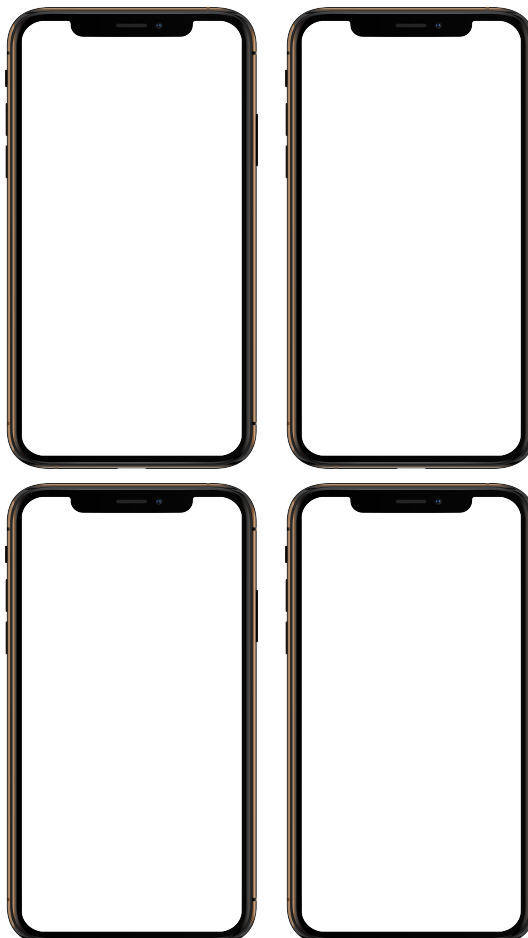## Proposal

- What problem does your app solve?
- Be as specific as possible; how does your app solve the problem?
- What is the mission statement?

Kids sometimes need time to think through and process things, especially when they've done something wrong. Often a parent will give the child a time-out as a form of discipline, but without definite goals of what it takes for the child to get back in the parent's good graces and fulfill the requirements of the discipline. This can cause more hysteria and tantrums.

The purpose of this app is to provide a clear goal for the child to reach as well as a simple distraction to allow the child to calm down enough to be able to think about what he or she has done and communicate reasonably.

This can also be used to give parents an easily enforced quiet time without making them arbitrarily decide how loud is too loud.

Features

- What features are required for your minimum viable product?
- What features may you wish to put in a future release?
- What do the top 3 similar apps do for their users?

In order to achieve a MVP, the app needs to have a timer that can be set, and be able to monitor sound levels while that timer is running. If the sound level gets too high, the timer will reset and start again.
As time permits, I hope to also do the following:
Use Firebase to implement a login and store the settings and timers that have been set up on the cloud.
Allow another device running the app and logged into the same account to receive notifications from the device that is doing the monitoring.
Implement a lock feature to pin the app so that the rest of the phone cannot be accessed without logging into phone while app is running.
Add a Push-To-Talk (PTT) button that would record a short audio clip and send it to other devices logged in.
Allow custom actions when too high of noise level is hit (e.g add a minute to the timer instead of reset. Only trigger if noise level is high for longer than x seconds)
Any not achieved at initial launch may be included in a future release.
Noise Meter & Timer for Kids does a lot of what I plan, but with no reset on high noise level and an annoying interface. Other baby monitoring or sound level apps do parts of what I plan but for different reasons and different audiences. Dormi is a baby monitoring app that I love that does monitoring well.

Model

- What data do you need?
- Where will you get the data?
- Will your data be stored locally or remotely?
- How is your data related?
- How will your data be represented in your app?
- **List all model objects with their properties and initializers.**

Data will be app settings as well as settings for each timer/monitor configuration.
If PTT is implemented, a temporary sound recording may be sent to the cloud and retrieved by another device.
All data will be created/added by the user. It will be stored in a user account on the cloud as well as locally.

Configurations will be displayed in tiles that convey most of the settings within it. (e.g. 5 minute timer with noise level set to Very Quiet).

View (Activity or Fragment)

- What activities do you need to create to meet each feature in your app?
- How will the user navigate to each activity?
- Revisit this regularly. Simplify each time. Focus on the user.
- What view elements/animations will you use to create each view?

Frameworks

- What 3rd party frameworks are you considering using?
- Do APIs require you to contact its maintainer to gain access?
- Are you required to pay to use the API?
- Have you considered using Google Play Services? (Maps, Cast, Fit?)

I intend to use Firebase to handle logins and data. I don't think I should need to use any other frameworks or APIs.

Target Audience

- Who is your target audience? Be specific.
- What feedback have you gotten from potential users?
- Have you validated the problem and your solution with your target audience? How?

My target audience would be parents of kids and to some degree the kids themselves.
I am a parent and would use this if I had it. I have not attempted to get other feedback.
My own experience has shown that my kids do better in time-out situations with visual feedback of when the time is over using a simple timer.

Monetization

- What avenues of income does your app provide?
- What features can you charge for in your app?
- Is there a possibility of a subscription model?

I would not attempt to make money off this app initially. If I get it to a point of release to a wider audience I would ask for some advice. Perhaps some unobtrusive ads could be added, with the option to remove with a purchase. Perhaps some of the cloud features planned could be available only on a premium paid version. I don't think a subscription model could be justified with the features as planned.

Classes

- What Activities and fragments do you need for your app?
- What other classes do you need for your app?
(Consider model, network, purchase, dataSource, and other specialized components)
- What will each class need to do?
- What frameworks do you need to integrate?
- **Write out properties and methods for each controller object**

Main activity: Show configurations with option to create new.
Edit activity: Allow user to edit an existing configuration or create a new one.
Run Config activity: UI for running a configuration.
Settings activity: Allow user to set/change app-level settings.
Noise monitor class: object to store settings for monitoring noise levels.
Timer class: object to manage a timer.
Config class (perhaps need a better name than this): contains a noise monitor object, timer object, and any other settings that may be needed.
Firebase Dao: manage the firebase transactions.
Config list adapter: Recycler view adapter for displaying configurations in main activity.
Perhaps classes for MySql database to store local info unless I can use Firebase's local storage options. Need to discuss.
Perhaps Config Repo and Config View Model classes for implementing MVVM unless Firebase can handle that for me.

Research

- Research thouroughly **before writing a single line of code**. Solidify the features of your app conceptually before implementation. Spend the weekend researching so you can hit the ground running on Monday.

Since we haven't done anything with recording or monitoring sound, I need to do research to figure out how to do that. I'd like to write some functioning test code before Monday.

Prototype Key Feature(s)

- This is the "bread and butter" of the app, this is what makes your app yours. Calculate how long it takes to implement these features and triple the time estimated. That way you'll have plenty of time to finish. It is preferred to drop features and spend more time working on your MVP features if needed.

The key feature is to able to monitor sound levels and reset timer when too high. Assuming I have functioning test code to monitor sound levels by Monday, I think I should be able to implement that by the end of the day Monday.

Monday
Suggested plan:
- Implement model (data object)
- Activities with mock data

Tuesday
Suggested plan:
- Connect Activities/Fragments
- Implement classes

Wednesday
Suggested plan:
- Ensure that the Key Feature(s) is/are working.
- Visual design.

Thursday + Friday Morning
Suggested plan:
- Polish visual design and features

Lambda