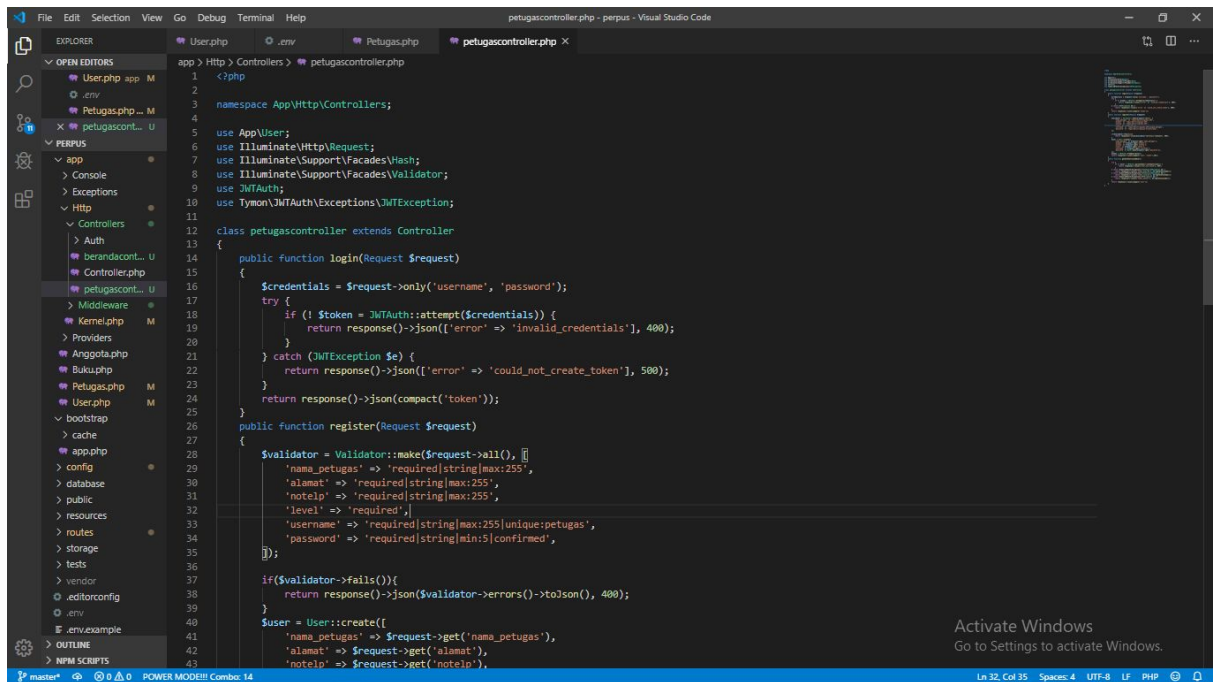


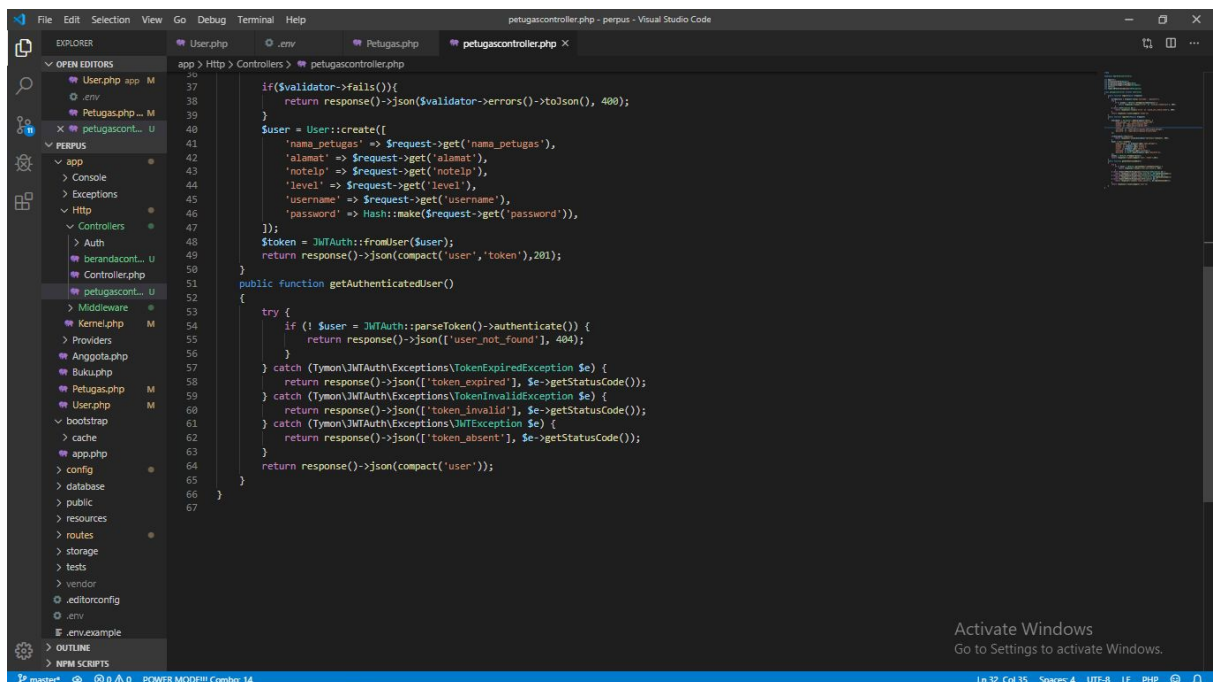
Nama : Saddam Raihan Ramadhan

No.Absen : 31

1. Controller



```
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use App\User;
6 use Illuminate\Http\Request;
7 use Illuminate\Support\Facades\Hash;
8 use Illuminate\Support\Facades\Validator;
9 use JWTAuth;
10 use Tymon\JWTAuth\Exceptions\JWTException;
11
12 class PetugasController extends Controller
13 {
14     public function login(Request $request)
15     {
16         $credentials = $request->only('username', 'password');
17         try {
18             if (! $token = JWTAuth::attempt($credentials)) {
19                 return response()->json(['error' => 'invalid_credentials'], 400);
20             }
21         } catch (JWTException $e) {
22             return response()->json(['error' => 'could_not_create_token'], 500);
23         }
24         return response()->json(compact('token'));
25     }
26
27     public function register(Request $request)
28     {
29         $validator = Validator::make($request->all(), [
30             'nama_petugas' => 'required|string|max:255',
31             'alamat' => 'required|string|max:255',
32             'notelp' => 'required|string|max:255',
33             'level' => 'required',
34             'username' => 'required|string|max:255|unique:petugas',
35             'password' => 'required|string|min:5|confirmed',
36         ]);
37
38         if ($validator->fails()) {
39             return response()->json($validator->errors()->toJson(), 400);
40         }
41
42         $user = User::create([
43             'nama_petugas' => $request->get('nama_petugas'),
44             'alamat' => $request->get('alamat'),
45             'notelp' => $request->get('notelp'),
46         ]);
47     }
48 }
```



```
36
37
38 if ($validator->fails()) {
39     return response()->json($validator->errors()->toJson(), 400);
40 }
41
42 $user = User::create([
43     'nama_petugas' => $request->get('nama_petugas'),
44     'alamat' => $request->get('alamat'),
45     'notelp' => $request->get('notelp'),
46     'level' => $request->get('level'),
47     'username' => $request->get('username'),
48     'password' => Hash::make($request->get('password')),
49 ]);
50 $token = JWTAuth::fromUser($user);
51 return response()->json(compact('user', 'token'), 201);
52 }
53
54 public function getAuthenticatedUser()
55 {
56     try {
57         if (! $user = JWTAuth::parseToken()->authenticate()) {
58             return response()->json(['user_not_found'], 404);
59         }
60     } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
61         return response()->json(['token_expired'], $e->getStatusCode());
62     } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
63         return response()->json(['token_invalid'], $e->getStatusCode());
64     } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
65         return response()->json(['token_absent'], $e->getStatusCode());
66     }
67     return response()->json(compact('user'));
68 }
```

2. Model User

```
1 <?php
2
3 namespace App;
4
5 use Illuminate\Notifications\Notifiable;
6 use Illuminate\Contracts\Auth\MustVerifyEmail;
7 use Illuminate\Foundation\Auth\User as Authenticatable;
8 use Tymon\JWTAuth\Contracts\JWTSubject;
9
10 class User extends Authenticatable implements JWTSubject
11 {
12     use Notifiable;
13
14     /**
15      * The attributes that are mass assignable.
16      *
17      * @var array
18      */
19     protected $fillable = [
20         'name', 'email', 'password',
21     ];
22
23     /**
24      * The attributes that should be hidden for arrays.
25      *
26      * @var array
27      */
28     protected $hidden = [
29         'password', 'remember_token',
30     ];
31
32     public function getJWTIdentifier()
33     {
34         return $this->getKey();
35     }
36
37     public function getJWTCustomClaims()
38     {
39         return [];
40     }
41 }
```

3. Middleware

```
1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use JWTAuth;
7 use Exception;
8 use Tymon\JWTAuth\Http\Middleware\BaseMiddleware;
9
10 class JwtMiddleware extends BaseMiddleware
11 {
12     /**
13      * Handle an incoming request.
14      *
15      * @param \Illuminate\Http\Request $request
16      * @param \Closure $next
17      * @return mixed
18      */
19     public function handle($request, Closure $next)
20     {
21         try {
22             $user = JWTAuth::parseToken()->authenticate();
23         } catch (Exception $e) {
24             if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenInvalidException){
25                 return response()->json(['status' => 'Token is Invalid']);
26             } else if ($e instanceof \Tymon\JWTAuth\Exceptions\TokenExpiredException){
27                 return response()->json(['status' => 'Token is Expired']);
28             } else{
29                 return response()->json(['status' => 'Authorization Token not found']);
30             }
31         }
32         return $next($request);
33     }
34 }
```

4. Api.php

