Webmaster Manual

Tools Used

1. GoDaddy (Ask Reed Treasury for login details)
2. Firebase
3. Github

Here is a table of contents for the manual, along with a short description of each section.

Table of Contents

Structure

The "website" effectively works by maintaining and ensuring the health of two interplaying elements:

a. Website codebase

The website source files are files found in the website directory, which can be found by logging on the GoDaddy website and accessing the file directory through cPanel.

b. Firebase database

The other element is the website-specific Firebase database collections. Firebase is a service provided by Google that hosts databases which has pre-exisiting infastructure for read/write abilities on data the website utilizes.

Database collections

There are 7 primary database collections, and they are listed below along with a short functional description.

Collections Breakdown

Collections

a. elections: event-indexed collections of the past and current student body election.
b. polls: event-indexed collections of the past and current funding poll.
c. nominations: dataset array containing nominee-submitted information about each candidate.
d. users_who_have_voted: event-indexed collections of the users who have voted in the funding polls and elections.
e. signatories: semester-dependent collection of all currently approved signators.
f. registered_clubs: semester-dependent collection of all currently registered clubs.
g. fund_requests: semester-dependent collection of all the signator-submitted funding requests.

The database collections may be partioned into two primary subsets. The semester dependent subset and the non-semester dependent subset.

Semester dependent

a. signatories
b. registered_clubs
c. fund_requests

Note: Every semester dependent collection has a subcollections which are named the semester-specific strings. For continuity, please ensure the string takes the form: $[semester] = term\text{-}year$.

Example: The string name for the any semester dependent subcollection for the Fall 2020 semester is [*fall-2020*].

Event-indexed (Non-semester depdendent)

a. elections
b. polls
c. users_who_have_voted
d. nominations

Note: We can also name the non-semester depedent collections the set of event-indexed collections. The reason is because every subcollection in these collections has a name which takes the form:

event-[semester]-SHA32id

1. The event is a two letter index of whether the event is a funding poll (fp) or an SB election (se).
2. The semester is simply the string as explained above, as follows: $[semester] = term\text{-}year$.
3. Lastly, the SHA32 index is simply a random SHA32 hash that uniquely identifies every (semester, event) pair.

Example: The funding poll happening in Fall 2020 with a SHA32 ID of ABC will be: *fp-fall-2020-ABC*.

For continuity, please ensure every event-indexed collection has subcollection names which take the form: *event-[semester]-SHA32id*.

Event and collections relations

The relation between these collections and events is given by: $event \leftarrow \{collections\}$.

a. Funding Poll ← {users_who_have_voted, registered_clubs}
b. Elections ← {users_who_have_voted, nominations, elections}

Different utilities/classes on the website utilize different subsets of these collections. The utility-collection subsets pairs are given below. We say that a utility **talks** with a subset of database collections if the website codebase is programmed to have a read-write relationship between that utility and the collections in that subset.

Funding-poll

Funding-poll **talks** to the subset:

a. registered_clubs: show club information and edit the votes entry in the club array (read/write)
b. users_who_have_voted: keeps track of users who have voted in the current event. (read/write)

Election

Election-class **talks** to the subset:

a. campaigns: read in the blurbs of each candidate (read)
b. users_who_have_voted: keeps track of users who have voted in the current event. (read/write)

Dash-door

Dash-door **talks** to the subset:

a. signatories: add and view current semester's signatories (read/write)
b. registered_clubs: view current semester's clubs information (read)
c. fund_requests: view funding requests (read)
d. elections: view current campaign data (read)

Maintainence and Event Running Routines

Updating the Website (R1)

Updating the website may be tricky at first but it should be easy once the Webmaster does a few practice tries. The standard routine of updating the website should **NOT** involve directly editing the files on cPanel. Doing so will create some large issues with Github version control in the future. Instead, please use Git or install Github Desktop (more user-friendly) and have a local copy of the website directory installed. Then, the Webmaster may edit the repository and propogate the changes through Github.

Suppose the Webmaster implements changes to a local copy of the website on their device. Once the changes are done, the Webmaster should build the *app.min.js* script if *app.js* is edited. Otherwise, push the changes onto the Github Repository. Then, log on to the GoDaddy website, and find the cPanel Adminstrator Panel. Then, go to 'Github Version Control' → 'Manage' → 'Pull and Deploy'. Pressing 'Update from Remote' should update the website.

Subroutine: Building the Script

To build the *app.min.js* script, proceed to the directory containing *build.sh*. On terminal, execute the script by writing:

$$bash\ build.sh$$

Semester Changes (R2)

Before every semester begins, change the *const CONFIG* expression in *app.js* to reflect the current semester. Then, ensure every semester dependent collection has a collection pertaining to the new semester. Note that **this must be done manually** by the Webmaster(s). Recall that the semester name must follow the form:

$$[semester] = term\text{-}year$$

For reference, these are the semester dependent collections:

Semester dependent

a. signatories
b. registered_clubs
c. fund_requests

Detailed Instructions

Here are more direct instructions for the semester change routine. The process should be clearer once the Webmaster explores the collections of previous semesters. Implementation of the new semesters should not be any different.

1. In *app.js*, the main code file, find the *const CONFIG* expression and rename the semester to the current [*semester*] string.
2. In Firebase, go to each semester-dependent collection and add a new collection for that semester using the exact same [*semester*] string.
3. Then, within each collection, create an **empty** array with the same name as that master collection. For example, if you are updating *signatories*, you will create a new empty *signatories* array in the new semester collection.
4. Update the website as in Routine 1 (R1).

Dashdoor Permissions

Every semester, Reed Treasury rotates and adds new members. These new members will ask for permission to Dashdoor, the adminstrative backdoor of the website. To give those users permission, you must edit the *const CONFIG* and add the relevant emails to the 'authorised_users' array. Then, remember to build the script and update the website as in Routine 1.

Note: The funding poll and student body elections applets do not seem to be functional as of Fall 2020. Also worth noting is that the parameter $direct_server_load$ in the *const CONFIG* file seems to crash the website when set to $false$.

Running Funding Poll

Some manual steps are necessary to run a funding poll event.

1. Edit the time parameters *fpopens* and *fpcloses* in the *const CONFIG* to ensure funding poll is open and closed at the right times. These parameters are determined by the Treasury.
2. Change the *poll-name* entry in the *const CONFIG* file to reflect the current funding poll ID. Recall that it must follow the form:

   fp-[semester]-SHA32id

Running Student Body Elections

1. Edit the time parameters *seopens* and *secloses* in the *const CONFIG* to ensure that the SB elections is open and closed at the right times. These parameters are determined by the Senate.
2. Change the *secampaign* entry in the *const CONFIG* file to reflect the current senate election ID. Recall that it must follow the form:

   se-[semester]-SHA32id