**Overview**

In this assignment you will create a simple web service .You are provided with client side code that runs in the browser and consumes your web service. Combined this simple web application allows posts to a message board.

**Your Task**

**Set Up**

create three files

post.php

getposts.php

posts.db

The \*.php files should have rw- permissions for you and r-- permissions for the group and others. This can be accomplished by running chmod 644 on these files. posts.db needs rw- permissions for you, the group, and others. This can be accomplished by running chmod 666 posts.db.

**What are these files for?**

posts.db is a simple database for storing the posts that have been submitted to your web service. Each post is stored on a separate line as an associate array with two fields time and message, more information regarding these below, serialized as JSON.

post.php is a PHP script that accepts a message via a HTTP POST request, and then saves this message to posts.db. It also echoes the final JSON saved to posts.db to STDOUT, and hence this is returned in the body of the HTTP response. The first two lines of PHP code in this file should be

header('Content-Type: application/json');

header('Access-Control-Allow-Origin: \*');

This gives the HTTP response our own headers. The first indicates that we are returning JSON. The second disables the default CORS security of the browser. *You do not need to understand CORS to continue with this assignment.*

Following this your code should do the following:

1. Check that the request method, see $\_SERVER['REQUEST\_METHOD'], is 'POST'. If this isn’t the case, then you need to create a PHP array with a single key, error, and the value of this key is a string indicating what the error is. Convert this array to a JSON string and echo it to STDOUT with echo json\_encode($array), where $array is your PHP array. After this, your script should exit immediately, see the PHP function exit.
2. Check that $\_POST['message'] is set with isset($\_POST['message']). If this isn’t set then return an error following the instructions from 1.
3. Check that the length of the message, see the PHP function strlen, is not greater than 140 characters. If the message is too long then return an error following the instructions from 1.
4. Validate the message by checking that it only contains spaces, characters a-z and A-Z, digits 0-9, and new line characters \n. If message contains characters other than these then return an error following the instructions from 1.
5. If all of your error checking above shows no problems then create an array

['message' => $message, 'time' => time()];

where $message is a variable containing the submitted message. 6. Encode this PHP array as JSON and **append** it to posts.db following the JSON string with a \n. The PHP functions fopen, fwrite, and fclose will be helpful. 7. Finally, echo the JSON string that you wrote to posts.db to STDOUT.

getposts.php is a PHP script that returns the messages saved in posts.db as JSON. Like in post.php the first two lines of PHP code in this file should be

header('Content-Type: application/json');

header('Access-Control-Allow-Origin: \*');

This script needs to be able to accept an optional query string with a single field called since. The value of this field will only contain digits, it is a UNIX timestamp. Your script should do the following:

1. Check if $\_GET['since'] is set. If it is, then save this value in a variable $since. Otherwise, set $since to null.
2. If $since is not null then validate it. It should be nonempty, and only contain digits 0-9. If this is not the case return an error response to the user as detailed for post.php.
3. If $since is not null then convert its type to an int with, for example, $since = (int)$since.
4. Read all the posts from posts.db, the PHP function file should be useful.
5. Construct a PHP array with the posts that have a time field larger than $since, or all posts if $since is null. For this, you should use json\_decode($line, true), where $line is a line read from the file, to convert the JSON string back to a PHP array. The posts in this array should be sorted by an increasing time field.
6. Finally, echo to STDOUT as JSON the PHP array ["posts" => $posts], where $posts is the array from 5.