GourMate on Mobile works as intended. It loads a list of recipes from the web backend and displays the title, picture, and prep and cook times in a list for selection. Users can long press on recipes to either edit the recipe or delete it. The “+” floating action button opens the “Add recipe” activity where users can enter in a new recipe and upload it to the backend. Selecting a recipe from the list opens a new activity that displays all data from the recipe in a single ListView. This proof of concept for multiple different rows in a single ListView worked incredibly well so it stayed in the final design. A custom class View had to be created that extends ImageView in order to make it stretch small images to fill the screen. The ListView is given a transparent header the size of the ImageView so that the image is visible through it and so that the ListView can be scrolled over top of the image. The ListView has four different types of rows: a recipe header, an ingredient, a step, and a separator to help in spacing. Each row type extends an interface with a method that returns the type of the row so that it can be properly drawn.

Within the ViewRecipeActivity is a class that handles transferring data from mobile to Wear. It holds the recipe as a set of ordered strings that can be interpreted by the app on Wear. To make the wear app more user-friendly, a message would need to be set from mobile to Wear to instruct the watch to start the app in order to receive the data. In its present implementation, data transfer requires the app to have already been started by the user.

**Android Wear Tips**

All of Wear is asynchronous. Everything has to be inside an event listener if it depends on anything outside of its own function. This is especially obvious for methods called on Views, which cannot be called inside OnCreate. Code is just scheduling tasks to be executed whenever the phone’s processor is free.

Adb over wifi on mobile seems to break adb over Bluetooth on Wear. I was never able to program or debug on the watch via wireless connection to mobile.

Wear API is still changing really fast. Functions used in v21 became deprecated in v22 but were still autopopulated inside Android Studio. Since a lot of the additional features are pretty helpful, it’s probably best to suffer through updating to the newest versions.

Syncing Wear over wifi does NOT break Bluetooth adb. If you think that’s your problem, it’s not.

Messages are for events and triggers. DataMapRequests are for data.

It would be better to plan a project that is not dependent on the mobile side to get off the ground. It is pretty different from mobile Android and takes a surprisingly long time to get used to.

Wear wants to take care of Material Design elements on its own, especially if you use built-in layouts.

The Moto 360 takes around 3 minutes copy an app and 5 to install even a minimal app. To keep the watch from dying, enable “Stay awake while charging” in Developer Options so that it can be charging while apps install. Otherwise the battery may only last an hour during testing.