UNIVERSITI TUNKU ABDUL RAHMAN

ACADEMIC YEAR 2020/2021

JANUARY 2021 TRIMESTER

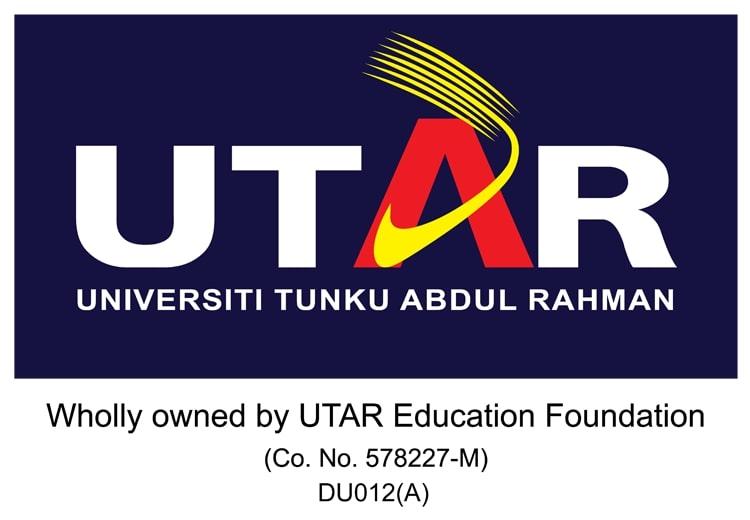
FINAL ASSESSMENT

**ANSWER SCRIPT**

**Candidate is required to fill in ALL the information below:**

|  |  |  |  |
| --- | --- | --- | --- |
| Name : (as stated in Student Identity Card) | TAN YING YAO 1703648 | | |
| Faculty /Institute/ Centre: | LKC FES | Programme : | SE |
| Index No.  (in numbers) : | U00584EBSEF | Index No.  (in words) : | U-ZERO-ZERO-FIVE-EIGHT-FOUR-EBSEF |
| Course Code : | UECS3294 | Course Description : | ADVANCED WEB APPLICATION DEVELOPMENT |
| Submission Date : | 27/04/21 | Time : | 1400PM |

|  |  |  |
| --- | --- | --- |
| **QUESTION NUMBER** | **FOR EXAMINER’S USE ONLY** | |
| **MARKS** | |
| **Internal** | **External** |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
|  |  |  |
| **TOTAL MARKS** |  |  |



**DECLARATION STATEMENT**

I, **TAN YING YAO** (Name), Student ID No. **1703648** hereby solemnly and sincerely declare and confirm that I have read, understood and shall abide and comply with all laws, rules, regulations, guidelines and lawful instruction of the University and its staff in relation to the commencement of any assessment / examination during my programme of study in Universiti Tunku Abdul Rahman.

I hereby declare that my submission for all assessment / examination during my programme of study in the University shall be based on my original work, not plagiarised from any source(s) except for citations and quotations which have been duly acknowledged. I am fully aware that students who are suspected of violating this pledge are liable to be referred to the Examination Disciplinary Committee of the University.

TANYINGYAO

Name: TAN YING YAO

Student’s I.C. / Passport No.: 971210-14-5673

Date: 27/04/21

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 1

1)a)

Eloquent ORM (Object-Relational Mapping) is a complex PHP implementation of the active-record pattern which provide internal methods for enforcing constraints on the relationships between databases. Eloquent ORM presents database tables as classes with their object instances tied to single table data rows. Each table has a corresponding model which is used to interact with the table stated.

ORM can help access and manage data systems easily as an extension of the current work. This allows developers to work faster due to its robust nature and reduce the chance of going through the entire storage backend. ORM also comes with migration tools that can manage database schema easily. This can help track changes and modification made to the database quickly and efficiently.

For example, you can easily retrieve all records of a database by just using function declarations:

**$records = Record::all();**

You can also filter the query by adding conditions:

**$records = Record::where(‘price’, ‘>’, 150)->take(10)->get;**

ORM also allows easy definable attributes:

**class Record extends Model {**

**Protected $fillable = [‘product’, ‘price’, ‘id’];**

**}**

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 2

(b)

There are three ways to access database: Raw SQL, Query Builder, ORM

Raw SQL requires the developer to write and manage the database queries manually. This can result in a lot of work but is very performance friendly when compared to generated SQL such as Query builder. Manually writing SQL can help improve performance as query builders may have unnecessary declarations, overly broad queries and other mistakes that can lead to a slowdown in database operation which result in a difficulty to debug. Raw SQL is also much more flexible. More complex queries can be written to take full advantage of the database engine.

Query builder provides a more direct database access alternative to Raw SQL. Instead of requiring SQL queries to be written directly, Laravel's query builder provides a set of classes and methods capable of building queries through coding. It also allows selectable caching of the results of executed queries. The advantage of using a Query builder is that it uses methods instead of manually written strings, it is fairly transparent and is more robust for those familiar with the database. Query Builder can also support multiple back-end operations of data. This results in the easy migration of databases.

It does not need to be mutually exclusive as both can be used to manage a database for ease of access of operations. Both of these operations have their own advantages and disadvantages but can be used together to a great extent.

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 3

(c)

Here are the architecture of the Laravel MVC framework:

i) Submit User Request

The Request is directed according to the route defined

**route(web.php)**

**route::get(‘productList’.[ProductController::class,’getProducts;]);**

ii) Route directs to the ProductController

**ProductController.php**

**public function getProducts(){**

**$request = Product::paginate(5);**

**return view(‘products, [‘products’ => $data]);}**

iii) Interact with Products Model and retrieve data from table from the database

**Product.php**

**Class Product extends Model {  
use HasFactory;}**

iv) Controller invokes the view from the Model

**products.blade.php**

**@foreach($products as $product)**

**<li>{{$product}}</li> @endforeach**

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 4

2)a)

**AuthServiceProvider.php**

public function boot()

{ $this->registerPolicies();

/\* define an Administrator role \*/

Gate::define('isAdmin', function($user) {

return $user->role == 'admin';

});

/\* define an Subscriber role \*/

Gate::define('isSubscriber', function($user) {

return $user->role == 'subscriber';

});

/\* define a user role \*/

Gate::define('isGuest', function($user) {

return $user->role == 'guest';

}); }

**Home.blade.php**

<div class="card-body">

Welcome to Laravel Web Application.

@if (session('status'))

<div class="alert alert-success" role="alert">

{{ session('status') }}

</div>

@endif

@can('isAdmin')

<div class="posts">

You have Admin Access. Admin can create, edit, delete and view all posts.

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 5

</div>

@elsecan('isSubscriber')

<div class="posts">

You have Subscriber Access. Subscriber can create and edit posts.

</div>

@else

<div class="posts ">

You have User Access. Users can only view all posts.

</div>

@endcan

</div>

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 6

(b)

**PostController.php**

<?php

namespace App\Http\Controllers;

use Illuminate\Support\Facades\Gate;

use Illuminate\Http\Request;

class PostController extends Controller

{ public function create() {

if (Gate::allows('isAdmin')) {

dd('Admin allowed to create new posts');

} else { dd('You are not an Admin'); } }

public function update()

{ if (Gate::allows('isAdmin’)) {

dd('Author allowed to update posts');

} else { dd('You are not an Admin'); } }

public function delete()

{ if (Gate::allows('isAdmin')) {

dd('Admin allowed to delete posts'); } else {

dd('You are not Admin'); }}}

**web.php**

use Illuminate\Support\Facades\Gate;

Route::get('/posts/create', [PostController::class, 'create'])-

>middleware('can: isAdmin')->name('post.create');

Route::get('/posts/update', [PostController::class, 'update'])-

>middleware('can: isAdmin')->name('post.update');

Route::get('/posts/delete', [PostController::class, 'delete'])-

>middleware('can:isAdmin')->name('post.delete');

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 7

(c)

1. via a **Request** instance

We can access the session via a Request instance, which is type-hinted on a route or controller method. When retrieving an item from the request, we can also pass values using the **get** method. The default value will be returned if the specified key does not exist in the instance. If passing a closure as the default value to the **get** method and the requested key does not exist, the closure will be executed and the result returned.

1. The global **session** helper

We can also use the global session PHP function to retrieve and store data in the session. When the session helper is called with a single, string argument, it will return the value of the session key. When the helper is called with an array of key/ value pairs, those value will be stored in the session.

Both are useful tool that can be used to retrieve and access data quickly and efficiently. They provide an easy way to request data and can help streamline the fetching of data in an application. Global session PHP function can be used to retrieve and store data in the session.

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 8

Q3. (a)

<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;

use App\Models\User;

use Validator;

class UserController extends Controller

{ /\*\* \* Create a new UserController instance.

\* \* @return void \*/

public function \_\_construct() {

$this->middleware('auth:api', ['except' => ['login', 'register']]);

}

/\*\* \* Get a JWT via given credentials.

\* \* @return \Illuminate\Http\JsonResponse \*/

public function login(Request $request){

$validator = Validator::make($request->all(), [

'email' => 'required|email',

'password' => 'required|string|min:6',

]);

if ($validator->fails()) {

return response()->json($validator->errors(), 422);

}

if (! $token = auth()->attempt($validator->validated())) {

return response()->json(['error' => 'Unauthorized'], 401);

}

return $this->createNewToken($token);

}

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 9

/\*\* \* Register a User.

\* \* @return \Illuminate\Http\JsonResponse \*/

public function register(Request $request) {

$validator = Validator::make($request->all(), [

'name' => 'required|string|between:2,100',

'email' => 'required|string|email|max:100|unique:users',

'password' => 'required|string|confirmed|min:6',

]);

if($validator->fails()){

return response()->json($validator->errors()->toJson(), 400);

}

$user = User::create(array\_merge(

$validator->validated(),

['password' => bcrypt($request->password)]

));

return response()->json([

'message' => 'User successfully registered',

'user' => $user

], 201);

}

/\*\* \* Log the user out (Invalidate the token).

\* \* @return \Illuminate\Http\JsonResponse \*/

public function logout() {

auth()->logout();

return response()->json(['message' => 'User successfully signed out']);

}

/\*\* \* Refresh a token.

\* \* @return \Illuminate\Http\JsonResponse \*/

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 10

public function refresh() {

return $this->createNewToken(auth()->refresh());

}

/\*\* \* Get the authenticated User. \*

\* @return \Illuminate\Http\JsonResponse \*/

public function userProfile() {

return response()->json(auth()->user());

}

/\*\* \* Get the token array structure. \*

\* @param string $token \*

\* @return \Illuminate\Http\JsonResponse \*/

protected function createNewToken($token){

return response()->json([

'access\_token' => $token,

'token\_type' => 'bearer',

'expires\_in' => auth()->factory()->getTTL() \* 21600, //**expires in 15 days**

'user' => auth()->user()

]);

}

}

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 11

(b)

<?php

use App\Http\Controllers\UserController;

Route::group([

'middleware' => 'api',

'prefix' => 'auth'

], function ($router) {

Route::get('post', [UserController::class, 'index']);

Route::post('/post', [UserController::class, 'store']);

Route::put('post/{id}', [UserController::class, 'update']);

Route::delete('post/{id}', [UserController::class, 'destroy']);

Route::post('/login', [UserController::class, 'login']);

Route::post('/register', [UserController::class, 'register']);

Route::post('/logout', [UserController::class, 'logout']);

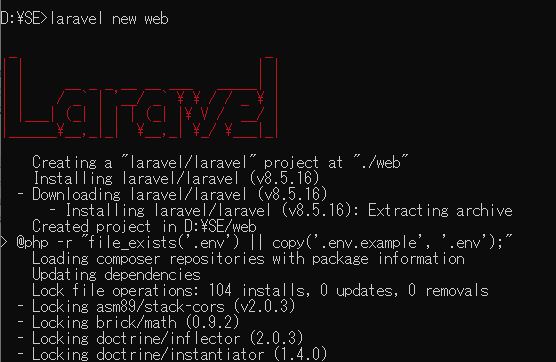
Route::post('/refresh', [UserController::class, 'refresh']);

Route::get('/user-profile', [UserController::class, 'userProfile']);

});

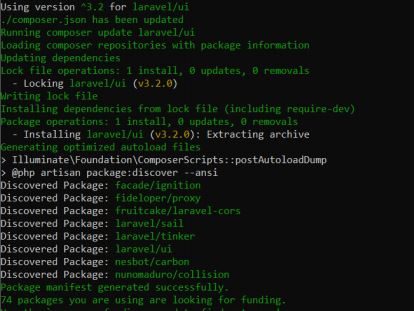
Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 12

Q4. (a)

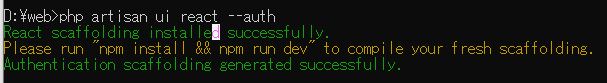
**run the command to initiate a new project:**laravel new web

Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 13

**run the command:**composer require laravel/ui --dev



**run the command to enable React Scaffoliding:**php artisan ui react --auth



**Declare API endpoints for CRUD function**

Route::middleware(‘auth:api’)->get(‘/user’, function (Request $request){

Return $request->user();});

Route::get(‘posts’, [ProductsController::class, ‘index’]);

Route::post(‘posts’, [ProductsController::class, ‘store’]);

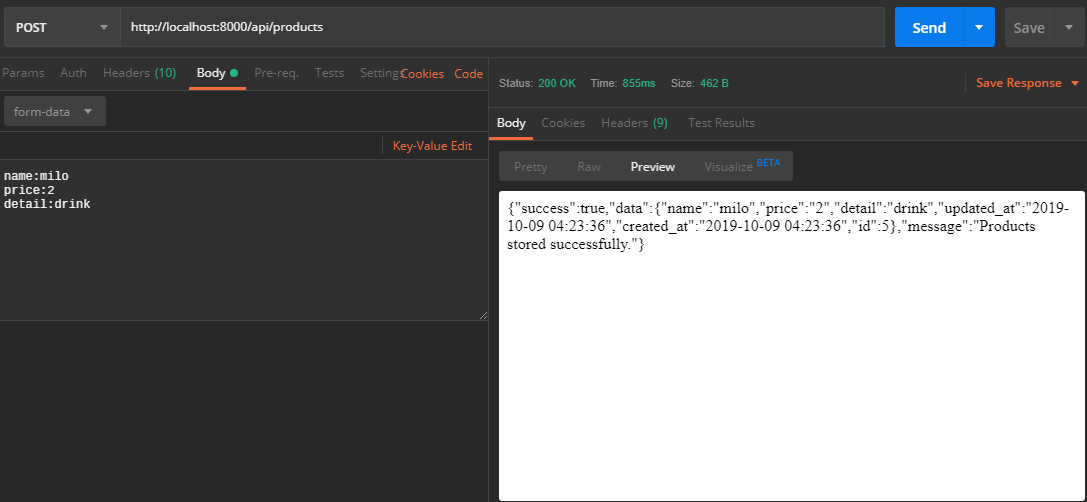
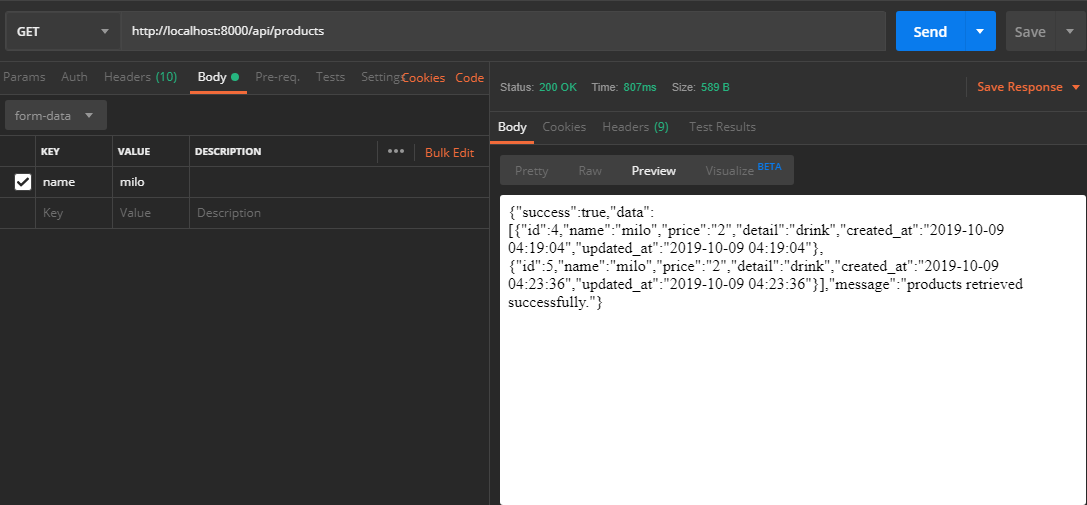
Route::get(‘posts/{id}’, [ProductsController::class, ‘update’]);

Route::get(‘posts/{id}’, [ProductsController::class, ‘destroy’]);

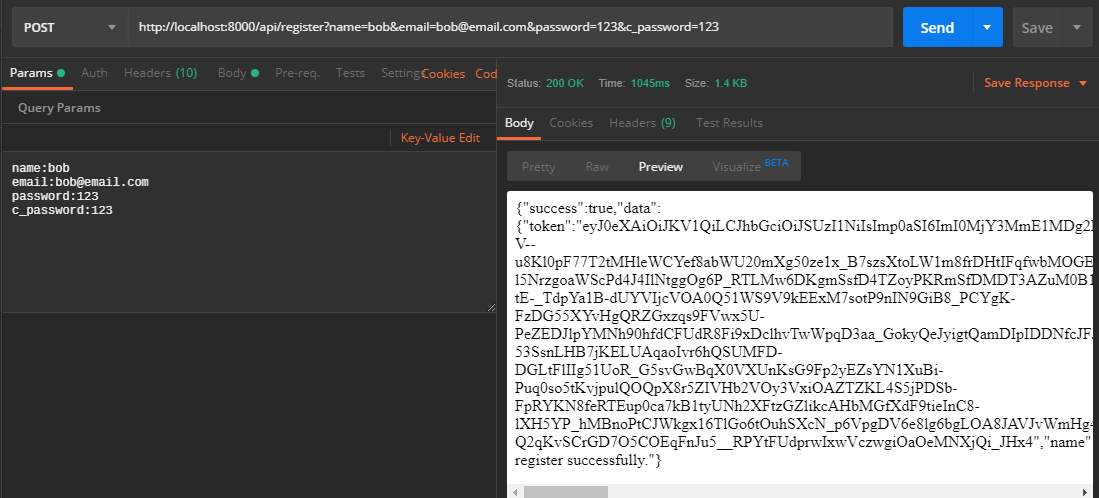
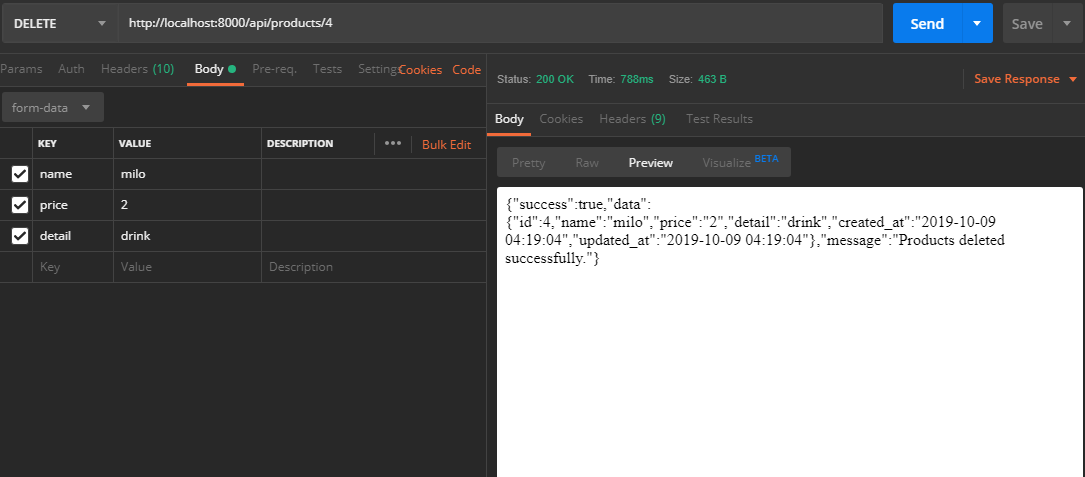
Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 14

**Validate API**

Finally we can test the API using Postman



Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 15



Index Number (in figure): U00584EBSEF Course Code: UECS3294 Page: 16

(b)

REST represent a set of rules that developer follows when creating an API. It states that one should be able to fetch data when one link to a specific URL. Each URL is called a request and the data sent back is refer to as a response. A RESTful interface has many advantages such as scalability. The difference between client and server allows the web application to be scaled by the developer with ease.

RESTful interface is also flexible and portable. It is possible to perform migration from one server to another and carry out changes to the database at any given time. Therefore, both front-end and back-end can be hosted on different servers which provides a significant ease of management.

The RESTful API also is largely independent which differentiates the server and client and allow for easy development across all area of the application independently. The REST API can adapt well to the application and offers opportunity to attempt different environments while developing.