PHP API Documentation

This is an overview of all of the .php documents you will find in the webpage source files. Please use this document to get an idea of how the project is organized and how you can expand its functionality in the future.

Code Overview

index.php and admin.php

These files are essentially HTML files that will load our classes and allow our user to interact with the database. The homepage (index.php) and admin panel (admin.php) use the Forms class to display a form for the user to enter a date range, as well as a timeframe for the data. Both classes then use the Display class to show the appropriate response. Both pages will only invoke the display class if the $_GET field is filled (indicating the user has submitted a request) otherwise will simply display the default forms by default. Admin has additional functionality with $_POST and will display a message when the user has made a deletion request and display a message. All function calls will insert HTML code inline with the page, so you are free to encase the function calls with <div> tags if you wanted to add additional styling with CSS for each module.

Forms.php

This function displays the form that contains 2 text forms for the user to enter the date range, a drop down to select the timeframe, as well as 3 submit buttons.

Here is what the user should expect from these inputs.

range_start

The first text box will indicate the beginning of the date range. Many date formats can be accepted but ideally the user should stick to the format provided by default. More information about accepted inputs can be found in the stringToDate function. The user can also type “min” to indicate they want the earliest data sample to be included. This text box will be automatically filled with the current $start value on load.

range_end

The second text box will indicate the end of the date range. Just as before, more information about accepted inputs can be found in the stringToDate function. The user can also type “max” to indicate they want the most recent data sample to be included. This text box will be automatically filled with the current $end value on load.

range_frame

This dropdown box will indicate the timeframe for the resulting data output. This will affect the resulting statics data to be in relation to this timeframe e.g. average number of students per hour. You can add items to this dropdown by looking at the `private $timeframes` array. This drop down will automatically select the current $frame value on load.

Submit

This button will submit all of the data to $_GET and reload the page. On reload $start, $end, and, $frame will be filled and index.php or admin.php will be able to invoke the Display class with the given data.

Show All

This button will add the ‘show_all’ value to $_GET and which will auto set $start and $end to min and max respectively. The $_GET for range_start and range_end may not reflect this, but the ‘show_all’ setting overrides this information. $frame will still reflect what $range_frame was set to.

Reset

This button will clear all of the information in $_GET and reload the page. This will result in the page the user loaded for the first time.

Variables

$start holds the start time of the data range, will accept a unix timestamp as int or ‘min’ as a string.
$end holds the end time of the data range, will accept a unix timestamp as int or ‘max’ as a string.
$frame stores a string that represents the time frame that the Display class will use. Will accept time frames ranging from ‘sec’ to ‘year’. NOTE Seconds was removed from the dropdown because of performance issues.
$timeframes stores and array of strings that defaultForms() will put in the range_frame dropdown. NOTE Seconds was removed from the dropdown because of performance issues.

defaultValues()

Will set $start and $end to exactly 30 days ago to now() and set $frame to ‘hour’. These default values are set when $_GET is empty (first time user load page or RESET is clicked) or when the inputs are invalid.

__construct()

This is the class constructor. It is called when the page is loaded. It will fill the $start, $end, and $frame variables to the appropriate information depending on the values in $_GET. These values have been outlined in the previous sections.

stringToDate($input)

This will take in a string (ideally, a human readable date) and convert it to a unix timestamp. The user will be able to define a timezone as well as use 24 or 12 hour time. It first tries to convert the date using the following formats: mm-dd-yy 24:59:59 CST and yy “mmm” d (eg 17 Dec 1) and the same formats but with a 4 digit year. If this fails to match the date format, we will fall back on the built in strtotime() which you can find information on here
http://php.net/manual/en/function.strtotime.php
If strtotime fails, then the function returns -1.

dateToString($input)

This function converts and unix timestamp to a date formatted as “mmm” d, yyyy, 24:59:59 (eg Mar 21 2018 HH:MM:SS) This function will use 24 hour time by default.

defaultForms()

This function outputs the HTML code for the form that contains the data range and submit buttons. Data from $start, $end, and $frame will be automatically filled.

DBLogin.php

This class will attempt to connect to the SQL database using the information provided in the class. It will then create a $pdo object that any class can use to make SQL queries. Although this job is mainly left to the Query class to keep everything simple.

Variables

This information will need to be filled in to connect to your database
$default_table : the table that Query will act upon by default. Many Query functions allow you to select a different table in case you want to store your data elsewhere.
$servername, $user, $pw, $database: This information should match your SQL database setup

connect()

Will attempt to make a PDO connection using the data given. If successful, the connection will be saved in the $pdo public variable. If there is a failure, the function will echo an error message and the $pdo will be left uninitialized.

Query.php

Query makes it easy for the other classes to select or delete database without the risk of using the incorrect SQL syntax.

__construct()

The constructor creates a DBLogin object and calls connect(). Query will save this connection in the private $pdo variable.

getResults($start, $end, $table = "")

Will take a start and end time, and optionally a different table, and then return the rows of that table between the the dates provided. The SQL table must have a column named “date” with the datatype being TIMESTAMP

deleteRowCount($count, $table = "")

Will take an integer ($count) and optional table name and will delete the row in the table with the column value “count”. The count column in the table must be an integer datatype. It will echo a message indicating the deletion was successful or unsuccessful with associated error message.

queryBuilder() and deleteRowBuilder()

A helper function called by getResults() and deleteRowCount() to return the actual SQL syntax required to fulfill that function’s action.

StaticData.php

This class has a set of functions that will return different statistical data from the database. Most API functions will take in a $starttime, $endtime unix timestamps, as well as an array of unix timestamps that should have come from the database, and finally a $timeframe representing the timeframe in which each function will base its calculation on.

<something>_in_range($starttime,$endtime,$fromDB,$timeframe)

Currently, this class has API functions avg_in_range(), median_in_range(), mode_in_range(), count_in_range() - doesn’t take a timeframe -. These functions will return the statistical analysis of the data provided within the timeframe provided. Also included are highs_in_spec_range(), and lows_in_spec_range() which will divide the time range into the $timeframe and return the unix timestamp that had the most or least hits respectively in that given $timeframe.

Also included in this class are some helper functions that should only be used by the API functions in this class, but are still public in case they may provide additional functionality for another class in the future.

min_in_set($data)

Will return the minimum value in an array or dictionary ($data) excluding zeros. Only when all of the values are 0 will it return 0.

max_in_set($data)

Will return the maximum value in an array or dictionary ($data) excluding zeros. Only when all of the values are 0 will it return 0.

in_range($starttime,$endtime,$value)

Will determine if a given timestamp ($value) is between the $starttime and $endtime. Returns TRUE or FALSE for each case.

prepare_data($starttime,$endtime,$fromDB,$timeframe)

This function will be called by most of the API functions. It takes in the raw array of timestamps $fromDB and convert it into a dictionary of timeframes where $timestamp => $count and $count is the value in which the timestamp from the database appears within the timeframe. The function will then return this dictionary as $date_count.

add_from_timeframe($datetime, $timeframe)

This function takes a DateTime class and then adds the given timeframe to it. Say we start off with Tuesday Feb 27 2018 and we want to add a ‘week’ to it. This function will return Tuesday March 3 2018. This function then returns the unix timestamp of that date.

to_floor_time ($unix_time, $timeframe)

This function floors the $unix_time and returns the $timeframe that the $unix_time resides in. For example an input of March 28, 2018 and ‘month’ will return March 2018 as a unix timestamp.

parse_timeframe ($timeframe)

This function simply converts each $timeframe to seconds. For example there are 604800 seconds in a week,

Display.php

This class is called by index.php or admin.php to display the appropriate data given by the timeframe, and start and end times. The API functions will normally accept the inputs ($starttime, $endtime, $timeframe). These API functions return HTML code directly in place where they were called.

__construct()

The constructor will create the classes required to display the data. In this case Query to receive data from the database, StaticData and GraphBuilder to put information on the page.

showTable($starttime, $endtime, $timeframe, $admin = 0)

Will show the database table between the data given in descending order (most recent to earliest). The $timeframe variable is not used. Currently, the function assumes the SQL table has the columns 'count', 'date', 'confidence', 'location', 'device_id'. Adding more columns to the arrays in showTableRow() will allow tables with other column information. The optional $admin variable when set to 1 will take on a DELETE button to the end of each row, allowing the user to delete the information in that row from the database. Adding a name or id tag to the first echo <table> can allow you to change the style of this block of information with CSS or Javascript.

showStaticData ($starttime, $endtime, $timeframe)

This will display the statistics information from the time range and time frame provided. This function will create an array of timestamps from the database and should pass them into all of the API functions of StaticData. Adding a name or id tag to the first echo <p> can allow you to change the style of this block of information with CSS or Javascript.

showGraph ($starttime, $endtime, $timeframe)

This will plot a graph using the time range and timeframe provided. This function will create an array of timestamps from the database and then call the GraphBuilder::showGraph($result, $timeframe) function.

GraphBuilder.php

loadGoogleCharts()

This is used to load the Google Charts API. It uses https://www.gstatic.com/charts/loader.js as its source and jQuery 1.8.2 from https://www.gstatic.com/charts/loader.js. This is necessary to be able to use the Scatter plot, which is included as a package.

drawChart()

This takes the JSON encoded data to produce the Number of Passes vs. Date graph, along with basic graph attributes such as the title and axes labels.

Predict.php

This file predicts the amount of traffic for a given time frame using a weighted average so that the more recent data is weighted more heavily. It then calculates the expected trend of the data. It presents all of this data to the user.