
GitHub Username: zacharytamas

Often

Description

Often allows users to enter tasks which repeat often in their lives, such as: brushing teeth, going to gym, drinking X ounces of water, washing the car, etc. Often allows users to keep up with these tasks, showing them a Today view which summarizes everything due to be done today. For habit formation, Often also allows users to keep track of streaks: how many consecutive occurrences of the task they completed on schedule. The desire to see the streak number for a task increase causes users to be more faithful to their new habits.

Intended User

Often is tailored towards anyone concerned with self-improvement and who like to use technology to run their lives more efficiently and less stressfully.

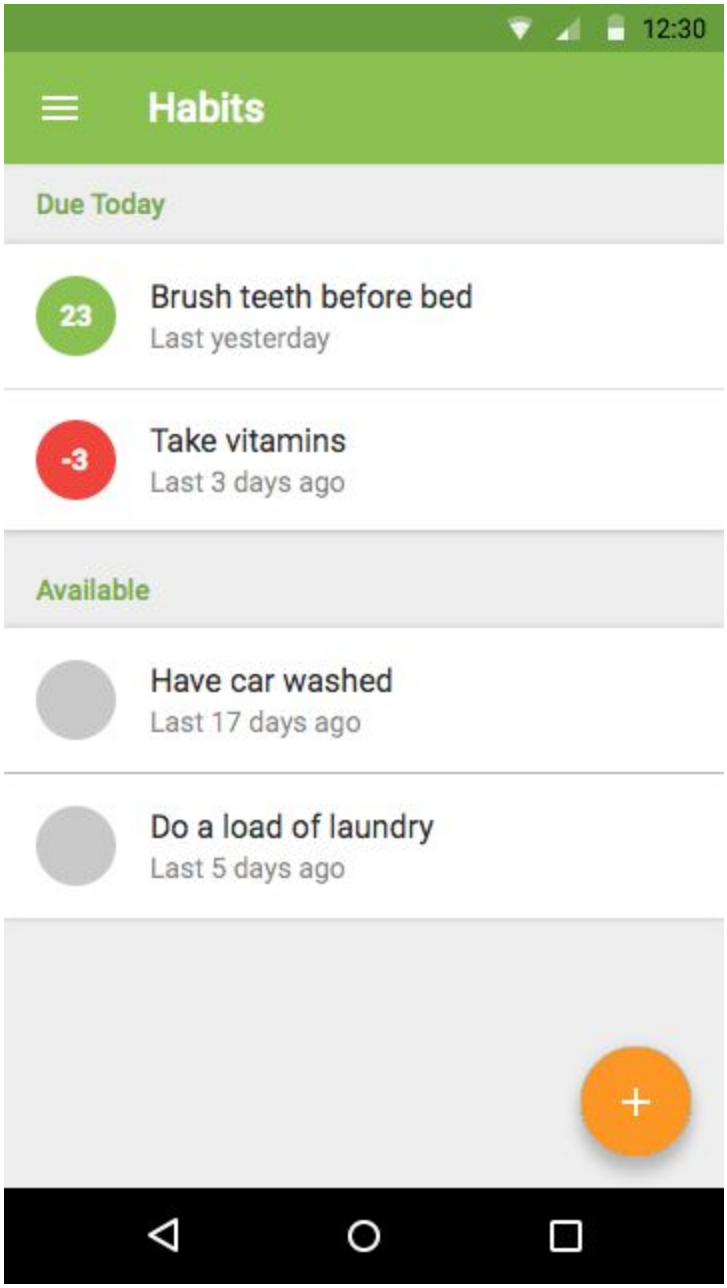
Features

- Allows users to add recurring tasks from their lives to be tracked by the application.
- Tasks can recur in a variety of manners, such as: every X days, every X weeks, or every week on Tuesday and Thursday.
- Tasks may be “required” or not. Required tasks must be done the day they are due to continue their streak. Misses will reset the streak. Tasks which aren’t required don’t use Streaks and are merely shown as Available for completion.
- At least for v1, the app does not actually store the user’s data off-device. We’ll have a server-side component which will anonymously log which tasks users add in order to present a list of top suggestions when users go to add a new Task. Common tasks will be surfaced this way and easy for the user to add and customize.

User Interface Mocks

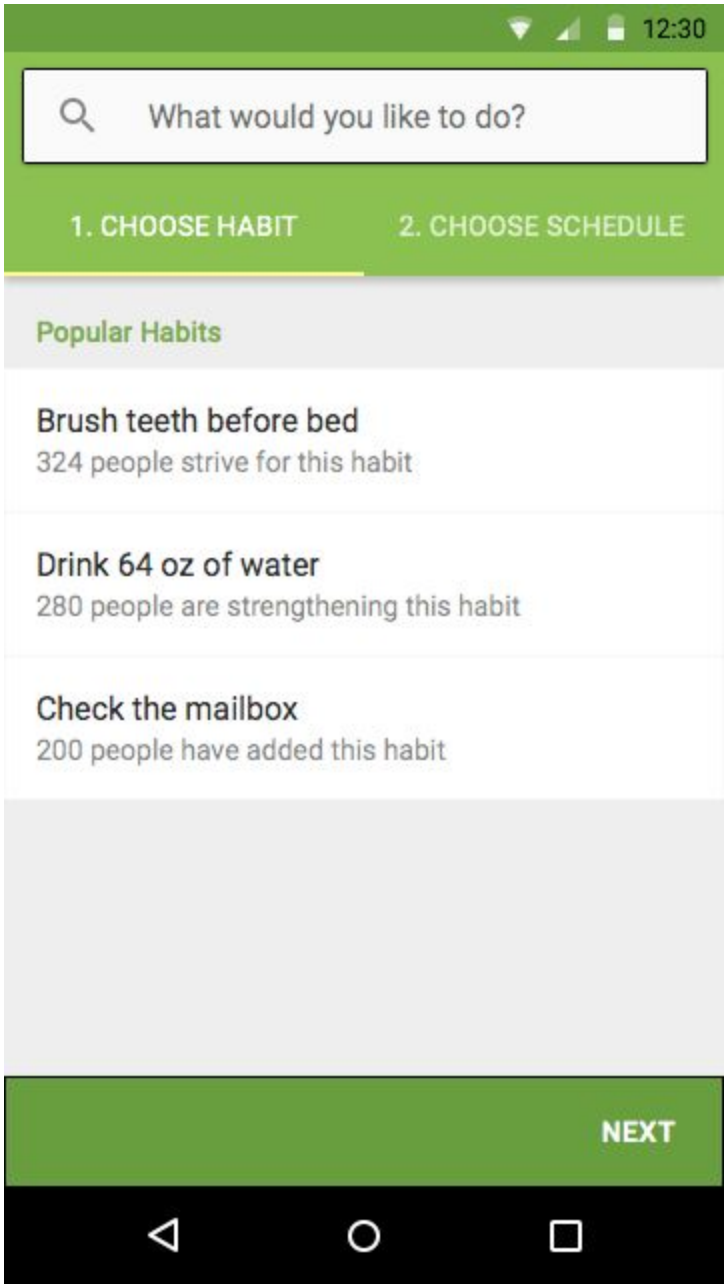
These can be created by hand (take a photo of your drawings and insert them in this flow), or using a program like Photoshop or Balsamiq.

Screen 1

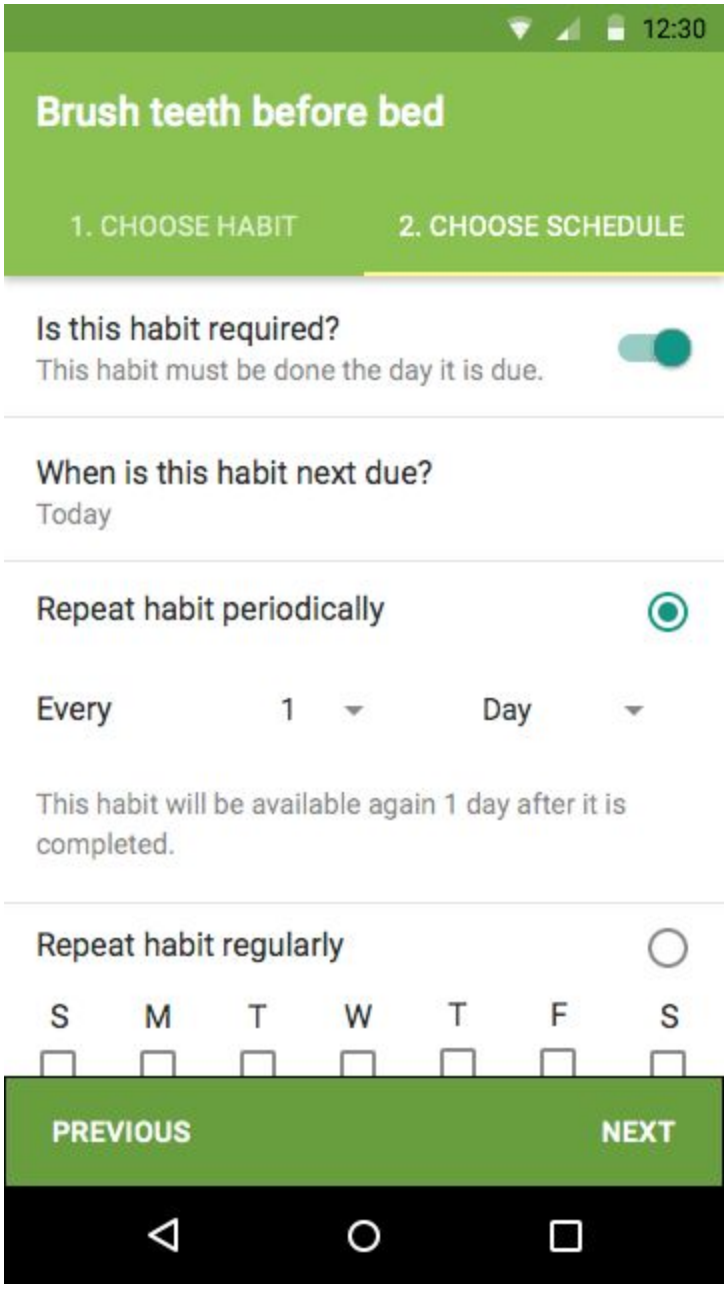


This is the main screen of the app and is where users can see what is on their schedule for the day as well as concise information about current streaks or anything that is overdue.

Screen 2



The user can add their own habits or add one of the most popular habits from other users of the app (these are compiled anonymously)



There are several features for defining how often your task repeats, such as “every X units” or “every Sunday”. Habits can be marked as “required” to enable streak tracking. Other habits, such as having your vehicle washed, may not be critical to be done the day they are due and are thus optional.

Key Considerations

How will your app handle data persistence?

I'm currently looking at using Realm.io. It has a lot of buzz about being really efficient and easier to work with. My background with databases is more ORM-based and so seeing a nicer interface for dealing with a database is definitely nice to see v.s. writing a whole bunch of SQL migration and query code. Realm.io also has an iOS SDK so in theory learning the concepts really well will help me in a cross-platform reality.

Another option I'm considering is Firebase because it would be handy to sync because I could then also make a desktop version using GitHub's Electron or just a webapp for browsers. My main worry there is cost since Firebase is a recurring cost, whereas income from this app would be pay-once meaning if sales slow it could end up not supporting itself.

The course here at Udacity had us set up a Cloud Endpoints solution with App Engine which is, I guess, the most affordably scalable solution but it seemed so complicated and I don't feel like the class taught it to me well enough to actually do something myself. Endpoints would also be easy to sync between different clients.

Minimally, I don't need to store data off-device in 1.0. But I do want to choose a data structure that will allow me to make that shift in future versions without needing to totally change the whole model layer.

Describe any corner cases in the UX.

I can't really think of any off the top of my head. My plans for 1.0 are really basic.

Describe any libraries you'll be using and share your reasoning for including them.

As stated above I may be using Realm.io, GCS for Cloud Endpoints, or Firebase.

I have an Android developer friend at Google who is all about Kotlin right now, so I may consider playing with that a bit.

I'm probably going to use Butter Knife for easy view bindings but I've read recently about a new Google-sanctioned way of doing this in the new Android Studio 1.4 so I need to research that too.

Next Steps: Required Tasks

This is the section where you can take the main features of your app (declared above) and decompose them into tangible technical tasks that you can complete incrementally until you have a finished app.

Task 1: Project Setup

First I'll need to create the project in Android Studio and get version control setup. Then lots of Gradle management: adding libraries I know I'll need, setting up the flavors I know I'll need, getting my test framework setup, etc. I suspect I'll struggle a bit here because it's been a while since I've done that.

Task 2: Design and build data layer

Then I'll need to design and build the data layer. This will include making final decisions about what tools I'm going to use for the data layer (Realm, SugarORM with SQLite, etc.) and how the actual data structure should be. I have the schema in mind already but is definitely still in a draft state).

Task 3: Build Today Activity

With the data structure ready, I can start on the home screen of the app: the Today Activity. It'll be really simple and just a ListView that shows the currently available Tasks. A Task can be slid left to complete it, which marks the Task done and then schedules a new copy in the future according to its recurrence rules. Because I won't have a UI yet to actually create Tasks, I'll probably fake it by adding mock data to the database. I'll have to create an API for doing this to use in my tests in Task 2 above anyway.

Task 4: Build Add Task flow

Next I'll need to build the Activities needed for the Adding Task flow. I've designed/mocked these out in Sketch already but they will have very intricate view interactions unlike anything I've had to do in this whole nanodegree. Lots of learning will be involved here...

Task 5: Productionizing MVP

I have several phases in mind for this app but I can't develop them all at first: partially because I want to be lean and fail fast if nobody actually wants this app, but mostly because I simply don't have the knowledge or experience to fully build what I imagine yet.

This isn't a very detailed or comprehensive plan but it's hard to explicate completely when this is my very first end-to-end, unguided Android application. I'm not entirely sure what to expect but practice makes perfect.

Submission Instructions

1. After you've completed all the sections, download this document as a PDF [File → Download as PDF]
2. Create a new GitHub repo for the capstone. Name it "**Capstone Project**"
3. Add this document to your repo. Make sure it's named "**Capstone_Stage1.pdf**"