FailSafe Technical Guide

Version 1.0

December 3, 2014

Contents

Databases

Our database schemas are located in the sql_files directory in the root of the FailSafe project. We use two databases for this project, with three tables total.

Database I: "directory"

Table I: "tblUser" :

UserID INT NOT NULL AUTO_INCREMENT,

Role VARCHAR(50) NOT NULL,

IsAdministrator BOOLEAN,

FirstName VARCHAR(50) NOT NULL,

LastName VARCHAR(50) NOT NULL,

CellPhone VARCHAR(14) NOT NULL,

HomePhone VARCHAR(14) NOT NULL,

PagerNumber VARCHAR(14) NOT NULL,

NetID VARCHAR(10),

PRIMARY KEY (UserID)

Database II: "calendar"

Table II: "schedule" :

Day DATE NOT NULL,

Faculty VARCHAR(50) NOT NULL,

Fellow VARCHAR(50) NOT NULL,

RN1 VARCHAR(50) NOT NULL,

RN2 VARCHAR(50) NOT NULL,

Tech1 VARCHAR(50) NOT NULL,

Tech2 VARCHAR(50) NOT NULL,

PRIMARY KEY(Day)

Table III: "substitutions" :

SubID int NOT NULL AUTO_INCREMENT,

StartTime DATETIME NOT NULL,

```
EndTime DATETIME NOT NULL,
Role VARCHAR(50),
SubName VARCHAR(50),
PRIMARY KEY(SubID)
```

Common Layout

Content that is common across all pages can be found under templates/layout.html. This html file contains information for both the actual navigation buttons (dashboard, directory, calendar, substitutions) and the activation buttons, which are responsible for activating & deactivating teams.

Navigation Bar

The navigation bar is essentially boilerplate code from Twitter Bootstrap. Twitter Bootstrap is well-documented, and so changes to our navigation bar can be easily made by following their documentation relative to the HTML at templates/layout.html

Activation Buttons

The Activation buttons are within the navigation bar, but have their own complex functionality and are crucial to the use of FailSafe. These buttons are controlled by the JavaScript at static/navbar.js (bad name), and are as follows:

Activate

The activate button sends an alert to all members of the active Call Team every 30 seconds until they respond to the Twilio server. The JavaScript code for initiating this process is located in the alertOnCall() method in navbar.js. This code initiates the call using a JavaScript interval, and thus re-pings the endpoint backend/contact every 30 seconds, which actually makes the calls through Twilio. To change this interval, simple edit the *alertFrequency* field at the top of the file to the desired number of milliseconds.

Deactivate

The deactivate button sends a single simple message to all members of a previously activated call team to inform them that the alert is *no longer* in effect, and that they should go home. This behavior is controlled by the deactivate() method in navbar.js, which calls the backend/form_team and backend/deactivate endpoints. The first consolidates members of the currently active team from the substitutions and schedule tables, and the second actually sends the messages. This button, similar to the Silence button, also breaks any alerting loops started by clicking the activate button.

Silence

The silence button simply breaks any alerting loops that have been started by clicking the activate button. All this has to do is clear every interval in the alertingIDs map in navbar.js. This behavior is done through the cancelAll() method (which is also called by the deactivate button).

Calendar View

Python Overview

The Calendar View is divided into two parts: the day view (substitutions), and the month view (call teams). All of the endpoints for the calendar view are organized under calendar_view/blueprint.py. You can find every endpoint under /calendar in this blueprint.

JavaScript Overview

All of the JavaScript code for the Calendar View is in the file calendar_view/static/eventScript.js. This is just a normal jQuery JavaScript file. All dialogs are created using the createDialog() function.

Moment.js

All of the storage of dates and times relies on the moment.js library, which allows one to easily parse and manipulate dates using Moment objects in JavaScript.

Month View vs. Day View

The month view is represented by the HTML file under calendar_view/templates/month_view.html. This includes the dialogs that pop up when one tries to create or edit call events. Similarly, the day view is represented by the HTML file at calendar_view/templates/day_view.html. The calendar itself (within the month view) is updated using the makeCalendar() function in calendar_view/static/eventScript.js. The substitutions table (within the day view) is updated using the makeDayView() function in the same file.

AJAX

Data is communicated back and forth to the Python endpoints using AJAX. The primary functions that we use in the calendar view to communicate this information are as follows:

Simple get methods: AJAXGetWrapper(endpoint)

Get methods with data: AJAXGetWithData(endpoint, requestParams)

All other HTTP methods: AJAXJSONWrapper(method, url, data)