**RPOJ2**

PART A - Implement

1. Web service that supports at least four (4) unique GET and four (4) unique POST actions
2. Build a correctly formed Database to implement the specifications of PROJ1
3. Integrates with a backend database for at least two (2) CRUD actions
4. Logging feature that accounts for every request with IP, browser, timestamp and action

It saves into a apache\_access.log

A close up of a device

Description automatically generated

1. Rate limit Web Service to one request per second per user session

A screenshot of a cell phone

Description automatically generated

1. Limit per session request to 1,000 in a 24hour period

A screenshot of a cell phone

Description automatically generated

1. Domain lock web service to a whitelist of referrers

A screenshot of a cell phone

Description automatically generated

PART B - Structure

1. One database object with all SQL actions
2. One session object with all user, logging and rate limiting actions
3. A default JSON error if request signature incorrect
4. Validation of GET and POST data before it is processed by any object

A screenshot of a cell phone

Description automatically generated

1. All output from Web Service is JSON

A valid result

A screenshot of a computer

Description automatically generated

A json message will be shown after an error

A screenshot of a cell phone

Description automatically generated

1. HTTP Error response codes three (3) implemented

HTTP error response code implemented in read.php / read\_single.php/ whitelist.php

A screenshot of a cell phone

Description automatically generated

A screenshot of a cell phone screen with text

Description automatically generated

A screenshot of a cell phone screen with text

Description automatically generated

PART C - Code Comments

1. Denote where and explain why you instantiated the database and session objects in that location

I put my DB and session object in a config folder so that I know where to look at when I need to require it from another file.

1. Explain the mathematics of either of your two rate limiting code.

For 1000 request in 24hrs code. I made three variables in session. One counting every access($accessCount), one record the current time($currentTime) and one remember the first-time server received a request($firstAccess).

If $accessCount goes up to 1000, it die() and echo a json\_encode(error);

If $currentTime are 1 day or more later than $firstAccess, it unset $firstAccess and set $accessCount to 0.

I tried to make a rolling 24 hrs counter but that doesn’t work.

1. Note where you are checking if a session pre-exists, what are you doing if it does.

I have a variable checking every time when server get a request($accessCount) as above question.

If the $accessCount goes up means session pre-exist.

If I have to reset the whole session, I will comment out the session\_destory() code;

1. Explain the code structure that checks all of the GET/POST structures
2. Write a README file that explains how to setup and configure Web Service
3. Write a test script that interacts with the web service to test all the known GET and POST requests as a part of Unit testing

read all posts: <http://localhost:8888/PROJ2/api/recipe/read.php>

read one post: <http://localhost:8888/PROJ2/api/recipe/read_single.php?id=6>

create a post: <http://localhost:8888/PROJ2/api/recipe/createRecipe.php>

submit this in the body

user\_id can choose ( 1, 2)

measurement can choose (1,2)

{

"name" : "submitted from postman",

"description" : "please work",

"user\_id" : "1",

"quantity": "30",

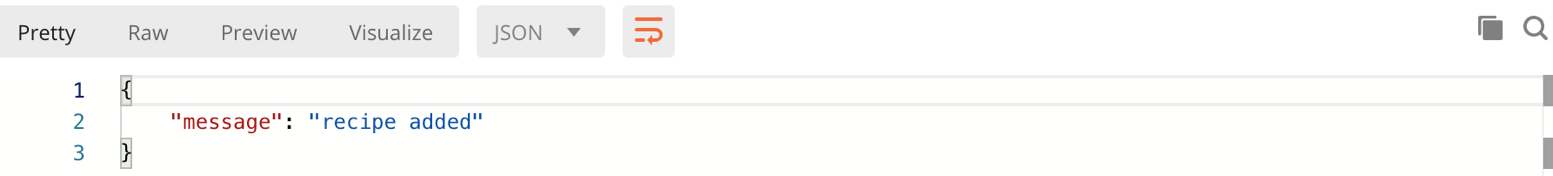
"measurement": "2",

"item": "tears",

"step": "just drink the tears."

}

If successfully submitted:



Update a post: http://localhost:8888/PROJ2/api/recipe/updateRecipe.php

{

"name" : "updated with postman",

"description": "work please",

"quantity": "1",

"measurement" : "2",

"item" : "sweat",

"recipe\_id" : "14",

"i\_id" : "17",

"step": "updated step",

"met\_id" : "8"

}

If successfully update:

A screenshot of a cell phone

Description automatically generated

Delete a post: <http://localhost:8888/PROJ2/api/recipe/deleteRecipe.php>

{

"recipe\_id" : "14"

}

If successfully deleted

A screenshot of a cell phone

Description automatically generated