What is/are your name(s)? What assignment group are you?

Our names are **Emily Tran** and **Chris Yuen**. We are in **assignment group 38**.

Which framework did you select (Fitbit, Alexa, A-Frame)?

We selected the **Fitbit** framework.

Q1: How, if at all, does this framework support package and library management?

**This framework supports both package and library management, however both of them are limited compared to web or mobile development due to Fitbit constraints. Package management is supported through npm, which allows developers to install dependencies and manage any configurations. On the other hand, library management is supported through FitBit SDK— which provides built-in libraries for UI features.**

Q2: How, if at all, does this framework support principles for code separation, like Model-View-Controller? Would the separation principles effectively support creation of a larger application? Why or why not?

**This framework supports some principles for code separation, but not exactly like Model-View-Controller. Fitbit apps separate UI components from the app and uses messaging to communicate between the two. This means that the logic would be kept in JavaScript (app/index.js), and the UI elements would be kept in SVG (resources/index.view). The separation principles used in this framework would not effectively support creation of a larger application because it may be harder to maintain on a Fitbit. The Fitbit is limited due to how small it is, so it would be difficult in terms of scalability for a larger application because there is less storage, worse performance, and the interactions are only based on events.**

Q3: In lecture, we discussed a few design recommendations for each respective device. How, if at all, does the framework support the recommendations for that device? What recommendations are left to the developer to decide how or whether to implement?

**The Fitbit framework supports many of the design recommendations for wrist-worn wearables by providing UI components made for smaller screens. The principle of reducing input objects is supported by the event-based interactions used in the Fitbit framework. It also minimizes navigation depth and focuses on glanceable feedback since there are limited interaction options and features. Some recommendations left for the developer to implement is the “one visual thought per screen” guideline and how “some apps don’t need a watch interface”.**

Q4: What appear to be benefits to using web technologies for development on this device? Conversely, what advantages might a native framework have over the framework you used?

**One benefit to using web technologies for development on a Fitbit is that it’s easier and more familiar for web developers to transition to since they both use JavaScript and other similar components/languages. This definitely reduces the learning curve since most web developers will be familiar with JavaScript, HTML, and CSS. Another benefit is the event based interactions, which is similar to the handling of user interactions in web development. Using a native framework would allow for better performance and efficiency compared to the web-based development used on a Fitbit. Fitbit apps are more constrained by the limited resources of the Fitbit, however native frameworks allow for more device specific features.**

Q5: What did you find easy and challenging about development in this framework?

**We found the setup extremely challenging when developing in this framework. We spent over an hour trying to get everything running due to storage issues on my MacBook, which prevented us from updating Xcode tools that were needed to install Node version 14. After finally clearing enough space on my MacBook, we had to downgrade our Node version to use the Fitbit CLI and simulator— which required a bit of adjusting to use. However, the actual coding experience was much easier. The framework felt familiar since it is written in JavaScript, and most components were straightforward to implement. The only issue we ran into was with buttons not working initially, which we resolved by setting “pointer-events=visible”. Overall, once the setup was complete, the development process was smooth.**