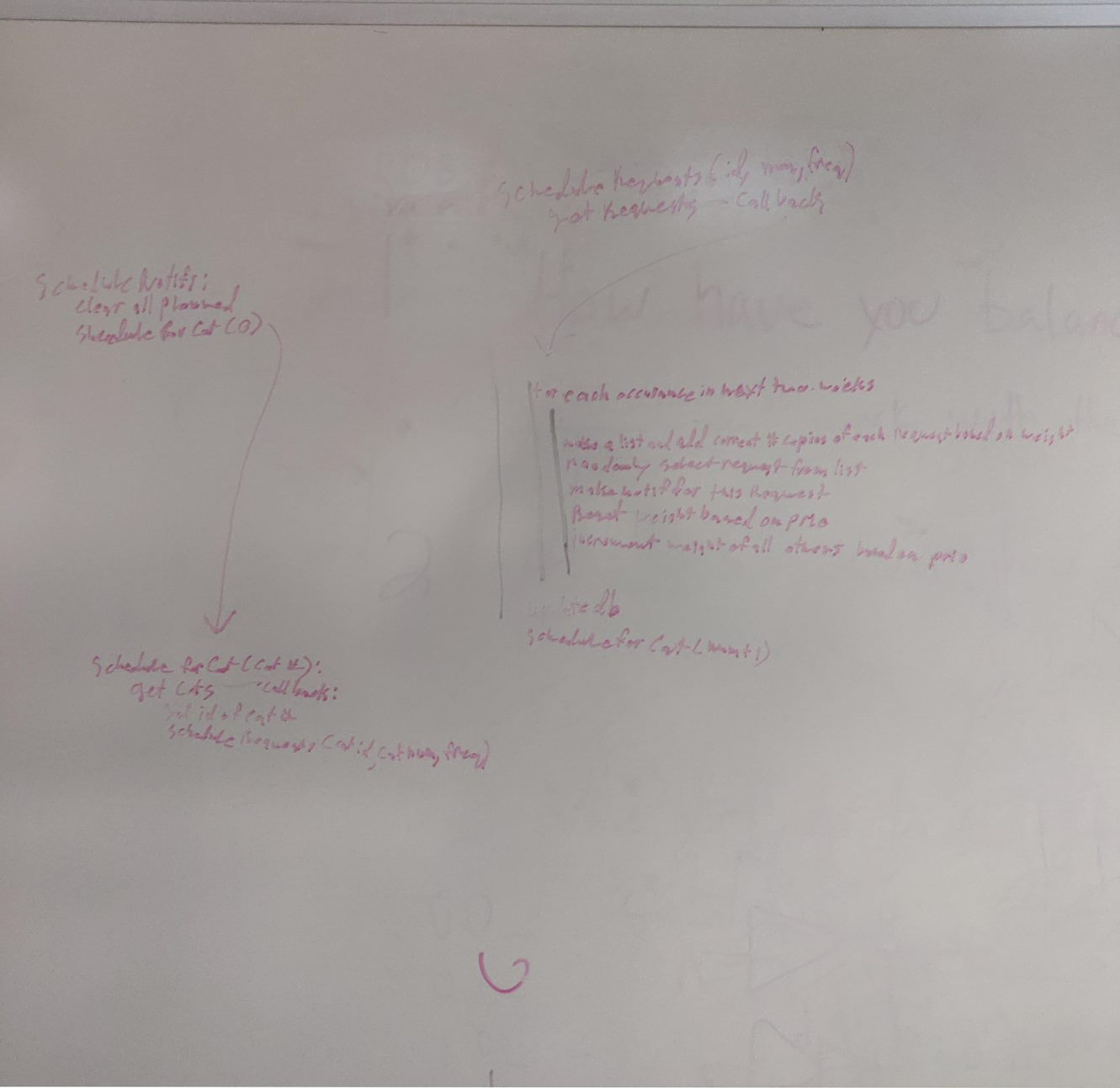
**Intro**

Welcome to my brain dump! (Un)Fortunately, my brain is small, so there really isn’t too much to unload but here it is!

**Scheduler**

This is the biggest thing I worked on. The two relevant files are scheduler.js from the schedule folder (kinda silly to have a whole folder for only one file but oh well that’s what I did) and passThroughQuery.js.

Here’s the main logic of the scheduler:



In case you can’t quite make that out, let me walk through that scheduler magic by explaining the scheduleNotifs() and scheduleForCat() functions.

scheduleNotifs() is simple. The scheduler’s goal is to plan out notifications for the next 14 days. When scheduleNotifs() is called, it first determines the soonest day that has not yet had notifications scheduled for it. This allows the scheduler to selectively schedule for days that have not already been scheduled for. Then, scheduleNotifs() tells scheduleForCat to schedule for the first category in the list of categories.

scheduleForCat() recursively schedules notifications for each category. A weakness with the way I implemented this is that the algorithm regets a list of the categories every time scheduleForCat() is called. The problem with that is that, if the categories come back in a different order in between calls, a category may be skipped or a category may be scheduled for twice. Maybe this vulnerability could be fixed by passing a list of the categories in between calls rather than rerunning the query and getting the list fresh every time? Anyway, scheduleForCat() gets a list of all the requests in the category and then, for each of the days that need to be scheduled for, it picks a request and schedules a reminder. Requests are picked by lottery. Each one has a weight associated with it, and that weight determines how likely the quest is to be selected. Every time a request is selected, the weight is reset. Every time a request is not selected, it’s weight is increased by its priority. As a result, unselected requests are more likely to be picked later and higher priority requests will increase in likelihood faster. Within scheduleForCat(), the algorithm essentially lines up all of the tickets that each request has (based on its respective weight) and then randomly selects a ticket from that list. The more tickets (higher weight) that a request has, the more likely it is to be chosen. When a request is chosen, a Reminder is stored in the database so that the algorithm can keep track of what is currently scheduled to pop up. To make scheduleForCat() recursive, I call scheduleForCat() on the next category once the current category is finished. With all the callbacks, this algorithm seems awkward, but it is the best way that I could think of to get around the asynchronous nature of javascript.

rescheduleNotifs() essentially just clears all notifications from the database and the schedule and then schedules all new notifications. This is called when things need to be reset i.e. after a request is archived or a reminder time is changed by the user.

passThroughQuery.js is a list of queries I created out of a special need. Most of them are modifications of regular queries from query.js, but I had to write them in a special way so that I’d be able to pass parameters through into callback functions. Without this, there are important variables that I would not be able to access within crucial parts of my algorithm. Each query currently consists of two functions (one to call the other). I tried to consolidate each into just one function, but it didn’t work :/. Maybe someone better than me will fix that some day…

Once the scheduler was finished, our advisor suggested a different style of scheduling that took some control away from the user for the sake of convenience. This is his idea: instead of requiring the user to determine what category triggers a notification at a specific time, the user would simply specify how many notifications/reminders they wanted on a specific day. Then, the app would determine what categories and requests to use. This change would require a rewrite of scheduleNotifs() and related functions. Some things that I would consider if I wrote this are:

* Possibly giving each category a weight to balance how often each is selected
* How to avoid one request coming up multiple times in a day
* Is there any good way for the user to specify what times they **don’t** want notifications to come up? This would be a quiet hours of sorts.

**QR Codes**

There’s a handy program i created to generate qr pngs from a text document of requests. The format of the text document is:

*Subject*

*Request*

*Subject*

*Request*

…

The java code is in the QRGenerator folder and is called QRToolv2.java. There is a version 1 which created a folder for each code, but it did not have the ability to encode multiple requests in one code. V2 lets you do that, but there is a limit to how much can be stored and a limit to how much should be stored in one code. At some point, the image gets really detailed and large, which makes it hard to share.

**Publishing**

1. Create an account with the Google Play Store
2. Be patient because there is a lot to wade through on your path to publishing