

Architecture:

The app's overall architecture is split into two components – the story board and the view controllers. The storyboard (Main.storyboard) is a visual representation of the application. As described in the user documentation, the application is a page based application with a menu bar at the bottom. The story board consists of five menu items: map, today page, appointments, contacts, and more (resources). Each of these menu items have a corresponding story board page and controller. The controllers for the first four menu items are located in the MainMenuControllers folder. The additional menus found in more are located in ResourcesController folder.

The app uses CocoaPods in order to use outside libraries including (documentation for usage of libraries in links):

<https://github.com/Alamofire/Alamofire>

<https://github.com/Mozharovsky/CVCalendar>

Although it is unnecessary to download and install CocoaPods to run preexisting libraries, if you want to add new libraries you must install using the following instructions:

<https://guides.cocoapods.org/using/getting-started.html>

Components of Application

Here are the main components and summaries of the different components in the app:

Today's Events and Calendar

TodayEventCell.swift: Swift file containing variable references to eventImage, eventName, and eventDetails. Used as a container for information to be displayed in the cells in the Today page.

TodayViewController.swift: File controller the Today View Page. Contains a struct that holds event data information. This struct is instantiated in viewDidLoad() function. Should be replaced with database information (See Database Integration in Overall Next Step section). This class is also a UITableViewDelegate and UITableViewDataSource and as such, follows the appropriate protocols:

Important Functions:

func tableView(tableView: UITableView, didSelectRow...): Passes table information to CalenderViewController.

CalenderViewController.swift: File controller for the Calender. Uses CVCalendar and follows the appropriate protocols for CVCalendarViewDelegate and CVCalendarMenuViewDelegate (see CVCalendar documentation link). Contains a struct that holds event data information. This struct is instantiated in viewDidLoad() function. Should be replaced with database information (See Database Integration in Overall Next Step section). Displays appropriate text into views. In the future, after adding database, can change event info displayed based on day selected.

Maps Page

The Map page is contained in the MapController class. To change various aspects about the Map, use the Google Maps API as described here: <https://developers.google.com/maps/documentation/ios-sdk/reference/index>

To place a marker on the map, create a new `GMSMarker()`, and set its position, title, snippet, and icon (optional) and add it to the current mapview. You can set the initial landing view of the map by changing the camera latitude and longitude. You can toggle the angle of the map by changing the `viewingAngle`, and change the direction of the map by modifying the bearing variable. You can set various restrictions on the zooming level and visual components of the map. To add an image overlay, specify the UI image to be used. The map is using the Google Maps API, and there are also resources here: <https://developers.google.com/maps/documentation/ios-sdk/intro>

to help you get started on basic functionality of Google Maps SDK.

Appointments

The `AppointmentsTableViewController` is the main class for the appointments section of the app implements the search feature and table of providers, which are structs called "Providers." When a specific provider is selected it navigates to the `AppointmentController`, which is the detailed view of a specific doctor.

Resources

The `MoreViewController` is the main class for the appointments section of the app. It provides segues to the `CalendarViewController` and the `TableViewController`s for Rest, Education, POI, Special, Support, and Travel. It uses the "Resources" struct to define each table entry. These methods each contain tables functions of their own, which navigate to their detailed pages about each support group, class, point of interest, etc. They use a specified Cell file, such as `RestTableViewCell` to identify the item selected. The detail view of each resource is named the same as their table view function except for "Detail" instead of "Table" (i.e. `RestDetailViewController`).

Overall Next Steps:

Database Integration and Schema Location

Database schema can be found on the Dream Factory website under Schema (see Figure below)

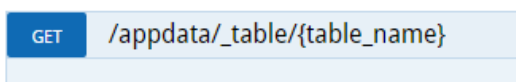
We've set up a database for you (see Technical Maintenance documentations for how to access and modify data entries) that you can make HTTP requests to grab data. Currently, the data in the app is stored as arrays of structs. See `CalendarViewController`. The next step is to input the actual data into the database (see Technical Maintenance) as well as connect make HTTP requests to retrieve the data for use in the app. For information on the schema of the database, click the Schema menu in Dream Factory (see Figure).

Instructions for how to do so are as follows:

1. Login to DreamFactory (see Technical Maintenance)
2. Click API Documentation at the top. This is a list of HTTP API calls Dream Factory has built in.

A horizontal navigation bar with several menu items: Home, Apps, Admins, Users, Roles, Services, Schema, Data, Files, Scripts, API Docs (highlighted in blue), Config, and Packages.

3. To read our table we want to use the following call:

A code snippet showing an HTTP GET request to a specific API endpoint. The text is: GET /appdata/_table/{table_name}

```
GET /appdata/_table/{table_name}
```

4. Once you generate the URL you may integrate it into the app using Alamofire (already integrated into the app). Documentation on Alamofire can be found here:

<https://github.com/Alamofire/Alamofire>

Making a Request

```
import Alamofire

Alamofire.request(.GET, "https://httpbin.org/get")
```

Response Handling

```
Alamofire.request(.GET, "https://httpbin.org/get", parameters: ["foo": "bar"])
    .responseJSON { response in
        print(response.request) // original URL request
        print(response.response) // URL response
        print(response.data) // server data
        print(response.result) // result of response serialization

        if let JSON = response.result.value {
            print("JSON: \(JSON)")
        }
    }
```

Push Notifications:

Some resources for integrating push notifications into the app (for events and wait times):

<https://developer.apple.com/notifications/>

<http://shrikar.com/ios-8-notifications-in-swift/>