The Webmaster Web Development Manual

Reed College

Spring 2021

Welcome to the Webmaster Web Development Manual. This document will assume that you have read the other manual (the Job Manual). If you haven't, please do so. Otherwise, we will now talk about everything pertaining to the Web Development side of the job. To start off, let us talk about the software/tools that are used to deploy the website.

Software/Tools Used

1. **GoDaddy:** GoDaddy is the web hosting service on which ***reedsb.com*** is hosted upon. They provide the DNS that we use and the control panel for the website is accessed through GoDaddy.
2. **cPanel:** cPanel is the primary control panel for the website. It is accessed through GoDaddy and from it, we can directly access the source code. However, as we will discuss later, website updates will be completed using the Git Version control feature found on cPanel.
3. **Firebase:** Firebase is a database service provided by Google that hosts the databases which has pre-exisiting infastructure for read/write abilities on data the website utilizes.
4. **Github:** Lastly, as mentioned, the website code (which we call the webstack) can be found on Github on *reedsb-webmaster*. Additionally, the documentation repository is also on Github.

It is highly recommended that you have bookmarks to these four websites. Bookmarking just these four websites and perhaps placing them in a folder will make your life much easier.

What is the Website?

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

1. **Github Webstack**
   (a) 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) The website is updated by using the Git Version control feature on cPanel.

   (c) The source code can be found in *reedsb-webmaster/reedsb.com*.

2. **Firebase Database**

   (a) 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.

   (b) Firebase can be accessed by using any of your Google accounts. However, you must share the /sin-app-reed/ project to that e-mail to be able to do so.

1 Webstack

The webstack is the source code that repersents the website. This can be found by going on cPanel and accessing the files through there directly. While you may use FTP, again, please use Github version control to update the website.

Note that the website HTML files are actually contained in the /en-us/. This was done in order to allow the website to be sourced from Github. So, while the domain may appear to be ***reedsb.com***, the actual domain is ***reedsb.com/en-us*** and is rerouted from the former domain. As a result of this architecture, there is a subtle matter that is critical to consider.

Warning. *If you find yourself needing to remove and pull the public-html directory from Github for whatever reason, you MUST reset the permissions of the folder. This can be done in cPanel. To reset permissions, right-click the public-html directory and set the permissions to 755, manually editing the permissions. If this is not done, you will probably recieve a 403 permission error when trying to enter the website.*

1.1 Dashdoor

Dashdoor is the adminstrative backend of the website. The Treasurers need dash-door privilages to access it. This can be modified in the ***CONFIG*** in the ***app.js*** file. More detailed instruction on this process are given below. Dashdoor can be accessed by using the following link:

www.reedsb.com/en-us/dash-door

Again, it is highly recommended to bookmark important pages. This is one of them, in addition to the website itself.

2 Firebase

The website works on a free subscription leasing database space from Google Firebase. The databases are needed to store information for running elections, funding poll, and other semester data Treasury needs. In the current website implementation, there are seven primary database collections. They are listed below along with a short description of their purposes.

1. elections: event-indexed collections of the past and current student body election.
2. polls: event-indexed collections of the past and current funding poll.
3. nominations: dataset array containing nominee-submitted information about each candidate.
4. users-who-have-voted: event-indexed collections of the users who have voted in the funding polls and elections.
5. signatories: semester-dependent collection of all currently approved signators.
6. registered-clubs: semester-dependent collection of all currently registered clubs.
7. 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. We could ostensibly refer to the non-semester dependent subset as the event collections.

Definition (Semester-Dependence)**.** *A semester-dependent database collection is any collection that is contingent on the current semester. For these database collections, the website will read the collection corrosponding to the current semester declared in the website's configuration.*

Table of Database Collections

Collection	Semester-Dependent?
signatories	Yes
registered-clubs	Yes
fund-requests	Yes
elections	No
polls	No
users-who-have-voted	No
nominations	No

Note (Semester Subcollections)**.** *Every semester dependent collection has subcollections which are named the semester-specific strings. For continuity, please ensure the string takes the form: [semester] = term-year.*

Example (Semester Naming)**.** *The string name for the any semester dependent subcollection for the Fall 2020 semester is [fall-2020].*

Note (Event Collections)**.** *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

To explain the various of that naming, we break down the label into three parts.

Event Naming Breakdown

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-year.*
3. Lastly, the SHA32 index is simply a random SHA32 hash that uniquely identifies every (semester, event) pair.

Example (Event Naming)**.** *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.*

3 Maintainence

3.1 Regular Maintainence Routines

(A) Updating the Website Codebase

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 by using the cPanel version control menu.

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, find the 'Update from Remote' button by taking the following path below. Once done, that should update the website.

'Github Version Control' → 'Manage' → 'Pull & Deploy' → 'Update from Remote'

So, to summarize, here is how you update the website.

1. Make your changes to the website code on your local Github repository and push your changes.
2. Run the website building script in terminal
3. In cPanel version control, pull the recent changes you've made by pressing 'Update from Remote'.

(B) Semester Updates

Before every semester begins, change the ***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:

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

For reference, these are the semester dependent collections:

Table of Database Collections

Collection	Semester-Dependent?
signatories	Yes
registered-clubs	Yes
fund-requests	Yes

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 ***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 A.

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 ***CONFIG*** and add the relevant emails to the 'authorised-users' array. Then, remember to build the script and update the website as in Routine A. So, to add someone with the [*email*] to dashdoor, here's what you need to do.

1. Add [email] to the array of authorised-users in the ***CONFIG***

2. Update the website as in Routine A.

3.2 Event Running Routines

As the website is used to host events like Student Body Elections and Funding Poll, we document the process of setting up these events below.

Warning. *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* ***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 ***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 ***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 ***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 ***CONFIG*** file to reflect the current senate election ID. Recall that it must follow the form:

   se-[semester]-SHA32id

Gaps in Information

Since the current Webmasters are new, we have never fully tested funding poll and the elections app. There may be some gaps in our knowledge but this is documenting all we know to the best of our abilities. Here are some things to consider learning about:

1. How does club registration interplay with funding poll? We know that given a semester, the funding poll depends on the clubs registered in the database for that semester. But, the relationship and interplay is unclear.
2. A similar question is asked about Senate elections and the campaigns database.

4 Common Subroutines

Here is a comprehensive list of low-level subroutines the Webmasters will commonly use. Feel free to use this reference and augment it with any useful subroutines you may find.

Building app.min.js

To build the ***app.min.js*** script, proceed to the directory containing *build.sh*. On terminal (assuming you have a bash shell), execute the script by writing:

```
bash build.sh
```

5 Closing Remarks

I would like to say one last thing. As you are learning, you might start to gain more knowledge than the person writing this manual. As such, do not take this manual to be the be-all-end-all. If you suspect there is a bug that can be fixed only be defying this manual, then feel free to fall back on your judgement, but update the manual.

On the same note, if you cannot solve a problem, it is not an unreasonable route to double-check the assumptions you have that come from this manual. These assumptions were inferred seperate from the website's designer. So in the end, you must use your best judgement.

Tips

All that being said, here are a few tips.

1. If you need to reconfigure the website DNS, you should go to GoDaddy DNS. You shouldn't need to do this though. However, in case you are updating the Google Firebase domain, you may need to add a TXT record with the authentication there.

2. Manually resetting the cache is your best friend.