done previously in the course. I actually have the logic for interaction with Firestore in JavaScript code hosted on Cloud Functions (see below). The code itself was very easy thanks

to the great documentation and plethora of examples provided, and the schema-on-read capability of NoSQL provided flexibility for quick changes to the data model while developing the app.

- Cloud Functions: lightweight JS functions hosted server-free in the Google Cloud, used to expose all data to the Android application. Firebase provides excellent tooling including local emulation for quick iteration and simple one-command deploy that made working with Cloud Functions by far the easiest part of this project.
- Spoonacular Recipe API: third-party API exposed through HTTPS endpoints. Used to get all Recipe data for the app.

Noteworthy frontend implementation details

I went with a single activity, multiple fragment application. I found that this made it easy to navigate across the app using a Navigation Bar. I actually wrapped a FragmentManager into a global singleton that was shared across the fragments in the application (see NavigationUtil.kt). I'm not sure if this is a supported pattern or just a well-intentioned hack but it nicely kept all the navigation logic in one place.

I also tried to use Material components throughout the app. I have used Material in web applications before and as a backend-focused developer I find it very useful in making a reasonably attractive user interface without too much design effort. I found the documentation for Android very easy to use and used the Material color designer (see Resources section at bottom of report) to easily import a color theme into my app.

Noteworthy backend implementation details

I used Google Cloud Functions as a service layer for exposing data to the Android application (code in index.js under *functions* directory). I retrieved data from two different sources, a third-party API and Firestore. Keeping those integrations all in once place was nice, plus it simplified the Android code which just used Retrofit to query the Cloud Functions via HTTPS.

Also, not enough can be said about the worry-free serverless management provided by Firebase, all at a total cost of $0 for this project.

Most Interesting

The tooling provided by Firebase really blew me away. I was able to start up a working Firestore instance and emulated Cloud Functions (with hot-reload for near instantaneous testing) on my local machine with one command for installation and one command to start everything. I was also able to connect to these services running on my local machine through my Android application running on the Android Studio Emulator on my local machine, which happened in about 10 minutes after a quick Google search to get the networking right. Having a full safe-to-tinker-with production-like development environment can be life-saving in real-world application development. I have worked in situations where running a whole application front-to-back can be difficult if not impossible, and where differences between local machines and actual deployed environments can cause major headaches, so finding it so easy to connect everything locally was a huge relief.

Most Difficult

I implemented a piece of swiping functionality over an ImageView in my app and was surprised at how difficult it was to get it working. I found a solution through a third-party Android tutorial that had what I needed, but was surprised Android didn't provide better support or out-of-the-box utilities for what seems like a commonly used behavior in mobile apps. Maybe I just did not look hard enough but I would think there would be some library for putting a listener on a component that exposes a simple "onSwipeRight" functionality.

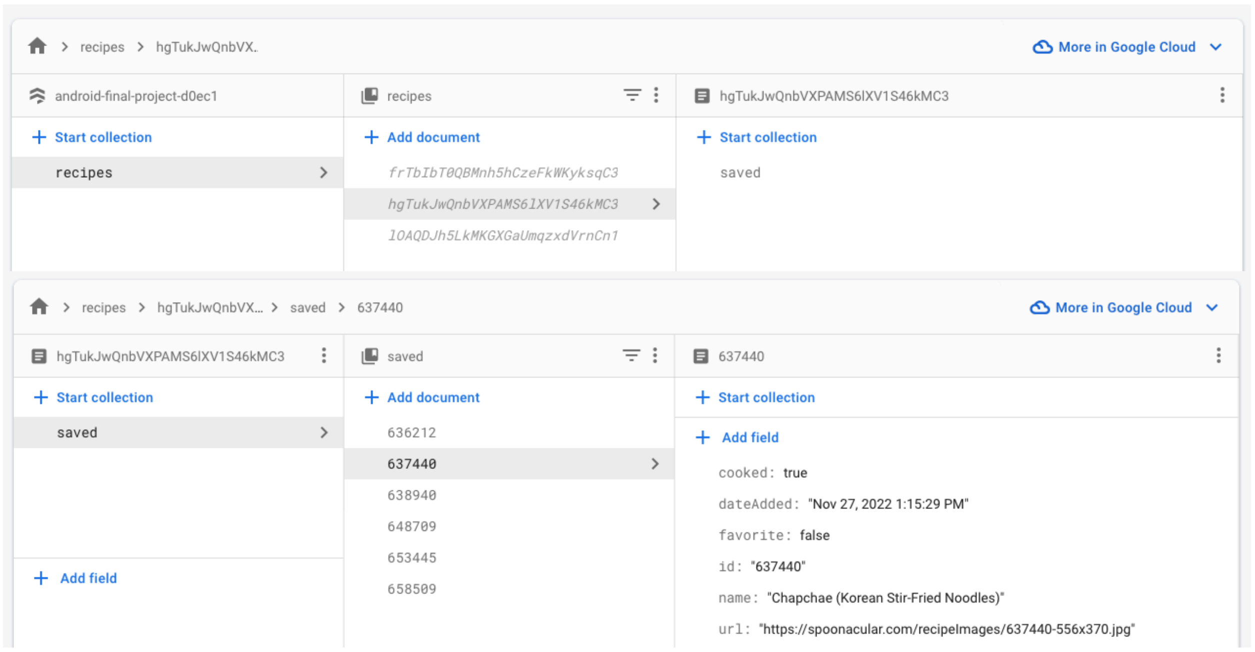
It was also amazing how difficult it was to snap to the top of a RecyclerView, which I do when the list provided to the RV changes (in my case is sorted or filtered). It probably only took an hour or hour and a half to get it working, and the final solution was very simple, but the ratio of functional-complexity to time-spent was way off for this.

How To Run

The Android application lives under the *app* folder. Import this into Android Studio and run as provided. This will run with the backend service hosted on Google Firebase. Feel free to use the app with username demo@demo.com, password test1234 for a user with already populated data, or create your own user and explore!

If interested, you can also run the backend Cloud Functions and Firestore locally. You will need to install a few tools provided by Firebase, which require Java and NPM. You also need to switch a flag in the Android code to use the endpoints on your local machine. Full details are in the project README.

Firestore Schema



```
recipes
frTbIbT0QBMnh5hCzeFkWKyksqC3
hgTukJwQnbVXPAMS6lXV1S46kMC3
lOAQDJh5LkMKGXGaUmqzxdVrnCn1
saved
```

```
saved

636212
637440
638940
648709
653445
658509

cooked: true
dateAdded: "Nov 27, 2022 1:15:29 PM"
favorite: false
id: "637440"
name: "Chapchae (Korean Stir-Fried Noodles)"
url: "https://spoonacular.com/recipeImages/637440-556x370.jpg"
```

Recipes are stored for each user by ID. Each user ID maps to a saved collection consisting of recipe IDs that map to the data for that recipe.

Line Count

Total: taken from application files, counts provided by cloc

Language	Files	Lines
Kotlin	30	1051
XML	29	919
JavaScript	1	172
Total	60	2142

Contributed: estimated by taking ~60% of Kotlin lines, ~50% of XML, and Javascript trimmed of imports and function signatures taken directly from Google Cloud Function documentation

Language	Files	Lines
Kotlin	30	600
XML	29	450
JavaScript	1	150
Total	60	1200

Resources

- Google Firebase: https://firebase.google.com/docs?authuser=0
    - Cloud Functions: https://firebase.google.com/docs/functions/get-started?authuser=0#review_complete_sample_code
    - Firestore: https://firebase.google.com/docs/firestore?authuser=0
    - Authentication: https://firebase.google.com/docs/auth/android/firebaseui?authuser=0
    - Local Emulation: https://firebase.google.com/docs/emulator-suite
- Spoonacular Recipe API: https://spoonacular.com/food-api/docs
- Material Design: https://m3.material.io/develop/android/mdc-android
    - Android components: https://github.com/material-components/material-components-android/blob/master/docs/getting-started.md
    - Color theme builder: https://m3.material.io/theme-builder#/custom

- Lottie: http://airbnb.io/lottie/#/android
  - Demo code: https://github.com/chetdeva/LottieDemo
- Connecting local Android Emulator to localhost: https://medium.com/livefront/how-to-connect-your-android-emulator-to-a-local-web-service-47c380bff350
- Swipe detection: https://www.tutorialspoint.com/how-to-detect-swipe-direction-between-left-right-and-up-down-in-android
- cloc (http://cloc.sourceforge.net/)