Weather Data API Assignment

Environment - Windows XAMPP Version - PHP Version 7.4.3 Framework- Codeigniter

SETUP-

To access -Weather details API from GITHUB repository-

- 1] Download code from GITHUB URL, extract zip & place it in xampp/htdocs. Rename the folder as ci_weatherdata
- 2] Start XAMPP server
- 3] XAMPP server needs to be configured as given below in explanation
- 4] Access URL http://localhost/ci_weatherdata/api/listWeatherData on postman
- 5] Select Method GET

Explanation-

Installed redis 5.3.7 for Windows - for PHP 7.4 - Downloaded package ->7.4 Thread Safe (TS) x64

In XAMPP -

Downloaded php_redis.dll and added to xampp/php/ext

In php.ini add new line extension=redis

In XAMPP with <?php phpinfo(); ?> Check if redis is enabled

redis

| Redis Support | enabled |
|------------------------|---------|
| Redis Version | 5.3.7 |
| Redis Sentinel Version | 0.1 |
| Available serializers | php |

Install redis- Redis-x64-3.0.504.msi ->from https://github.com/MicrosoftArchive/redis/releases Select port 6379 Start redis server.

1] WEATHER API-

In xampp/htdocs add folder folder - ci_weatherapi from github link In ci_weatherapi I have added - application/config/rest.php application/libraries/Format.php application/libraries/REST_Controller.php application/controllers/Api.php

For fetching weather api data registered on https://home.openweathermap.org/users/sign_up Taken OpenWeatherMap API key Default API key- 53026eb212faddb3a68f235d72994172 For city id to pass in code - 1259229 as city id for Pune city

```
https://bulk.openweathermap.org/sample/city.list.json.gz - Downloaded zip for city id from link
```

```
For Pune city data is-
{
    "id": 1259229,
    "name": "Pune",
    "state": "",
    "country": "IN",
    "coord": {
        "lon": 73.855347,
        "lat": 18.519569
    }
},
```

2] To store data in Redis cache for 10 mins & for better response time-In application/controllers/Api.php -

In controller construct added-

```
$this->load->driver('cache', array('adapter' => 'redis', 'backup' => 'file')));
```

In controller listWeatherData_get method added-

Added 600 seconds for 10 Minutes storage

In application/config/config.php-

To enable Redis, set the session driver to "redis" and we can set the value of sess_save_path localhost for local testing.

```
Modify from

$config['sess_driver'] = 'files';

$config['sess_save_path'] = NULL;

to

$config['sess_driver'] = 'redis'; //enable redis session

$config['sess_save_path'] = 'tcp://localhost:6379'; // redis hostname:port
```

3] For HTTP request rate limit-

Created object of redis and connected to redis port. Initialized maximum calls limit, time period, total calls.in listWeatherData get() function.

```
1] $max_calls_limit-
```

Maximum number of calls a user can access the resource (5 calls)

2] \$time_period-

Defines the timeframe seconds within which a user is allowed to access the resource per the \$max_calls_limit.(20 seconds)

3] \$total_user_calls-

Initialises a variable that retrieves the number of times a user has requested access to the resource in the given timeframe. (0 total user calls)

Added if else statement is used to check if there is a key defined with the IP address on the Redis server.

Retrieved user ip address of the user requesting the web resource using PHP built-in functions.

If the key doesn't exist, if (!\$redis->exists(\$user_ip_address)) {...}, we set it and define its value to 1 using the code \$redis->set(\$user_ip_address, 1);

The \$redis->expire(\$user_ip_address, \$time_period); sets the key to expire within the time period, in this case, 20 seconds.

If the user's IP address does not exist as a Redis key, we set the variable \$total_user_calls to 1.

In the ...else {...}... statement block, we use the \$redis->INCR(\$user_ip_address); command to increment the value of the Redis key set for each IP address key by 1. This only happens when the key is already set in the Redis server and counts as a repeat request.

In if else conditions we use curl to fetch data from weather api.

The statement \$total_user_calls = \$redis->get(\$user_ip_address); retrieves the total requests the user makes by checking their IP address-based key on the Redis server.

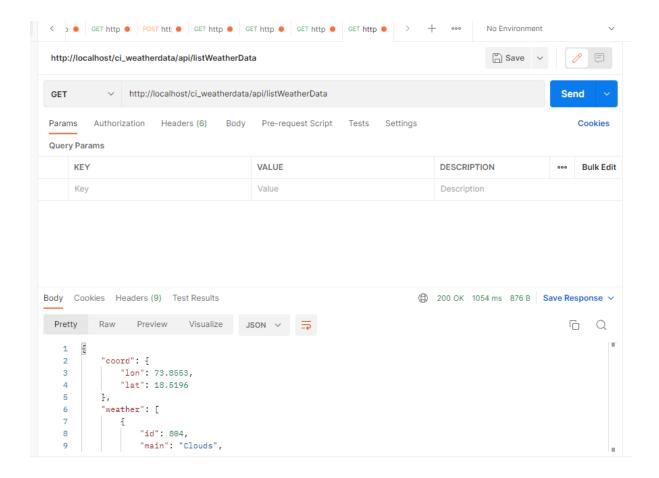
Toward the end of the file, we use the ...if (\$total_user_calls > \$max_calls_limit) {... }.. statement to check if the limit is exceeded; if so, you alert the user with echo "User " . \$user_ip_address . " limit exceeded.";.

In 20 seconds it allows 5 consecutive api calls & temporarily restricts the user on 6th api call after that i.e. gives message limit exceeded.

Again resume it for next 20 seconds.

Postman API Output-

After accessing weather data api postman gives the output as below-



After 5 consecutive API calls in 20 seconds it gives the output as below-

