UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

Practical 9 : Laravel REST API Authentication with JWT.

With knowledge of creating a Laravel web application with Laravel Framework Backend and React Frontend as well as creating RESTful API for the backend (server) to exchange information with frontend (client) we will learn how to create secure REST APIs in Laravel using JSON Web Token (JWT).

With JWT, user information such as username and password are sent to web-server using HTTP GET and POST requests. Web server identifies user information and generates a JWT Token and sends it back to client. Client stores that token into session and also set it to the header. On next HTTP requests, that token is verified by the server, which returns the response to the client.

1. Install and Configure JWT Authentication Package.

First, install the JWT package through composer as follows:

```
composer require tymon/jwt-auth
```

Then, register the JWT Authentication Package in config/app.php in providers and aliases array as shown in the following codes/scripts segment.

```
'providers' => [
    ...
    ...
    Tymon\JWTAuth\Providers\LaravelServiceProvider::class,
],
'aliases' => [
    ...
    'JWTAuth' => Tymon\JWTAuth\Facades\JWTAuth::class,
    'JWTFactory' => Tymon\JWTAuth\Facades\JWTFactory::class,
    ...
],
```

The next command in command line publish the package's configuration by copying the JWT Auth files from vendor folder to config/jwt.php file.

```
php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
```

Generate a secret key to handle token encryption (will be found in .env file) by the following command:

```
php artisan jwt:secret
```

2. Define User Model.

Let's define User model to implement JWT-Auth package in the User model as shown in Figure 1.

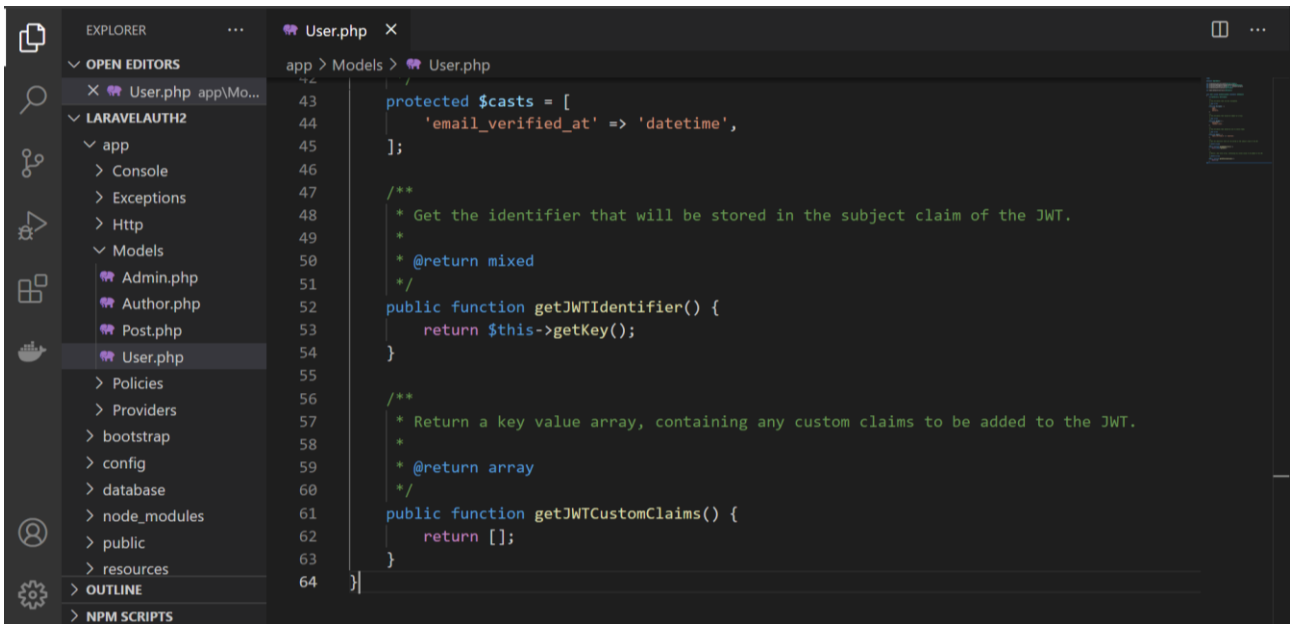


Figure 1: Define Tymon\JWTAuth\Contracts\JWTSubject contract before the User model.

```php
<?php

namespace App\Models;

use Illuminate\Contracts\Auth\MustVerifyEmail;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Foundation\Auth\User as Authenticatable;
use Illuminate\Notifications\Notifiable;

use Tymon\JWTAuth\Contracts\JWTSubject;

class User extends Authenticatable implements JWTSubject
{
    use HasFactory, Notifiable;

    /**
     * The attributes that are mass assignable.
     *
     * @var array
     */
    protected $fillable = [
        'name',
        'email',
        'password',
    ];
    /**
     * The attributes that should be hidden for arrays.
     *
     * @var array
     */
    protected $hidden = [
        'password',
        'remember_token',
    ];
    /**
```

```
     * The attributes that should be cast to native types.
     *
     * @var array
     */
    protected $casts = [
        'email_verified_at' => 'datetime',
    ];
    /**
     * Get the identifier that will be stored in the subject claim of the JWT.
     *
     * @return mixed
     */
    public function getJWTIdentifier() {
        return $this->getKey();
    }
    /**
     * Return a key value array, containing any custom claims to be added to
the JWT.
     *
     * @return array
     */
    public function getJWTCustomClaims() {
        return [];
    }
}
```

3. Define Auth Guard.

Set up the JWT Auth Guard to secure the Laravel application's authentication process by setting the defaults guard to API, and the API guards to use JWT driver as shown in the following codes/scripts segment.

```
<?php

return [

    'defaults' => [
        'guard' => 'api',
        'passwords' => 'users',
    ],


    'guards' => [
        'web' => [
            'driver' => 'session',
            'provider' => 'users',
        ],

        'api' => [
            'driver' => 'jwt',
            'provider' => 'users',
            'hash' => false,
        ],
    ],
```

4. Define Authentication Controller.

Next, create the JWT authentication controller with the following command:

```
php artisan make:controller AuthController
```

In this AuthController, define the application logics for secure authentication process as shown in the following codes/scripts segment.

```
<?php

namespace App\Http\Controllers;
use Illuminate\Http\Request;

use Illuminate\Support\Facades\Auth;
use App\Models\User;
use Validator;

class AuthController extends Controller
{
    /**
     * Create a new AuthController 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);
    }
    /**
     * 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
     */
    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() * 60,
            'user' => auth()->user()
        ]);
    }
}
```

5. Define REST API Authentication Endpoints.

Next, create REST API authentication endpoints for authentication processes in Laravel JWT Authentication application as shown in the following codes/scripts segment.

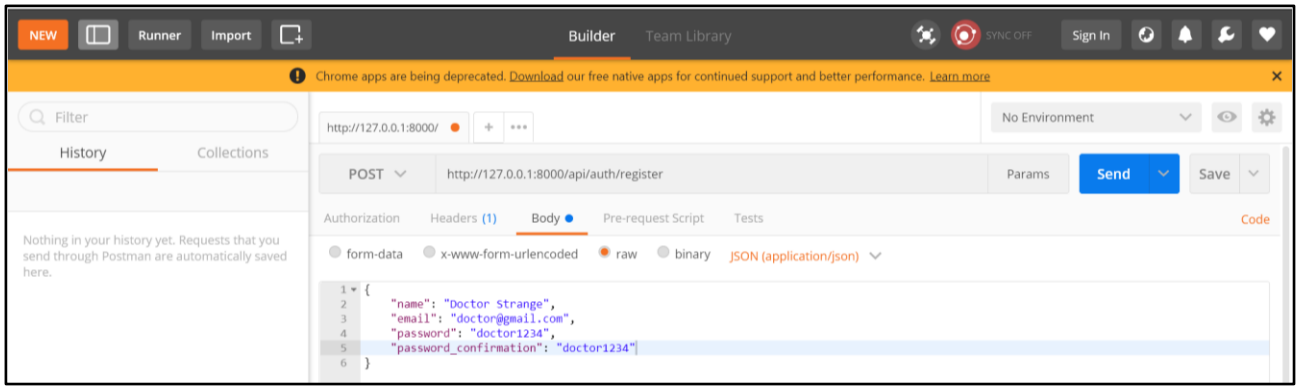
```
<?php
...
use App\Http\Controllers\AuthController;
...
Route::group([
    'middleware' => 'api',
    'prefix' => 'auth'

], function ($router) {
    Route::post('/login', [AuthController::class, 'login']);
    Route::post('/register', [AuthController::class, 'register']);
    Route::post('/logout', [AuthController::class, 'logout']);
    Route::post('/refresh', [AuthController::class, 'refresh']);
    Route::get('/user-profile', [AuthController::class, 'userProfile']);
});
```

6. Validate REST API Authentication Endpoints in Postman.

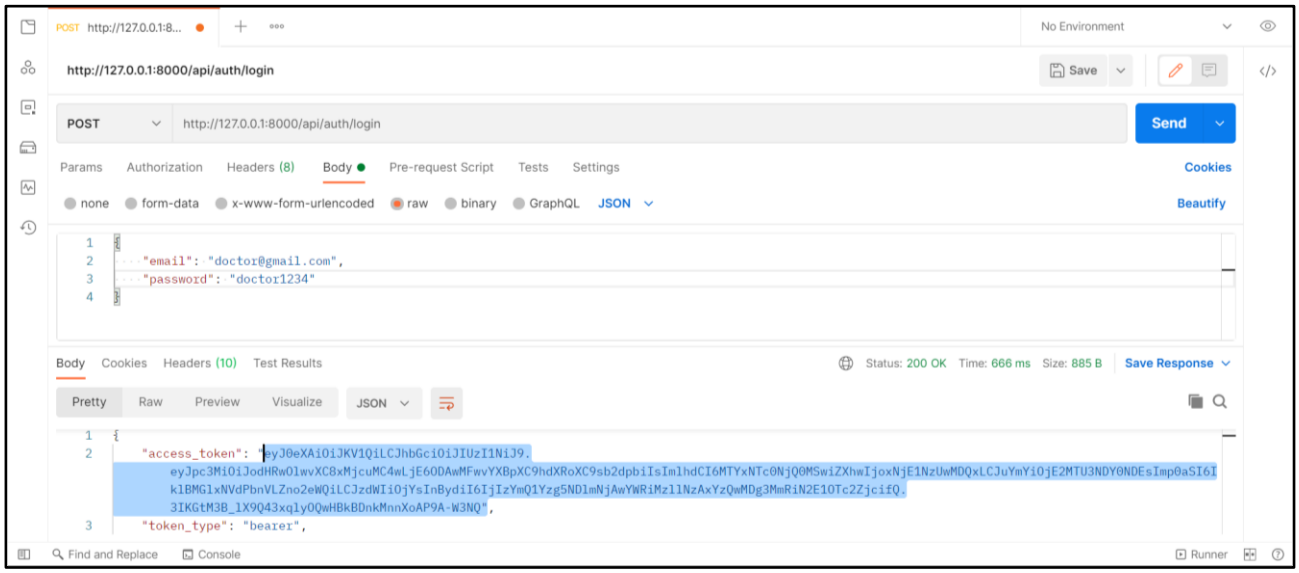
Subsequent to creation of REST API using JWT authentication, test the authentication APIs for Login, Register, User Profile, Token Refresh and Logout using Postman application.

User Registration API



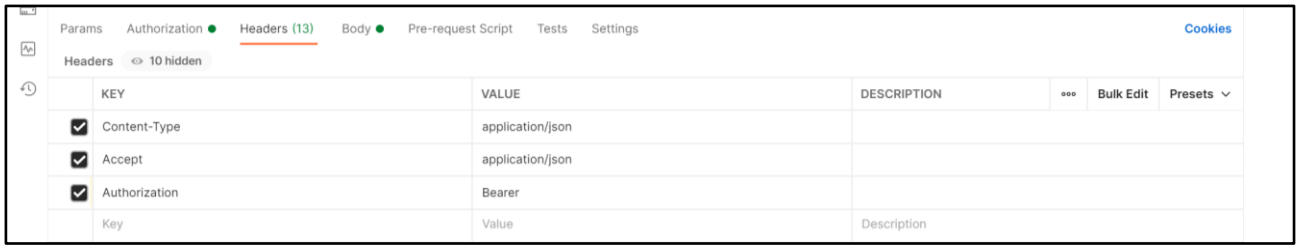
UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

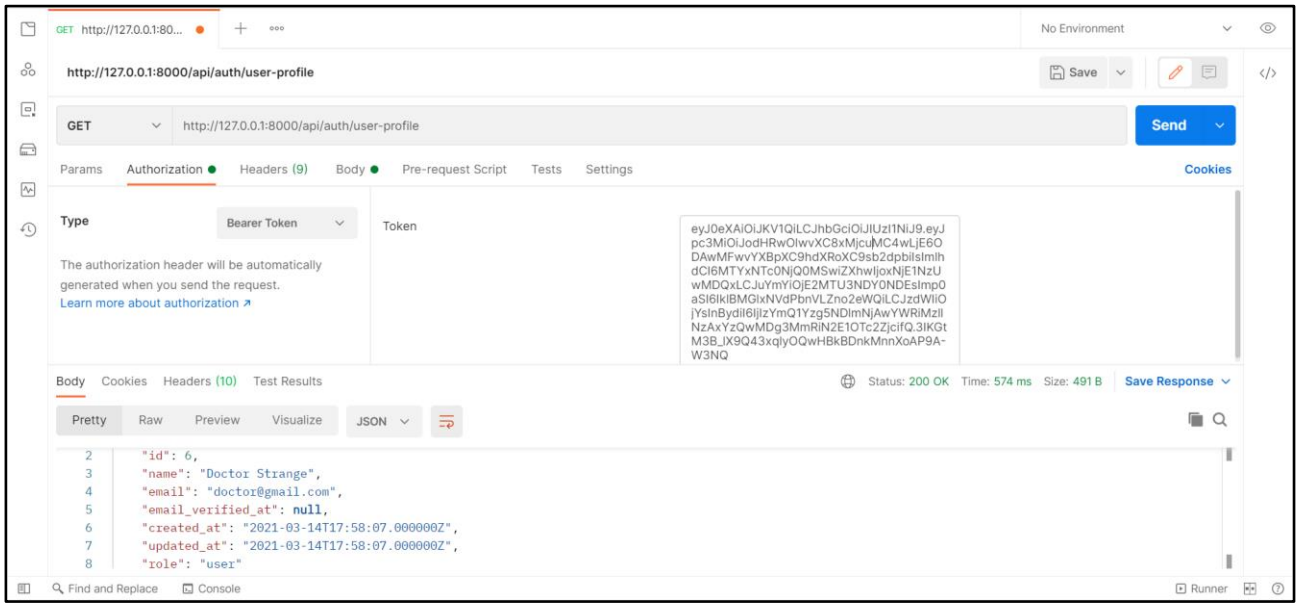
User Login API



User Profile API

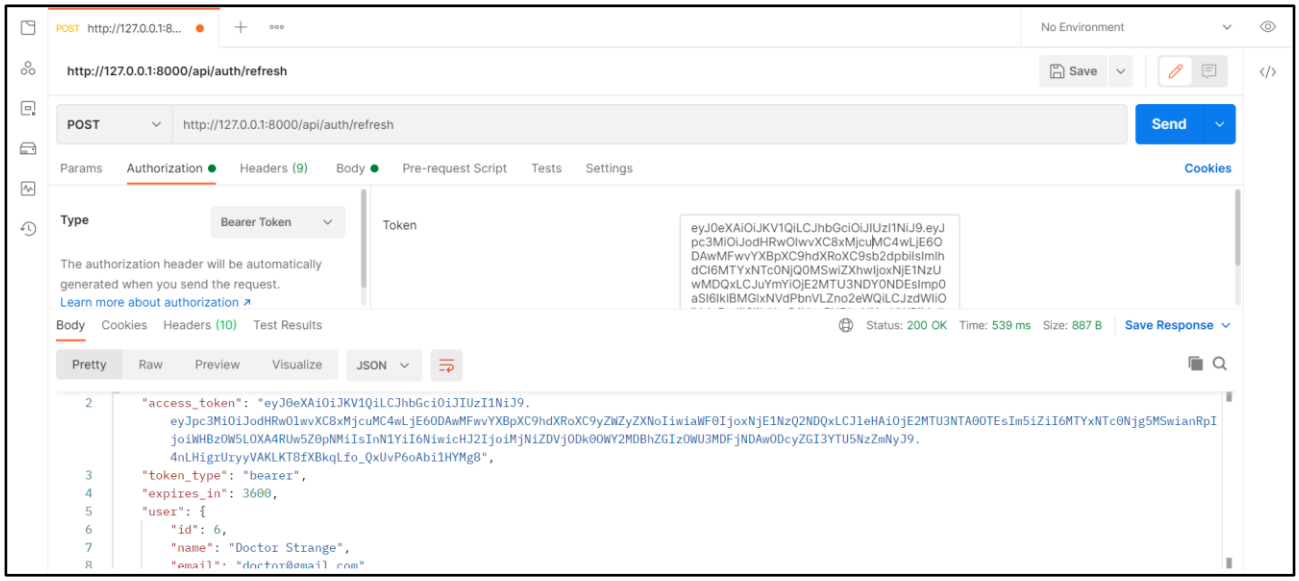
Make sure to define the access token as a header field "Authorization: Bearer" for User Profile, Token Refresh, and Logout REST APIs.





UECS3294 ADVANCED WEB APPLICATION DEVELOPMENT

JWT Token Refresh



User Logout API