# limkokwing-interview test

This is sample code which are represents the answers of four interview questions. Each question have a module/function which can be run/test by inserting the value inside the form fields that has been provided.

## Quick start

To start with this project

* git clone <this project git link. Copy from above>
* open up the terminal/cmd
* go to application root folder
* Type composer update
* Once it done ,open env file and put your database name,username ,and password then at the terminal run "php artisan migrate" and this will create a few table which has been define at migration file `

## Documentation

### Question 1 Technical explanation

Below are the explanation of how sample code for question 1 work

a) Front-end

* This module use ajax feature in order to send a request to the back-end.

````

$("#question-1").submit(function(e){

e.preventDefault();

var data = $("#question-1").serialize();

$.ajax({

headers: {

'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')

},

type:'POST',

url: '{{route('question1')}}',

method: 'POST',

data:data,

dataType: "json",

success:function(data){

console.log(data)

$("#ans").html(data.result);

}

});

});

````

* This ajax can be found in question1.blade.php
* As you can see the function of CSRF token inside the header is to use user-specific token in all form submissions and side-effect URLs to prevent Cross-Site Request Forgeries.
* We are using the 'POST' method and send the request with json type
* Success property(which is apart of ajax property) will receive a responds from the api and allocate those responds back into selected div. In this case we display the answer in ' $("#ans").html(data.result)';

b) Back-end

Function for this module can be found in Question1 Controller.

* all the request must be declare first. In this case I declare request value inside $a for variable A , $b for variable B, and $c for variable C.

Below is how the function look like

`````````````````````````````````````````````````````````````````

public function getresult(Request $request)

{

$a= $request->var\_a;

$b= $request->var\_b;

$c= $request->var\_c;

$range=range('A', 'Z');

$step = 0;

$t=0;

$string[] = '';

for($step=0;$step<= $c;$step += $b){

if(end($string)){

$t = 0;

}

if($step%$a == 0){

$string[] .= $range[$t];

$t++;

}

else {

$string[] .= $step;

}

}

return response()->json(['result' => $string]);

}

}

````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

* $range variable is range of alphabet, we use this variable as a indicator to loop the process.
* For the first the step of the loop it will check if the character "Z" is exist at the end of array index or not. IF TRUE then we set the $t to zero .this will give $range value start from A to Z again.
* $step is a variable that I use it as a starting and the end point of the looping process. This value also will be use to be divided by the variable of $a and this will determine whether the remainder is 0 or higher.
* If the remainder is 0 then we replace $string with a $t which starting from A-Z.
* If the remainder is higher than 0 then we put $step value into it current index of $string which the result will be string.
* Lastly we will return the result in json format and send it back to the front-end.

----------------------------------------------------------------------------------------------------------------------------------

### Question 2 Technical explanation

Below are the explanation of how the answer for question 2 sample code work

a) Front -end

* For the front-end part I provide a scalable input fields which giving an option for user to add the details as many as they like.
* Below is the example of the sample code :

``````````````````````````````````````````````````````````````````````````````````````

$(function() {

var scntDiv = $('#populate');

i = $('#populate').find('#blue').size() + 1;

$("#addScnt").click(function(){

$('<div id='+i+' class="row justify-content-md-center">' +

'<div class="col-2"> ' +

'<label for="exampleFormControlInput1">Name User '+i+'</label> ' +

'<input class="form-control" id="var\_a" name="name[]" type="text" id="exampleFormControlInput1" placeholder="name@example.com">' +

'</div>' +

'<div class="col-1">' +

'<label for="exampleFormControlInput1">Age</label> ' +

'<input class="form-control" id="var\_b" name="age[]" type="text" id="exampleFormControlInput1" placeholder="name@example.com">').appendTo(scntDiv);

i++;

return false;

});

});

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

* var i is global variable that we use to access in " $("#addScnt").click(function()" as to indicate users index
* once the users click #addScnt div then this function will execute and append a new fields inside #populate div.
* if the user want to remove additional field that already has been added, users can click remove button and this will exucite function rem(s) and remove selected field.

`````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

function rem(s){

$('#'+s).remove();

i--;

}

``````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

* Whenever the user click the button remove, user index will decrease by one. Hence this will give a true a number of user is being entered.

b)Back end

* For the back end part I used the the number of $name to loop the process since the number of name can be represent as a index of array for each users .
* Hence whenever the loop is run then $name index can be use to access for another array. In this case i used the $NAME index to find the corelated age.

```` ```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

public function getresult(Request $request)

{

$name = $request->input('name');

$age = $request->input('age');

$key = $request->input('key');

$country = $request->input('country');

$state = $request->input('state');

$i=0;

foreach($name as $index => $input) {

$user[] = array('name' => $input, 'age' => $age[$index], 'key' => array\_slice($key,$i,3));

$i+=3;

}

foreach($country as $index => $input) {

$address[] = array('country' => $input, 'state' => $state[$index]);

}

return response()->json(['user'=>$user,'address'=>$address]);

}

}

``````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

* In order to find the value of the 'key' I used 'array\_slice' function in order to select particular key array. For instance , in this case I use an $i as a starting point of the selected array. Then '3' indicate as which position that I would like to take into an array. Ind this case I mae it 3 because we have the maximumof 3 input box for key element ,if we have more than 3 then we can put more than that.
* Next , I use the same method to structure country and state array. In these case I use country to loop the process and use country index to access another array.

once everything is done then we return user and address array in json format.

-----------------------------------------------------------------------------------------------------------------------

question 3

Answer for question 3 as below

* "select a.name,c.freq from clients a inner join (select b.user\_id,count(\*) as freq from orders b group by b.user\_id)c

on a.id = c.user\_id"

-----------------------------------------------------------------------------------------------------------------------

Question 4

* In this part I would like to focus more on crud process ,security ,as well as validation control mechanism.

A) introduction

* This module consist of create ,read ,update ,and delete function.
* In this function i use laravel resource default method to handle a variety of actions on the resource. The generated controller will already have methods stubbed for each of these actions, including notes informing you of the HTTP verbs and URIs they handle.
* Below is how i define route using resource approach.

```````````````````````````````````````````````````````````````````````````````````````````````````````

Route::resource('crud','CrudsController');

``````````````````````````````````````````````````````````````````````````````````````````````````````

* Note ,if you like to know the detail of every existing route that you have in your project ,you can simply type "php artisan route:list". in terminal.

B) validation control

front-end validation control

* For the front end form validation , i prefer to use parsley.js in order to control the validation for every single element field inside the form
* In this sample code I use require , date-format,numeric and phone-number format as a validation .
* Below is a parsely attribute which can be found in input type.

``````````````````````````````````````````````````````````````````````````````````

data-parsley-required

````````````````````````````````````````````````````````````````````````````````

Back-end validation control

* Using a laravel default validation in order to control back-end validation
* we can put this validation inside middleware. But in this case , I put it inside the controller so that we can easily work on it. Below is how to define validation for particular request.

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

$request->validate([

'student\_name' => 'required',

'date\_of\_birth' => 'required|date',

'nationality' => 'required',

'phone' => 'required',

'image' => 'required|image|max:2048'

]);

````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

C) CRUD PROCESS

Create method

* Once the request has been validate from the front-end as well as the back-end now its Turn for the create method to take over the process.
* To save it into the table its good to have a model for a particular table first.in this case i name the model 'Student' and use this model to do crud operation.
* Below is how do I declare crud model and use it to save data.

``````````````````````````````````````````````````````````````````

use App\Students;

public function store(Request $request)

{

$request->validate([

'student\_name' => 'required',

'date\_of\_birth' => 'required|date',

'nationality' => 'required',

'phone' => 'required',

'image' => 'required|image|max:2048'

]);

$image = $request->file('image');

$new\_name = rand() . '.' . $image->getClientOriginalExtension();

$image->move(public\_path('images'), $new\_name);

$form\_data = array(

'student\_name' => $request->student\_name,

'date\_of\_birth' => $request->date\_of\_birth,

'nationality' => $request->nationality,

'phone' => $request->phone,

'image' => $new\_name

);

//dd( $form\_data);

Students::create($form\_data);

return redirect('crud')->with('success', 'Data Added successfully.');

}

``````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

update operation

* Same goes to update operation ,we need to create a validation first then pass it to the controller in order to update selected row.
* In order to update for particular row we need to use method post or patch and we need to pass an ID that represent which row that it would like to update. Below is the example of how do we define a route inside form element .

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

<form method="post" action="{{ route('crud.update', $data->id) }}" enctype="multipart/form-data">

`````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

Delete operation

* Delete operation use a GeT method.
* In this case we use soft delete approach which mean we are not really delete it but we mark a date inside deleted at column.

below is how we make delete function code

```````````````````````````````````````````````````````````````````````````````````````````````````````

public function destroy($id)

{

$data = Students::findOrFail($id);

$data->delete();

return redirect('crud')->with('success', 'Data is successfully deleted');

}

```````````````````````````````````````````````````````````````````````````````````````````````````````````````

D) security aspect.

* Define a column name as a Fillable inside a model. This will allow data to be save if only the name of the parameter request same as define inside Fillable.

``````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

protected $fillable = ['student\_name', 'phone', 'date\_of\_birth','image','nationality'];

```````````````````````````````````````````````````````````````````````````````````````````````````````````````````````

CSRF.

* use " \App\Http\Middleware\VerifyCsrfToken::class," for every incoming post method request

Styling approach

* Using bootstrap 4.1.0 to customize layout design.

* Using Blade feature such as yield and extend function in order to structure the design layout.

* Use customize css file which cab be found in public/css/app.css.