Section 1: Application Programming Interface

Json example: { “users”:[{“id”:”1”,”username”:”test”,”password”:”test”}]}

a. i. GET

GET used to retrieve data from a server at the specified resource is identified by the Request-URI

https://api-exampleurl.com/api/v1/users/1

It should retrieve all the accessible information about the user with the **ID** 1

ii. **POST**

POST used to send data to the API sever to create or udpate a resource. The data sent to the server is stored in the request body of the HTTP request

https://api-exampleurl.com/api/v1/users

This should be used to create a new user can send all the required information username, password, etc. using the body of the **HTTP** request

iii. **UPDATE**

UPDATE is not a http method to update resource

iv. **PUT**

PUT used to send data to the API to create or update a resource. The difference is that PUT requests are idempotent**.**

https://api-exampleurl.com/api/v1/users/1

information regarding user with the **ID** 1 with the information sent in the request**.**

b. Oauth2 is best for web service and secure

Two authentication endpoints: the authorization endpoint and the token request endpoint. These endpoints are
https://example-api.com/oauth2/authorize and
https:// example-api. com /oauth2/token

```
<?php
$curl = curl_init( 'https://example-api..com/oauth2/token' );
curl_setopt( $curl, CURLOPT_POST, true );
curl_setopt( $curl, CURLOPT_POSTFIELDS, array(
   'client_id' => 1234,
   'client_secret' => 1234fr4rer53,
```

```
    'grant_type' => 'pas23er5',
    'username' => 12333,
    'password' =>34223323,
) );
curl_setopt( $curl, CURLOPT_RETURNTRANSFER, 1);
$auth = curl_exec( $curl );
$auth = json_decode($auth);
$access_key = $auth->access_token;
```

Making an api call

```
<?php
$access_key = 'API_TOKEN';
$curl = curl_init( 'https://example-api.com/rest/v1/me/' );
curl_setopt( $curl, CURLOPT_HTTPHEADER, array( 'Authorization: Bearer ' . $access_key ) );
curl_exec( $curl );
?>
```

C. json is best returned by an API as a response

json easy-to-parse and lightweight data-interchange format. it's easier to read and write compare to xml.example

{ "users":[{"id":"1","username":"hi","password":"jk14"}]}

Section 2: Simple Checkout System

```
<?php

class Checkout{

    protected $total = 0;
    protected $ipd = 0;
    protected $mbp = 0;
```

```
        protected $atv = 0;
        protected $vga = 0;
        protected $pricingRules = "";

        public function __construct($jsonPricingRules = "")
  {
                $this->pricingRules =  (array) json_decode($jsonPricingRules);
  }

        public function scan($item){
                if($item == 'ipd') $this->ipd++;
                if($item == 'mbp') $this->mbp++;
                if($item == 'atv') $this->atv++;
                if($item == 'vga') $this->vga++;
        }

        public function total()
        {
                $this->total += ($this->atv - floor($this->atv/3))*$this->pricingRules['atv'];
                $this->total += $this->ipd  * (($this->ipd >= 4)?($this->pricingRules['ipd']-50):$this->pricingRules['ipd']);
                $this->total += ($this->vga - ($this->mbp>0?($this->mbp<=$this->vga?$this->mbp:$this->vga):0)) * $this->pricingRules['vga'];
                $this->total += $this->mbp * $this->pricingRules['mbp'];
                return $this->total;
        }

}
$jsonPricingRules = '{"ipd":"549.99","mbp":"1399.99","atv":"109.5","vga":"30"}';
$co = new Checkout($jsonPricingRules);

$co->scan('atv');
```

```
$co->scan('ipd');

$co->scan('ipd');

$co->scan('atv');

$co->scan('ipd');

$co->scan('ipd');

$co->scan('ipd');


echo $co->total();


?>
```