**Section 1: Application Programming Interface**

Using simple code, explain what kind of situations would you use the methods:

i. GET

ii. POST

iii. UPDATE

iv. PUT

GET: This method is used for getting data from the server. The data may be anything a HTML document, an image, XML file etc

// List documents in a collection

$response=example\_rest("GET"," www.example.com/api/document","collection=test");

POST: This method is used to create a new resource or to update an existing resource in the server.  
// Create a document

$response = example\_rest(“POST”,”www.example.com/api/document ","collection=test",'{ "hello":"World" }');

PATCH/UPDATE This method can be used to update partial resources. For instance, when you only need to update one field of the resource, we can use this  
utilizing less bandwidth.  
$document\_handle= $response['documents'][0];  
$response=example\_rest("PATCH",$document\_handle,"",'{ "hello" : "World of mine" }');

PUT: PUT is often utilized for update capabilities. Use PUT whenever you need to update a resource completely through a specific resource. For instance, if you know that an article resides at http://example.org/article/1234, you can PUT a new resource representation of this article directly through a PUT on this URL. PHP doesn’t have $\_PUT like it does in $\_POST, Here’s how to access the content of a PUT request in PHP:

$\_PUT = array();

if($\_SERVER['REQUEST\_METHOD'] == 'PUT') { parse\_str(file\_get\_contents('php://input'), $\_PUT); }

b. Explain in your own words, what kind of “authentication” works best for a web service

that needs to be secure, yet easy to implement across different programming languages.

You may want to give an example of how to call this API.

b.

There are many methods to authenticate an API securely and every method has its pros and cons. One of those methods is JWT authentication. It is becoming very popular these days. JWT is a type of token-based authentication. For every single request from a client to the server, a token is passed for authentication. It supports the stateless API calls.In this, a token is encoded from a data payload using a secret key. That token is passed to the client. Whenever the client sends that token along with a request, the server validates it and sends back the response. JWT has many advantages that is making it extremely popular nowdays. It doesn’t use any database access so it reduces load from database server and make your application faster to response. Development time of JWT is faster as developer can use its existing libraries. It uses the Json format thus its parser is easily available across all programming languages hence its cross platform compatibility is also very good.

We need to download and call its library within our application .

STEP 1: we can include below files to encode JWT

include\_once 'config/core.php';  
include\_once 'libs/php-jwt-master/src/BeforeValidException.php';  
include\_once 'libs/php-jwt-master/src/ExpiredException.php';  
include\_once 'libs/php-jwt-master/src/SignatureInvalidException.php';  
include\_once 'libs/php-jwt-master/src/JWT.php';  
use \Firebase\JWT\JWT;

STEP2: Generating JSON access token

// check if email exists and if password is correct  
if($email\_exists && password\_verify($data->password, $user->password)){  
   
 $token = array(  
 "iss" => $iss,  
 "aud" => $aud,  
 "iat" => $iat,  
 "nbf" => $nbf,  
 "data" => array(  
 "id" => $user->id,  
 "firstname" => $user->firstname,  
 "lastname" => $user->lastname,  
 "email" => $user->email  
 )  
 );  
   
 // set response code  
 http\_response\_code(200);  
   
 // generate jwt  
 $jwt = JWT::encode($token, $key);  
 echo json\_encode(  
 array(  
 "message" => "Successful login.",  
 "jwt" => $jwt  
 )  
 );  
   
}

STEP 3: Setting up core configuration file of JWT config/core.php

<?php  
// show error reporting  
error\_reporting(E\_ALL);  
   
// set your default time-zone  
date\_default\_timezone\_set('Asia/Kolkata);  
   
// variables used for jwt  
$key = "abc\_key";  
$iss = "http://abc.in";  
$aud = "http://abc.in";  
$iat = 1354567437;  
$nbf = 1477000000;

?>

STEP 4: creating a file to Validate token

api/validate\_token.php

<?php  
// required headers  
header("Access-Control-Allow-Origin: http://localhost/rest-api-authentication-example/");  
header("Content-Type: application/json; charset=UTF-8");  
header("Access-Control-Allow-Methods: POST");  
header("Access-Control-Max-Age: 3600");  
header("Access-Control-Allow-Headers: Content-Type, Access-Control-Allow-Headers, Authorization, X-Requested-With");

?>

STEP 5: files to decode JWT

include\_once 'config/core.php';  
include\_once 'libs/php-jwt-master/src/BeforeValidException.php';  
include\_once 'libs/php-jwt-master/src/ExpiredException.php';  
include\_once 'libs/php-jwt-master/src/SignatureInvalidException.php';  
include\_once 'libs/php-jwt-master/src/JWT.php';  
use \Firebase\JWT\JWT;

STEP 6: retrieve given JWT

// get posted data  
$data = json\_decode(file\_get\_contents("php://input"));  
   
// get jwt  
$jwt=isset($data->jwt) ? $data->jwt : "";  
   
STEP 7: Decode JWT

// if jwt is not empty  
if($jwt){  
   
 // if decode succeed, show user details  
 try {  
 // decode jwt  
 $decoded = JWT::decode($jwt, $key, array('HS256'));  
   
 // set response code  
 http\_response\_code(200);  
   
 // show user details  
 echo json\_encode(array(  
 "message" => "Access granted.",  
 "data" => $decoded->data  
 ));  
   
 }  
   
}

c. What kind of format is best to be returned by an API as a response? Explain your

answer and provide comparisons, if needed.

There are many format that can be return by an API as a response. Few of them are listed below:

1. XML
2. JSON
3. CSV
4. HTML

In the old days, web services used XML as their primary data format as a response, but since JSON appeared it has been the preferred format because it is much more lightweight, much more simpler and has very concise format as compared to XML or any other format.

It is a way of serializing in such a way, that it becomes JavaScript code. When executed, this code creates and returns a JavaScript object which contains the data you serialized.

For one, it's a comfortable way to pass data from your server backend to your JavaScript code. Due to its simple structure parsing it is much more faster and reading it is much more easier. Its parser is also available for almost every programming languages and hence passing data between two independent platforms is also possible.

Many programming languages has built in function to easily encode and decode it to their own data structure.

**Section 2: Simple Checkout System**

Digi-X is starting a computer store. You have been engaged to build the checkout system.

We will start with the following products in our catalogue:

| SKU | Name | Price |

| -------- |:--------------------:| --------:|

| ipd | Super iPad | $549.99 |

| mbp | MacBook Pro | $1399.99 |

| atv | Apple TV | $109.50 |

| vga | VGA adapter | $30.00 |

As we're launching our new computer store, we would like to have a few opening day specials.

● We're going to have a 3 for 2 deal on Apple TVs. For example, if you buy 3 units of

Apple TVs, you will only pay for the price of 2 units

● The brand new Super iPad will have a bulk discount applied, where the price will drop to

$499.99 each, if someone buys more than 4 units

● We will bundle in a VGA adapter free of charge with every MacBook Pro sold

Our checkout system can scan items in any order.

(continued on next page)

The interface to our checkout looks like this (example shown in php):

php

Checkout co = new Checkout(pricingRules);

co->scan(item1);

co->scan(item2);

co->total();

Your task is to implement a checkout system that fulfils the requirements described above.

Example scenarios:

SKUs Scanned: atv, atv, atv, vga

Total expected: $249.00

SKUs Scanned: atv, ipd, ipd,