A4: Sleep Tracker in Ionic

Due Friday, March 5th, 11:59pm PST

In this assignment, you'll demonstrate your ability to make a mobile app by developing a tool for tracking sleep and sleepiness in Ionic. There are many sleep tracking apps in the Apple App and Google Play Stores. You will implement a few of the common features.

This assignment may be completed in pairs (exactly two students enrolled in the class) or individually. You may choose your own partner or use Slack to help find one. Because the assignment features build on top of one another, it is recommended that you pair program rather than distribute the assignment features. There is no grade benefit or penalty to completing the assignment individually.

By the end of the day on Friday, February 26th, please navigate to the "People" tab on Canvas to sign up for an A4 group. Do correspond with your partner while signing up for groups. If you're the first member to join a group, please sign up for the next group that's empty. Keep a note of the Assignment Group Number that you've signed up for and add it to your readme file.

You will not be able to swap to another group after the 26th. If something happens after you sign up and you decide you're best off working alone (team conflicts, incompatible schedules, etc.), you can disband your group and let the course staff know. However, you will not be able to team up with someone else, even if they do not have a partner. You will need to complete the assignment alone.

Starter code

A starter repository is on GitHub Classroom.

Repository structure

The repository contains a large number files. The repository contains a `sleeptracker` folder with starter code for the assignment. Most of the files are generated by Ionic when creating a new (blank) project. However, a few files have been added to ease the ability to start making a sleep tracking app. **These files can be edited, and you are not required to use them**. You can also start over and create a new Ionic project from scratch by running `ionic start` on the command line. You may want to add variables and functions to these files, or create new subclasses. They are:

- Three classes in the `data` folder: `sleep-data`, `overnight-sleep-data`, and `stanford-sleepiness-data`. These files provide some data structures for logging sleep data, with the latter two being subclasses of the first. Subclassing enables the two types of logged data to be displayed together (or not, as you see fit for your design).
- One service in the `services` folder: `sleep.service`. This service provides static variables for storing the logged data and a method for loading default (fake) data.

As with A3, you can ignore the `.spec.ts` files for the purposes of this class.

You will likely want to create additional components, pages, or services to build your app. Running `ionic generate [filename]` will generate skeleton code for these files (documentation here). The decision on how many pages, components, etc. to add and what they contain is up to you.

There is also a `readme.txt` file in the root folder. **The readme is more important to this assignment than to previous ones**, as it is required to include justifications of your design choices. You will also need to create a screen or video recording of your app, `demo.mp4` (or some other reasonable file format such as `.mov` or `.avi`).

Setting up your Workspace

Some of the packages/libraries depend on newer versions of Node JS.

Check what version of Node JS you have installed with `node --version`. If you installed Node JS before this class, make sure you've updated to at least 10, such as with nvm or downloading the latest version from the node website. If your version of Node worked for A3, you should not need to update. But it's worth checking the version number if you're encountering issues.

To run the Ionic app, you will need to install Ionic through npm. To do this, you will install the Ionic Command Line Interface (CLI) globally with `npm install -g @ionic/cli`. The Ionic project is already set up in the `sleeptracker` folder, but the CLI is necessary to run the project.

You will also need to install the dependencies for the Ionic app. These dependencies are defined in each project's respective `package.json` files. `cd` into the sleeptracker folder and install the dependencies with `npm install`.

When installing, if you run into issues which look like "permission denied, try changing the permissions of the node_modules directory. If you are still having trouble, ask on Slack or talk to the course staff. It's important to try to get the setup working sooner rather than later, even if you do not plan on working on the assignment until close to the deadline.

Running the App

To run the Ionic app, `cd` into the sleeptracker folder and run `ionic lab`. This will start the app at `localhost:8200`. `ionic serve` will also run the app in the browser, but `lab` is recommended because Ionic will then replicate what the app would look like on iOS and Android. Be sure the dependencies have been installed first via `npm install`.

Any code changes will be automatically reloaded.

Working in pairs

If you are completing the assignment in pairs, you should work in one shared GitHub repository. The easiest way to do this is to have one of you create the repository and add the other as a collaborator.

The downside of working in pairs is that you open yourself up to potential issues merging one person's code into another. Make sure you always update the version of the code on your computer prior to beginning work. This guide offers some suggestions for how to resolve potential conflicts which might arise. But more generally, we suggest that you complete this assignment via pair programming, alternating roles as drivers and navigators.

Using another framework

As with the previous assignment, it is an option to use another framework such as React Native to complete this assignment in order to pick up a new skill. Similarly, it is an option to use the bindings that Ionic provides for React or Vue. **While these are potential options, you are required to get approval from the course staff prior to pursuing this option.** We expect to only allow grant approval for students who have used Angular, React, or another framework already.

Should you choose to use another framework, it is done with the understanding that the course staff will not support you in learning the framework or debugging your code. The course staff will not be more lenient in grading or give any extra credit for using something not covered in the course. You are required to complete the same assignment, which will likely mean adapting the provided starter code to the new framework. Should you choose this path, **you are on your own**. Godspeed!

Please include in your readme.txt any information we would need in order to run your code.

Requirements

Your sleep tracker app is required to have the following features:

- The ability to log overnight sleep.
- The ability to log sleepiness during the day.
- The ability to view these two categories of logged data.
- One of the following:
    - Using a native resource of the device.
    - Storing data locally or in a database.

It is up to you to decide exactly how your app implements these features. You will be evaluated on inclusion of these features as well as how well your app follows good principles of mobile design.

Log Overnight Sleep

Your app must allow someone to log the sleep they had overnight. At minimum, this means allowing a person to log what time they went to bed and when they woke up. It's up to you to decide how to support this logging, with the goal of aligning with good principles of mobile design. As you go about deciding how to support logging, think about how someone might use your app (feel free to identify a particular user or type of user, such as college students). Do they start the log when they go to sleep, or log their full night's sleep in the morning? Will a particular input field make logging easier? Are there other things a person might want to log about their sleep?

The data file `overnight-sleep-data.ts` provides a starting point for logging this data. The fields can be changed, additional fields can be added, and helper methods can be added. So long as the questions of when someone went to bed and when they woke up can be logged and viewed in your app, you can change the fields however you wish. You can assume that a person logs overnight sleep only once per day.

All data logged will be cleared when the app is restarted (such as when a change to the code is made). This is expected and allowed for this assignment, though the "storing data" option will allow data to persist when the app is restarted.

Log Sleepiness During the Day

Your app must allow someone to log how sleepy they feel throughout the day according to the Stanford Sleepiness Scale. At minimum, this means allowing a person to log how they felt on the scale's 1-7 rating system and when they felt this way. Again, it's up to you to decide how to log this data. Consider whether there are other things a person may want to log alongside their sleepiness. A person may log sleepiness more than once a day--in the morning before and after coffee, for example. This might impact how you support implementing the scale.

Sleepiness throughout the day is often logged through Experience Sampling, a method in which someone is asked to log information about how they are feeling or what they are doing at multiple points over time. Technology can help facilitate experience sampling by providing a digital diary to log with. Technology can reduce the burden even further by sending notifications when it is time to journal.

The data file `stanford-sleepiness-data.ts` provides a starting point, but you may change the fields or add new ones as you wish.

Again, all data logged will be cleared when the app is restarted (such as when a change to the code is made). This is expected and allowed for this assignment.

View Logged Data

Your app must allow someone to view the overnight sleep and sleepiness data they have logged. These two types of data can be viewed together or separately. It is up to you to decide how the data should be presented. So long as a person's Overnight and Sleepiness data is visible in your app, you will get credit for the completion portion of this feature.

Option 1: Using a Native Device Resource

You must add a feature which uses a native device resource. The added feature must *change the app's experience for a user in a meaningful way*. It is not sufficient to read a device's manufacturer and present it back, for example, because that does not change how a user would log or view their sleep. Please contact the course staff if you have any question about what would or would not qualify as a feature.

Ionic supports a long list of native device resources. However, some of these are in Cordova, while others are in Capacitor. We recommend sticking to Official, Capacitor plugins. Ionic also supports Premier plugins designed for enterprise applications; do not look into those in this assignment. Each Ionic resource provides instruction on how it should be installed. Make sure you are using the correct version of a plugin.

Native resources require installation on your device (or an emulator, depending on the feature). Ionic provides installation guides for iOS and for Android. These installations can be tricky, so if you plan to select this option we **strongly recommend trying to install the app on your device early**. If you're having trouble, online resources could be useful to getting set up. We may be able to help in office hours, but these processes are notoriously fickle. You can also complete this option using just an emulator, if you support native resources which the emulator can provide (like notifications).

Your native device resource must go beyond reading and displaying a value to actually enhancing the experience of the app. It would, for example, be insufficient to display a "Hello, World!" toast message in response to an interaction or display the accelerometer value on screen. A few suggestions which enhance the experience of the app:

- Send notifications or vibrate the device to remind someone when it is time to journal their sleepiness (a well-studied method for reducing the burden of logging).
- Approximate how restful someone's overnight sleep is by measuring how much the accelerometer value varies while they sleep.
- Use the social sharing plugin to allow a person to text their friend or significant other with how much they slept.

Debugging native features can be challenging because console statements are not visible on a device. One option is to use bindings to display variables or debugging messages on the screen.

If you choose this option, your demo video must show your solution running on either an emulator or a native device (e.g., an iPhone or Android phone). You may leverage the browser for debugging (e.g., with `ionic serve` or `ionic lab`), but part of the goal for this option is to get experience deploying on a device. The discussion slides and recording can help assist with setup.

Option 2: Backing Up Logged Data

Initially, all data logged would be cleared when the app is restarted (such as when a change to the code is made). For this option, you must back up all data a person logs to either a Firebase database or to local storage. When a person re-opens the app, all of the data they previously logged should be visible. When a person logs a new record, it must be saved to this storage.

You may structure your storage however you like, so long as both overnight sleep and sleepiness logs can be saved and retrieved.

Capacitor Storage is likely easier than using Firebase, as Ionic has a built-in library to support it. However, we will not discuss it extensively in lecture as databases serve a wider set of use cases (e.g., sharing data between devices, persistent backup in the case of a device or app malfunction).

If you choose this option, you can develop entirely in the browser and record your demo video from the browser. You do not need to run your app on an emulator or native device, but you are welcome to do so if you would like to get experience with deployment.

Additional Firebase suggestions

With Firebase, one challenge is that Firebase expects a JSON object rather than a TypeScript class. You will need to write helper functions to export a class to a JSON object and re-import it. You may want to change the name of the collection in the starter code or create multiple collections for different types of data.

As described in lecture, set your Firebase database's permissions to be readable by everyone on the web (this is generally a bad practice, of course). You may also assume that there is only one user of the app (e.g., you do not need to support authentication).

You may use an alternate database technology with permission from the course staff, such as MongoDB or MySQL. However, the staff will not provide any support for those options. You are on your own if you choose to pursue another technology. Make sure the readme describes anything the course staff would need to know in order to run your code.

Readme

Please update your `readme.txt` with how long the assignment took, who or what online resources you consulted with, any bonus features you added, and anything else we should know. If the assignment was completed in a pair, the readme should list names/emails/ids of both students. Only one submission needs to be made, however.

The readme for this assignment is critical, as it provides you an opportunity to explain why you made your design choices for each feature. Office hours are a great opportunity to get feedback on your design choices, and your assignment will be in part evaluated on how well they align with good principles of mobile design. If you ran out of time or struggled to implement something, feel free to use the readme to describe what you

intended to implement or what you struggled wtih.

Demo

Please record a demo video (`demo.mp4` or a different file extension) which shows the demo features of your app: the ability to log overnight sleep, the ability to log sleepiness, the ability to view these two categories of data, and evidence of your native device resource or data backup. For your native device resource, you must show the use of that resource on either an emulator or a physical device. For the data backup, you can refresh your page to demonstrate data persiistence. If you need, you can annotate the timestamps within the video where you demonstrate different features in your `readme.txt`.

We would prefer a single video walking through each of the different parts of your app, but you may instead find it useful to make separate videos for different parts. If you do record multiple videos, give each a descriptive file name (like `demo_sleepiness.mp4`). This video does not need to be polished. It is intended to help us understand the interface you have designed or developed and how it operates, but it's not intended to be a marketing pitch. We expect most videos can be 1 minute or less.

Recommendations

There are a lot of different ways of fulfilling the requirements of this assignment. Though fulfilling the requirements is potentially doable with the lecture material, creating an effective mobile design will likely require reading Ionic's documentation to undertstand the different components available and how to work with them. We therefore recommend that you build up your assignment over time. Complete the requirements first with buttons, inputs, items, and lists on a single page of the app. Once the required features have been achieved, you can begin exploring modals and multiple pages.

Make sure you are looking at the documentation for **Angular**, as the syntax differs slightly for React and other platforms.

We discussed four major principles of good mobile design in lecture:

- A useful initial view
- The "uh-oh" button
- Error prevention
- Follow platform conventions(Android)(iOS)

These principles are by no means exhaustive, but provide a good foundation for thinking through the design of your app. Feel free to pick a single platform to prioritize (and specify which in your readme). Your app will also be evaluated on whether the app would impress a new user in your target audience (specified in your readme). Impressing the target audience can be either aesthetic or functional--e.g., by changing the app's styling or adding additional features. Some examples of styling would be making good use of a range of Ionic components or updating the style of widgets to create a consistent design aesthetic. Some examples of additional technical features would be the ability to edit and delete previous logs and updating storage accordingly or leveraging features from multiple native device libraries. Looking at what features other sleep tracker apps implement and how they are designed can help inform these choices.

We have not explicitly covered good principles for visual design in this class. Platform conventions offer some recommendations about visual design, such as using a concise set of colors and ensuring legible fonts. We expect that, at minimum, your app edits Ionic's global styling variables and customizes the appearance of some components.

A few other things you may encounter during development (list may grow):

- By default, all new components, services, etc. made with `ionic generate [componentname]` will be added to the app folder. The files can be moved around to other folders (such as the data folder used in the starter code). But you may have to adjust imports and routing files to import them correctly.
- If you add new pages, the module file `app.module.ts` may need to be changed. In general, any page added programmatically, such as for a Modal, needs to be added as an entryComponent.

Submitting

To submit, zip your repository and upload it to Canvas. As with A3, your project is probably quite large; please remove the `node_modules` folder prior to zipping. As discussed in lecture, the advantage of using npm is that we can reference your `package.json` and `package_lock.json` to download the same packages and libraries that you used on your computer. Any late uploads are subject to the course's late policy.

Because many people will complete this assignment in pairs, you are also required to complete a short evaluation on Canvas signifying how work was distributed. The partner evaluation is available at https://canvas.eee.uci.edu/courses/32801/quizzes/151376.

Grading

This assignment will be graded on a scale of 10 points, broken down as follows:

- The ability to log overnight sleep (1 point)
- The ability to log sleepiness during the day (1 point)
- The ability to view these two categories of logged data (1 point)
- Either using a native device resource or backing up logged data (2 points)
- Following good principles of mobile design (2 points)

- Creating a compelling app (2 points)
- A readme and demo video which explain how these features were implemented and their design rationale (1 point)

You can receive 1 point of extra credit for both using a native device resource and backing up logged data. Similar to the spirit of the assignment portion in A1, we will additionally give one point of extra credit to the most compelling apps submitted (about 10% of submissions). The maximum grade for this assignment is therefore 12/10.

Some apps which received 1 bonus point for being compelling from previous offerings: Thomas 1and 2, Marawin, and Yingyu. Apps can be compelling through developing complex technical features, being well-designed, or a mix of both.

The course staff reserves the right to redistribute, penalize, or award bonus points to one or both students in situations where work was not distributed equitably among partnerships.

In prior courses, you've been asked to follow good principles for indentation, naming variables, commenting, etc. We expect you to do the same in this course, but aim to avoid being draconian in our enforcement of these principles. Egregiously poor formatting, completely uncommented code, etc. may incur a small penalty (e.g., -1 point), but we expect this to be rarely applied.
