```php
<?php

namespace App\Helpers;


use GuzzleHttp\Exception\GuzzleException;
use GuzzleHttp\Client;

/**
 * Class to handle all RESTful requests
 */
class HttpHelper
{
    private $guzzle;
    private $un;
    private $pw;

    /**
     * HttpHelper constructor.
     */
    public function __construct()
    {
        $this->guzzle = new Client(['base_uri' => 'https://sample.api/api/v1/']);
        $this->un = getenv("API_USERNAME");
        $this->pw = getenv("API_PASSWORD");
    }

    /**
     * @param $endpoint
     * @param $array - Array of data to be JSON encoded
     * @return mixed
     */
    public function post($endpoint, $array) {

        $response = $this->guzzle->post($this->cleanEndpoint($endpoint), [
            'headers' => [
                'Content-Type' => 'application/json; charset=UTF8',
                'timeout' => 10,
            ],
            'auth' => [
                $this->un,
                $this->pw,
            ],
```

```
            'json' => $array
        ]);

        $body = json_decode($response->getBody());

        return $body->data;
    }

    /**
     * @param $endpoint
     * @param int $page
     * @return mixed
     */
    public function get($endpoint, $page = 1) {
        $page = intval($page);

        $response = $this->guzzle->get($this->cleanEndpoint($endpoint) . "?page=$page", [
            'headers' => [
                'Content-Type' => 'application/json; charset=UTF8',
                'timeout' => 10,
            ],
            'auth' => [
                $this->un,
                $this->pw,
            ],
        ]);

        $body = json_decode($response->getBody());

        return $body->data;
    }

    /**
     * @param $endpoint
     * @param $array - Array of data to be JSON encoded
     * @return mixed
     */
    public function patch($endpoint, $array) {

        $response = $this->guzzle->patch($this->cleanEndpoint($endpoint), [
            'headers' => [
                'Content-Type' => 'application/json; charset=UTF8',
                'timeout' => 10,
```

```
        ],
        'auth' => [
            $this->un,
            $this->pw,
        ],
        'json' => $array
    ]);

    $body = json_decode($response->getBody());

    return $body->data;
}

/**
 * @param $endpoint
 * @return mixed
 */
public function delete($endpoint) {

    $response = $this->guzzle->delete($this->cleanEndpoint($endpoint), [
        'headers' => [
            'Content-Type' => 'application/json; charset=UTF8',
            'timeout' => 10,
        ],
        'auth' => [
            $this->un,
            $this->pw,
        ],
    ]);

    $body = json_decode($response->getBody());

    return $body->data;
}

/**
 * Remove leading or trailing forward slashes from the endpoint.
 * @param $endpoint
 * @return string
 */
private function cleanEndpoint($endpoint) {
    $endpoint = ltrim($endpoint,"/");
    $endpoint = rtrim($endpoint,"/");
```

```
        return $endpoint;
    }

}
```