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

# Kentrick Felix Kepawitono, student ID: 56462267, kkepawit@uci.edu

* Stella Adriana, student ID: 53877464, sadriana@uci.edu
* Assignment group: 22

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

Fitbit

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

The development of the Fitbit OS-compatible apps and clock faces can be done through command-line interface (CLI) or Fitbit Studio by utilizing the Fitbit OS Software Development Kit (SDK). In order to develop an app using this development kit, developers need a way to install (by running npm) and execute packages (by running npx). When developers run “npm add --dev @fitbit/sdk”—npm (Node Package Manager) is a JavaScript programming language’s package manager—within a command-line interface, it will automatically install all the required packages and libraries as dependencies needed within that intended project, stored inside the node\_modules folder, to use the Fitbit SDK. Additionally, when developers run “npm add --dev @fitbit/sdk-cli,” it will install the SDK CLI packages and libraries as dependencies needed to help develop that particular project. In order to execute/utilize this SDK, developers will need to call “npx fitbit-build”—npx (Node Package Execute) is a JavaScript programming language’s package executer to run installed packages— in order to execute all installed packages and libraries to build that particular app. That is how Fitbit framework calls and utilizes its packages and libraries as dependencies to develop apps.

# Q2: How, if at all, does this framework support principles for code separation, like Model-View-Controller? Would how this framework supports separation effectively enable creation of a larger application?

Fitbit framework supports Model-View-Controller well. From the code skeleton given in the starter code, it separates the user interface from the substance of the application. The index.js under the app folder can be seen as the model. This portion of the code holds the raw data and the essential components of the application. It gets current local time, converts time zones, etc. The view, on the other hand, is represented by the index.view and style.css. index.view allows the user to interact with the program, while style.css makes the app look nice. The controller is implemented in both index.js and index.view. This code acts as a liaison between the Model and the View, receiving user input from the View and deciding what to do with it in the Model.

Fitbit, with the support of Model-View-Controller, will definitely enable the creation of larger applications. A larger application will have more than one developer working on a project. With the support of Model-View-Controller, it is possible for a developer to work on the view while the other works on the controller. Besides, it also allows developers to modify one part without affecting the entire model (e.g. changing the user interface will not affect the Model and the Controller).

# 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?

Fitbit comes with Design Guidelines and User Interface (UI) Guidelines developers can access in order to help the development process to produce Fitbit-agreed-upon design styles and components to make the design consistent and user-friendly according to Fitbit’s standard. By having this open-source documentation pertaining to its design and UI guidelines, Fitbit helps provide guidance for developers to streamline the design process and make sure the design is consistent with their branding and style guidelines. The UI guidelines provides documentation on how to produce the intended design into a reality through the implementation of CSS, JavaScript, SVG, components, and animations, while as the design guidelines provides information on what and how the UI should look like—how many buttons should each screen has, how the margins, paddings, and spacings should look like, how the information is displayed, how the users should interact with the device in order to streamline the user experience, etc. In other words, the design guidelines documentation helps developers understand the design hierarchy and standard of Fitbit’s branding and aesthetics while as UI guidelines documentation provides information on how to code and produce the actual design with CSS, JavaScript, SVG, SVG components, and animations.

Link to the design guidelines: <https://dev.fitbit.com/build/guides/design-guidelines/>

Link to the UI guidelines: <https://dev.fitbit.com/build/guides/user-interface/>

# Q4: What did you find easy about development in this framework?

It was very easy to get started with the Fitbit framework. There is a lot of documentation and examples that Fitbit has provided in their websites. The description for each section in the documentation is short and the sample codes improve our understanding of the description. Sometimes we encountered issues while implementing some functionalities in the framework. Fortunately, Fitbit has its own forum for Fitbit developers. It helped us in resolving any issues while developing. Finally, the framework can be developed with JavaScript, CSS, and SVG. Those three languages are commonly used by developers out there; therefore, it is easy for us to get started with it.

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

We initially planned to drag and drop starter code files to Fitbit Studio and work on that. However, we found it difficult to work on the code as a team. We just don’t want to share one of our Fitbit credentials due to security reasons. Therefore, we decided to use Fitbit command line interface (CLI) to edit the starter code files, which we could push to our local repository on GitHub. Whenever I made some changes, I could simply push it so that my partner could take a look at the changes made.