# **Event Data Web App**

The purpose of this web app is to populate a database that has the necessary information to personalize the android LightSys event app. The database interface is the php documents under the edit folder. The database is used to create two JSON files which the android event app uses. These JSON are created in getevent.php and getnotifications.php. More details follows.

Table Of Contents

* Database Schema (pg 2-3)
* Database Relation to Event App (pg 4-7)
* Cascading Updates (pg 8)
* Templates Folder (pg 9)
* Miscellaneous Folders (pg 10)
* Helper.php (pg 11)
* Connection.php (pg 12) --Read Before Using GitHub
* Still To Do and Known Issues (pg 13)

# Database Schema

We decided to store information about created events in a database. The information stored in the database can then be pulled when the events need to be edited or when the JSON files need to be created using **getevent.php** or **getnotifications.php**.

The database used to store information about the events that have been created currently contains 13 tables. The database is created with **eventApp-data.sql** and populated using the php files in the **edit** folder. The database schema is as follows:

*event*(*ID, internal\_ID, name, year, refresh\_expire, time\_zone, welcome\_message, logo, contact\_nav, contact\_icon, sched\_nav, sched\_icon, housing\_nav, housing\_icon, prayer\_nav, prayer\_icon, notif\_nav, notif\_icon, theme\_dark, theme\_medium, theme\_color, visible, admin, TZcatagory, custom\_tz, view\_remote, config\_version, notif\_version, refresh\_rate,*)

*themes*(*ID, event\_ID, sequential\_ID, theme\_name, theme\_color*)

*contact\_page\_sections*(*ID, event\_ID, sequential\_ID, header, content*)

*contacts*(*ID, event\_ID, sequential\_ID, name, address, phone*)

*schedule\_items*(*ID, event\_ID, sequential\_ID, date, start\_time, length, description, location, category*)

*housing*(*ID, event\_ID, sequential\_ID, host\_name, driver,host\_id*)

*prayer\_partners*(*group\_ID, event\_ID, sequential\_ID*)

*attendees*(*ID, event\_ID, sequential\_ID, name, house\_ID, prayer\_group\_ID*)

*notifications*(*ID, event\_ID, sequential ID, title, body, date*)

*info\_page*(*ID, event\_ID, sequential\_ID, nav, icon*)

*info\_page\_sections*(*ID, info\_page\_ID, sequential\_ID, header, content*)

*users*(*ID, username, password*)

*event\_users*(*ID, user\_ID, event\_ID*)

For a given *event\_ID*, the *sequential\_ID* of any table acts like an auto-incrementing ID within that event.

The database schema pictured below reflects the schema above and shows the relationships between the tables. This is how it was originally set up, and is currently outdated but mostly correct.

**Suggested changes:** Change the primary keys of the tables to the *event\_ID* paired with the *sequential\_ID*, instead of using an auto-incrementing integer. This change must be reflected in most of the existing files.

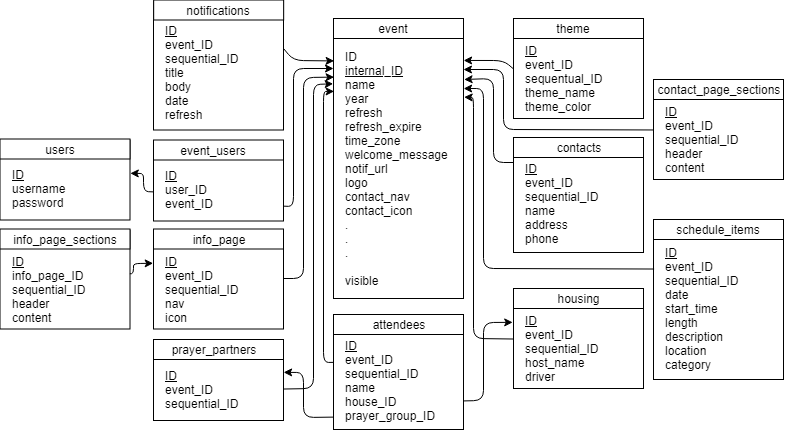


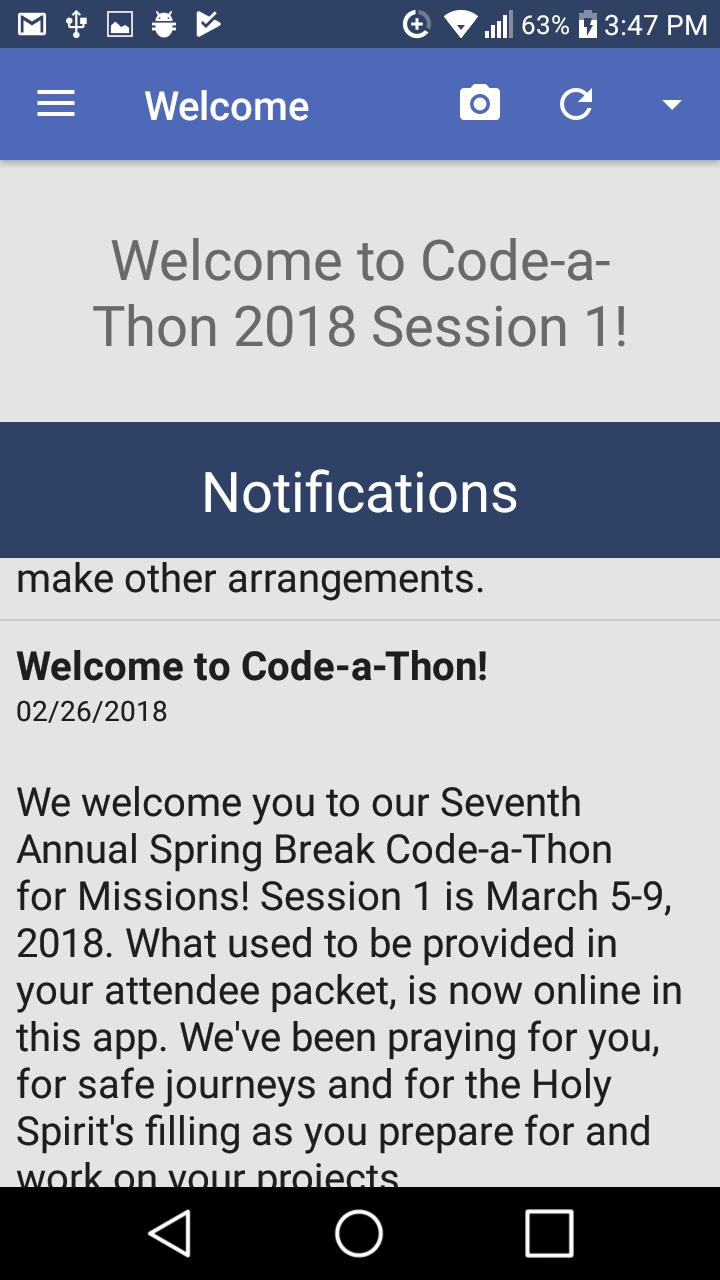
Figure 1. Event App Database Schema. (outdated)

# 

# Database Relation to Event App

This section demonstrates how the information contained in each table relates directly to the information shown in the app and available to users. Although the database is used to generate the JSON files necessary for the app to display an event and only can be used by the app with this intermediate step, this section shows what information from the database is eventually made visible in the app, using the LightSys Code-a-Thon 2018 Session 1 event as an example. (This section does not discuss every field in every table.)

The first table, *events*, contains two ID attributes, and both individually distinguish each event. The first ID attribute, called *ID*, differentiates them for the users, while the other, called *internal\_ID*, differentiates them to the rest of the database and is only used internally.



Figures 2: *Welcome\_message* attributes.

The text contained in the *welcome\_message* attribute is displayed at the top of the Welcome screen, as shown in Figure 2. In this case, the welcome message is “Welcome to Code-a-Thon 2018 Session 1!”

The contents of the *logo* attribute as well as each *nav* and and *icon* attribute in the *events* table are represented as the images and headers in the navigation menu. Note that these may not include every icon and associated header in the navigation menu; additional headers that can appear will be discussed below, also icon has been hardcoded. The *logo* is pulled from the database and shown at the top of the navigation menu, as shown in the figure below. The data contained in the *notif\_nav*, “Notifications” in this case, as well as the bell icon associated with it in the *notif\_icon* attribute, make up the first link in the navigation menu, and the other associated *navs* and *icons* are similarly displayed:

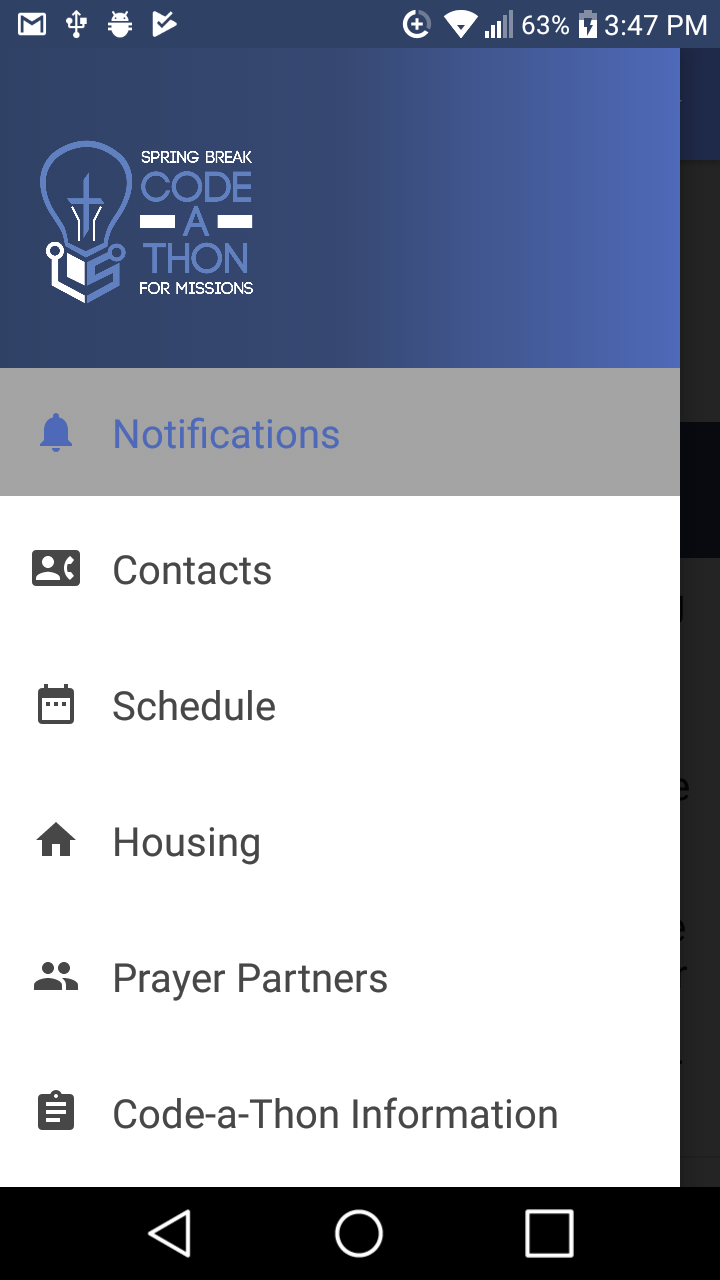


Figure 4: Contents of the *logo*, *nav*, and *icon* attributes displayed in the app.

Pages with additional information about the event can be specified by the event creator. The database stores information about these pages in the *info\_page* and *info\_page\_sections* tables. Each of these pages has a *nav* and *icon* associated with it (contained in the *info\_page* table), and these are displayed in the app’s navigation menu below those that are pulled from the *event* table. In the figure above, “Code-a-Thon Information” is the *nav* of one of these user-defined pages.

Each of these user-defined pages can have one or more sections, and the content of each of these sections is contained in the *info\_page\_sections* table. For the “Code-a-Thon Information” page, the *header* attribute of the first section is “WiFi,” and the contents of its associated *contents* attribute are displayed directly beneath it, as shown below:

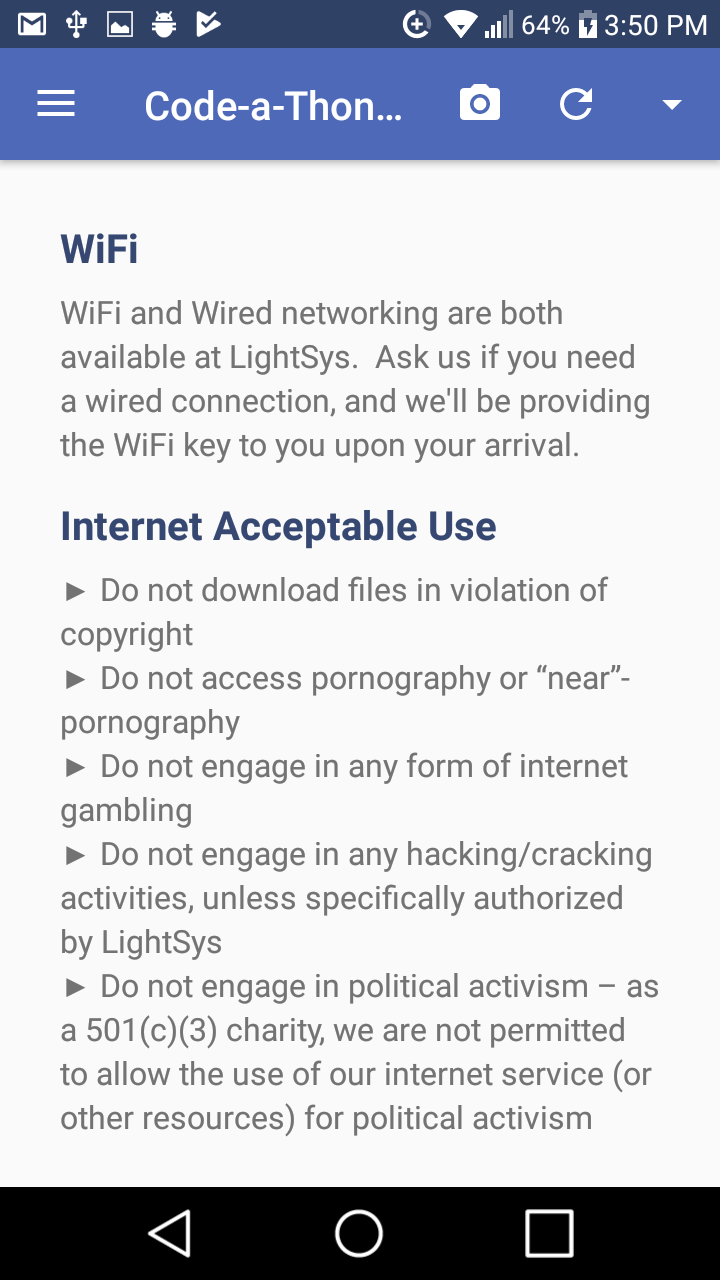


Figure 5: “Code-a-Thon Information” as an additional user-defined page.

“Internet Acceptable Use,” as expected, is the *header* of another of the sections on this page.

Information for the notifications that populate the section at the bottom of the Welcome page is contained in the *notifications* table. Each notification has a title pulled from this table’s *title* attribute a message pulled from the table’s *body* attribute, and the date it was created and posted pulled from the *date* attribute. In the figure below, the *title* of the notification shown is “Welcome to Code-a-Thon!”, and its associated date and message are shown below it:

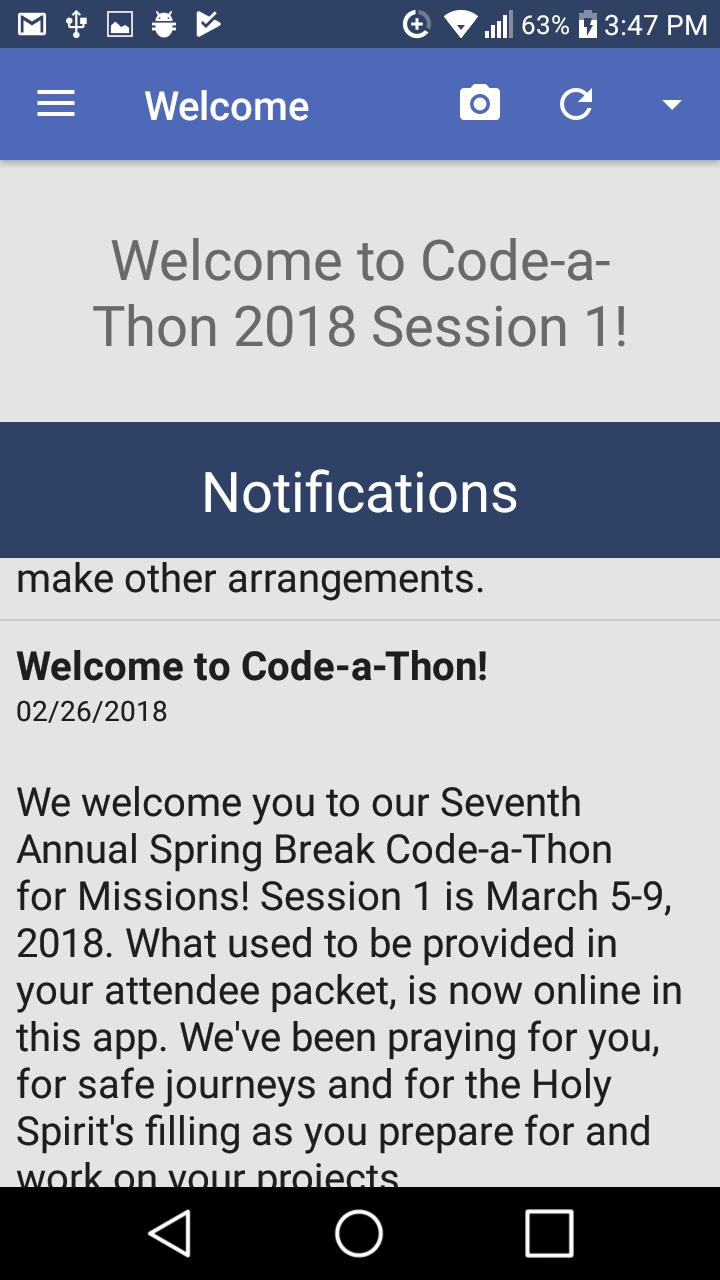


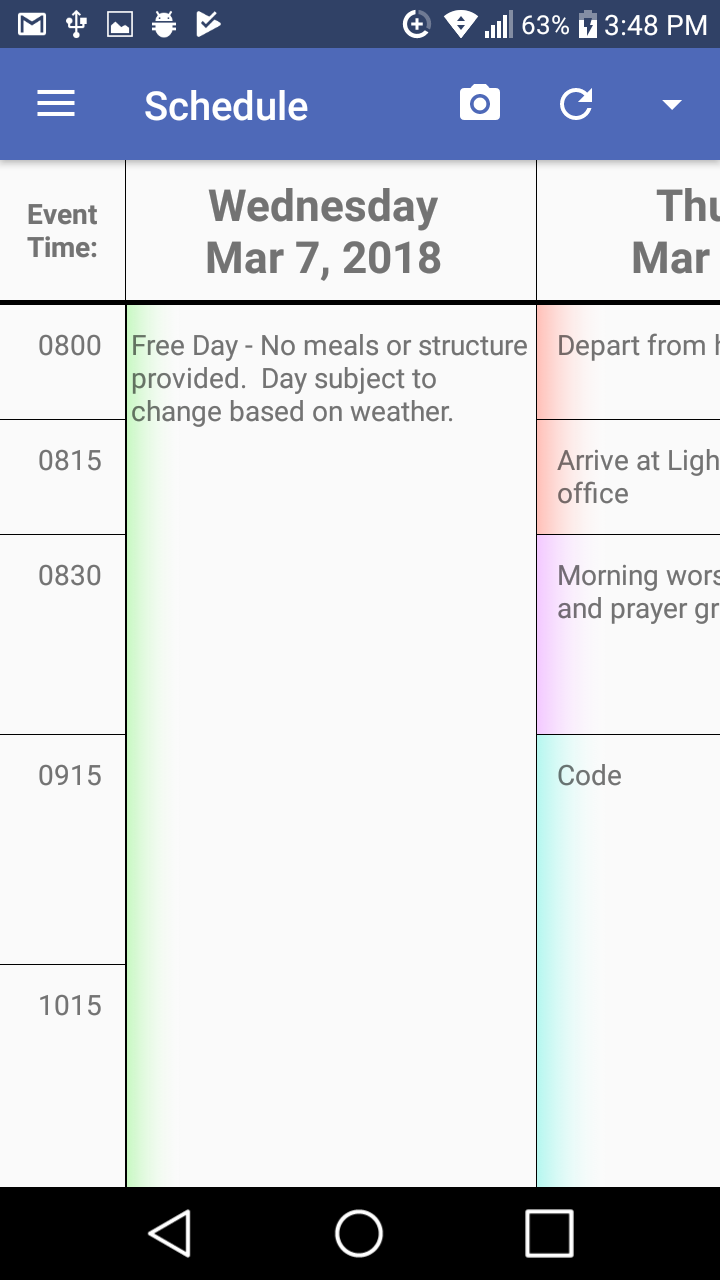
Figure 6: “Welcome to Code-a-Thon!” notification.

The Contacts page is set up and populated using the information stored in the *contact\_page\_sections* and *contacts* tables. The information contained in the *contact\_page\_sections* table structures a contact page in essentially the same way that information in the *info\_page\_sections* table structures a user-defined information page. Contacts are pulled in by looking at the names in the *content* attribute of the final contact page section and finding and displaying the information (from the *address* and *phone* attributes of *contacts*) associated with those names.

The Housing page also draws from the *contacts* table, as every host (defined in the *host\_name* attribute) is also a contact with a name, address, and possible phone number. The *housing* table also contains a *driver* attribute that tells who will drive the attendees housed at that event to and from the event. This name is displayed above the associated addresses as shown below. The names of the guests listed on the right next to the addresses where they will be housed are pulled from the *name* attribute of the *attendees* table.

Both the Housing page and the Prayer Partners page pull from the *attendees* table. For the Prayer Partners page, names from the *attendees* table are displayed next to the number of the prayer groups the attendees have been placed in.

The Schedule page lays out a schedule for each day of the event. The *date* attribute of the *schedule\_items* table is used for the date at the top of each column, and the *start\_time* and *length* determine where in a column something is displayed, using the times shown on the left side. The *category* attribute references the *theme* table, which determines the color of something on the schedule. The *location* attribute references the *contacts* table as each location should have an address and phone number.



# Cascading Updates

# Several tables are referenced by other tables, and if something in the original table is changed the other tables need to reflect it. These original tables are attendees, contacts, and themes. Attendees saves the ID of the thing referencing it (house\_ID and prayer\_group\_ID). This way if an attendee name is changed it still has the same housing and prayer group. Also housing and prayer group can only have one instance of a person, a person cannot be in multiple groups. Contacts and Themes are reversed. The things that save contacts and themes save their ID so that if a name is changed they are still linked. The ID’s are converted into names for the json file. In this way a theme can be used for multiple schedule items and a contact can host several homes or be printed multiple times on contact page.

# Templates Folder

This folder includes:

* Left-nav.php (provides the navigation bar for the user)
* Head.php (provides needed links for the pages)
* Logout.php (destroys session data and redirects user to login screen)

The code takes advantage of the PHP include function to template portions of the page, such as the left nav. Thus to put the left nav in the page you use:

<?php include("../templates/left-nav.php"); ?>

To edit the left nav then modify the included file and it will update everywhere.

On a given page edit the CSS locally, using the ID of the link, such as

<style>

#general {

background-color: grey;

color: white;

}

</style>

# Note that if you include head and left-nav you must include head before left-nav or it may be overwritten.

# Temp Folder

This folder holds logos during the upload process and then deletes them when done. The images will be stored in the folder briefly while they are stored in the database, and then they are deleted. Note that currently this is the same folder for all users, so multiple users happening to save it the same time could cause an error.

# Phpqrcode Folder

This folder has all of the nessicarry files, licenses and more to generate a qr code for the configuration JSON (getevent.php). This qr code is used by the app to get to the event JSON.

Styles Folder

This folder houses the styles used in the web app. Note that leftnav overwrites the styles for fill color and text color changing all the buttons used in the web app to reflect the user defined theme color.

Scripts Folder

This folder houses several Java script files that support the web app

# Helper.php

This nugget of gold contains quite a few Quality of Life functions that improve… your quality of life. Some of the standouts will be mentioned in detail here. If you find a function in a file your working on that is not base php, likely its housed here. Some real winners are:

Json\_encode\_noescape()

This guy handles making your jsons, he is good at it to. You feed an associative array into it, and it spits out a json string.

Note: this guy functions using recursion, example of use is getevent.php.

Secure()

This does what it says on the tin. It handles session timers, affirming login credentials, and keeps legitimate and illegitimate users from navigating to portions of the web app that they should not have access to. Any user who fails a security test has their session destroyed and is logged out. If the user’s browser does not allow redirects, it kills the users page with a die() call.

Must have a previous session\_start() call in the page to work properly, as well as occuring before any other information is sent to the client.

EventSecure()

This is a version of the Secure() function that is used on pages where a user may not be tied to a particular event (for example, during event creation). Main appearance is in general.

Inc\_config\_ver()

Is callled whenever a user action (besides those on notifications) is preformed that affects the database( and should be used for such future situations) this increments the config\_version value in the event table so the mobile app knows when to update.

Inc\_notif\_ver()

Behaves as Inc\_config\_ver() except that it only triggers on actions affecting the notifications table, and increments the notif\_version.

# Connection.php

This is the PHP file that handles making the database connection. Whenever you want to call the database, you must include this file.

This file has some config information for your database which means two things;

1. If the password, ip-address, or username changes, this must be updated for connections to work.
2. IMPORTANT! CAUTION! WARNING! ATTENTION!

Because this file contains the login credentials to your database, this should NOT be uploaded to github without dummy credentials. This will create much work for the boss, and that not a good thing.

# Still To Do and Known Issues

* Add form validation to make sure people don’t miss important things or enter random garbage. Required fields should be added. For instance the app won’t run without at least one information page.
* Add multiple event locations, the person making the json can enter several timezones, or a place name coupled with a timezone, and the app lets the user choose the location.
* Make the contact page able to have more than one writing section and one people section. (Some of the base code is written, but it needs a lot of work. Keep an eye on the json generator “getevent.php” for how the changes will effect it.
* Fix the temp folder so that multiple people uploading at the same time will not break it.