HikingJunkie Document Specification

Team Members:

Diego Covarrubias

Jane (Jaeyeon Kang)

Jake (Jong Min)

Vincent

**Purpose:**

HikingJunkie was designed to provide users with a resource for all things hiking. With features such as searching and sorting, it makes it easy to find the trail you’re looking for. It also provides users with a way to expand the trails that they can access with an AddTrail feature, and they can edit existing trails with the EditTrail feature. Users can also add trails to a favorites list, so that they don’t have to go through the trouble of finding their favorite trails again.

**ViewControllers:**

**RevealViewController:**

RevealViewController is the initial view controller which provides the functionality for a slide-out menu. It uses a third party objective-c file which is free to use. It is under a folder called SWRevealViewController, which includes the .h and .m file, and Bridging header file that makes it compatible with swift.

In order to use it an sw\_frontviewController has to be set, using a custom segue, this is the first view controller that is shows when the app loads. Also an sw\_rearviewController has to be set, which is the viewController for the menu, and it also set using a custom segue, titled as such.

**MenuController:**

MenuController is the viewController for the slide-in/slide-out sidebar menu. It contains buttons which trigger custom “push” segues. The buttons represent different menu items, which in this case are Home, Favorites, and Add/Edit Trail.

**Home/FavoritesTableView:**

On startup the app populates this tableviewas “Home”, meaning it loads all of the Trails saved in Core Data. It displays the trails individually in each cell, with each cell displaying 1 image, the name, city, distance of the trail, and average time to complete. When the trail is loaded by clicking the FavoritesButton from the Sidebar menu, it loads all of the trails saved in core data again, but only populates the cells with the trails that are set as favorites. It also changes the title accordingly.

Clicking on a cell triggers a segue, which passes the data of the selected trail, to a TabBarController.

**NSManagedObject Trail:**

Trail is an NSManagedObject, which holds all of the information of a trail instance, including the filenames of the images saved under it.

**TabBarController:**

The TabBarController displays the first item in its tabs, which in this case is the InfoViewController. It also passes the data of the selected trail to all of its tab items (view controllers), in its viewDidLoad method. The three viewControllers under the TabBarController are the InfoViewController, ImagesViewController, and MapViewController.

**InfoViewController:**

The InfoViewController is the same concept as the DetailViewController from the To-Do-List App. It displays the data of the trail in an organized format. It gets passed a trail object by the TabBarController so that it can have data to work with.

**ImagesViewController:**

The ImagesViewController hosts a scrollView, which is then populated with ImageViews. It is used to display the images linked to the selectedTrail. It split the picturesString associated with the trail object, with a comma as the delimeter. It then uses that string directly, or appends it to a DocumentsDirectory path where the images are stored.

**MapViewController:**

The MapViewController displays a route for based on the information that is passed to it. By default, it tries to get the location of the user and sets the value of the FromTextField to that location. If no location is found, as is the case with the simulator, it sets it by default to San Francisco, CA, but this is only for demonstration purposes. It also sets the ToTextField automatically based on the Address and City attributes of the Trail object. It also lets the user change that data if they wish to.

Clicking on show route displays the route, and then unhides the showDirections button. Clicking on the showDirections triggers a segue to a DirectionsViewController, and passes a directions string which contains the steps separated by a comma.

**DirectionsViewController:**

The DirectionsViewController hosts a ScrollView which is populated with customized textViews. The textViews each display a step.

**SearchController:**

The SearchController has a searchbar that accept input from the user. The search is filtered by trail name. Any trail name that matches the input will be shown in the results. Pressing x when there is an input in the toolbar clears the input. Pressing cancel returns the user to the “Home” view. Clicking on any of the results segues to the InfoView of the trail.