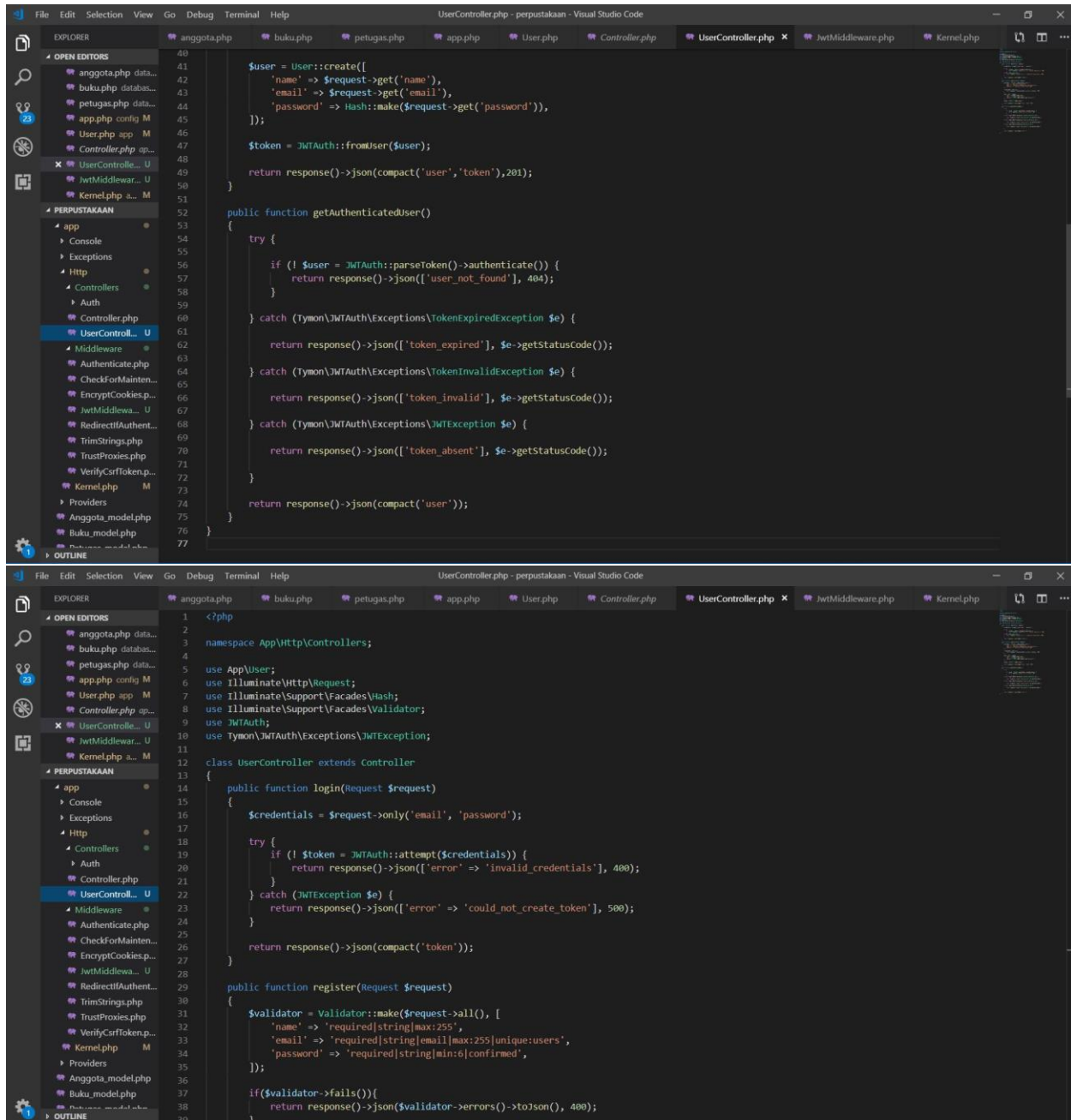


Tegar Harianto Putra (XIRPL6/39)

Script

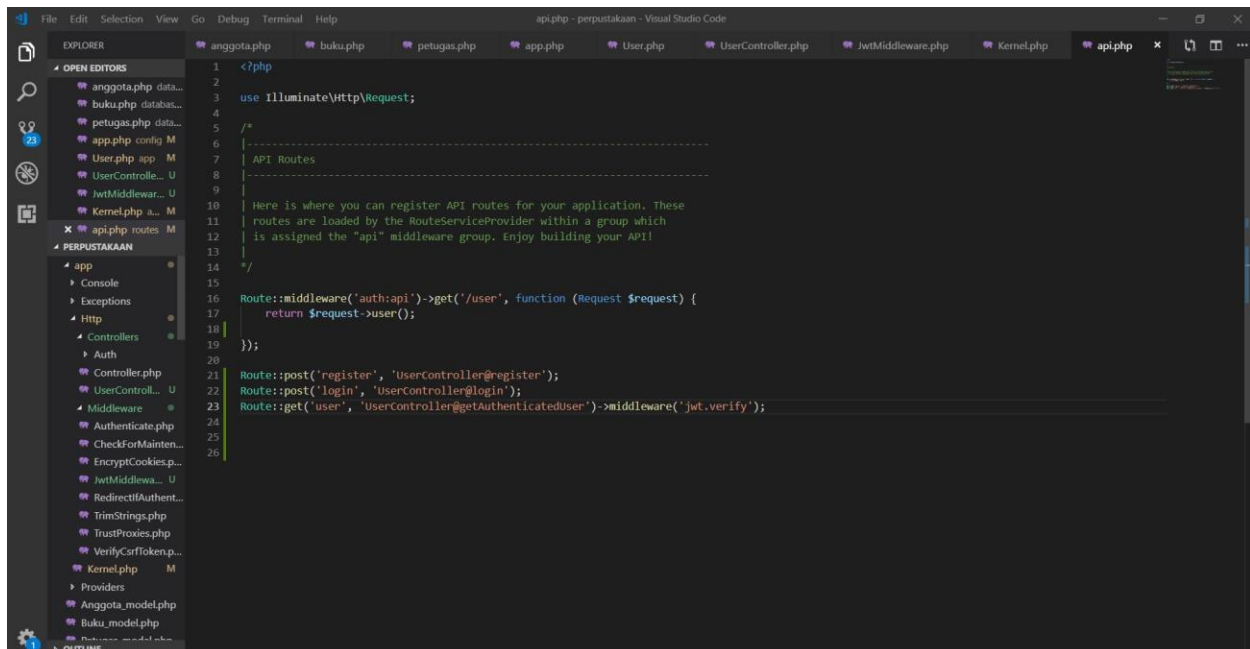


```
UserController.php - perpustakaan - Visual Studio Code

EXPLORER
  OPEN EDITORS
    anggota.php data...
    buku.php databas...
    petugas.php data...
    app.php config M
    User.php app M
    Controller.php ap...
    UserController... U
    JwtMiddleware... U
    Kernel.php a... M
  PERPUSTAKAAN
    app
    Console
    Exceptions
    Http
    Controllers
    Auth
    Controller.php
    UserController... U
    Middleware
    Authenticate.php
    CheckForMainten...
    EncryptCookies.p...
    JwtMiddleware... U
    RedirectIfAuthent...
    TrimStrings.php
    TrustProxies.php
    VerifyCsrfToken.p...
    Kernel.php M
    Providers
    Anggota_model.php
    Buku_model.php
    RouteServiceProvider
  OUTLINE

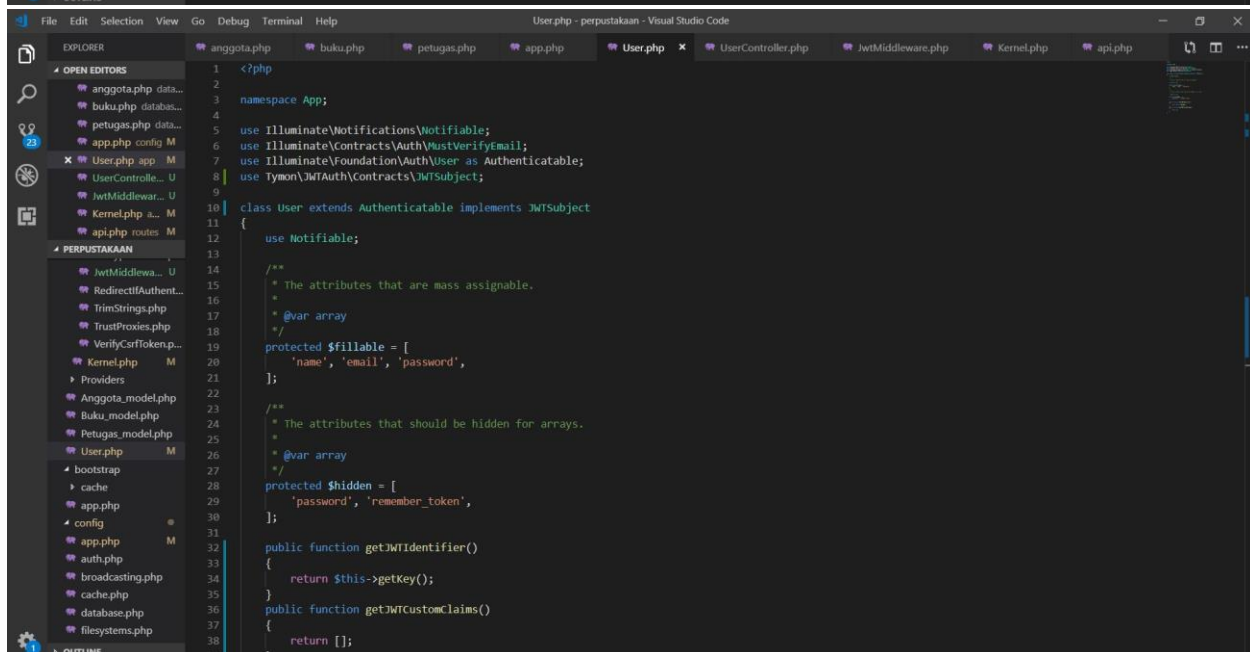
UserController.php - perpustakaan - Visual Studio Code

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 UserController extends Controller
13 {
14     public function login(Request $request)
15     {
16         $credentials = $request->only('email', 'password');
17
18         try {
19             if (! $token = JWTAuth::attempt($credentials)) {
20                 return response()->json(['error' => 'invalid_credentials'], 400);
21             }
22         } catch (JWTException $e) {
23             return response()->json(['error' => 'could_not_create_token'], 500);
24         }
25
26         return response()->json(compact('token'));
27     }
28
29     public function register(Request $request)
30     {
31         $validator = Validator::make($request->all(), [
32             'name' => 'required|string|max:255',
33             'email' => 'required|string|email|max:255|unique:users',
34             'password' => 'required|string|min:6|confirmed',
35         ]);
36
37         if ($validator->fails()) {
38             return response()->json($validator->errors()->toJson(), 400);
39         }
40
41         $user = User::create([
42             'name' => $request->get('name'),
43             'email' => $request->get('email'),
44             'password' => Hash::make($request->get('password')),
45         ]);
46
47         $token = JWTAuth::fromUser($user);
48
49         return response()->json(compact('user', 'token'), 201);
50     }
51
52     public function getAuthenticatedUser()
53     {
54         try {
55             if (! $user = JWTAuth::parseToken()->authenticate()) {
56                 return response()->json(['user_not_found'], 404);
57             }
58         } catch (Tymon\JWTAuth\Exceptions\TokenExpiredException $e) {
59             return response()->json(['token_expired'], $e->getStatusCode());
60         } catch (Tymon\JWTAuth\Exceptions\TokenInvalidException $e) {
61             return response()->json(['token_invalid'], $e->getStatusCode());
62         } catch (Tymon\JWTAuth\Exceptions\JWTException $e) {
63             return response()->json(['token_absent'], $e->getStatusCode());
64         }
65
66         return response()->json(compact('user'));
67     }
68 }
```



The screenshot shows the Visual Studio Code editor with the file `api.php` open. The Explorer sidebar on the left shows the project structure for `PERPUSTAKAAN`, including `app`, `bootstrap`, `cache`, `config`, `database.php`, `filesystems.php`, `kernel.php`, `providers`, `user.php`, `user_controller.php`, `jwtmiddleware.php`, `kernel.php`, and `api.php`. The main editor area displays the following PHP code:

```
1 <?php
2
3 use Illuminate\Http\Request;
4
5 /*
6 |-----|
7 | API Routes
8 |-----|
9
10 | Here is where you can register API routes for your application. These
11 | routes are loaded by the RouteServiceProvider within a group which
12 | is assigned the "api" middleware group. Enjoy building your API!
13 |
14 |*/
15
16 Route::middleware('auth:api')->get('/user', function (Request $request) {
17     return $request->user();
18 });
19
20
21 Route::post('register', 'UserController@register');
22 Route::post('login', 'UserController@login');
23 Route::get('user', 'UserController@getAuthenticatedUser')->middleware('jwt.verify');
```



The screenshot shows the Visual Studio Code editor with the file `User.php` open. The Explorer sidebar on the left shows the project structure for `PERPUSTAKAAN`, including `app`, `bootstrap`, `cache`, `config`, `database.php`, `filesystems.php`, `kernel.php`, `providers`, `user.php`, `user_controller.php`, `jwtmiddleware.php`, `kernel.php`, and `api.php`. The main editor area displays the following PHP code:

```
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      * @var array
17      */
18     protected $fillable = [
19         'name', 'email', 'password',
20     ];
21
22     /**
23      * The attributes that should be hidden for arrays.
24      * @var array
25      */
26     protected $hidden = [
27         'password', 'remember_token',
28     ];
29
30     public function getJWTIdentifier()
31     {
32         return $this->getKey();
33     }
34
35     public function getJWTCustomClaims()
36     {
37         return [];
38     }
39 }
```

The image displays two sequential screenshots of the Postman web interface. The top screenshot shows a 'POST' request to 'localhost:8000/api/register'. The 'Body' tab is selected, displaying form-data parameters: name (tegar), email (tegar@gmail.com), password (123456), and password_confirmation (123456). The response body is shown in JSON format, indicating a successful registration with a token and user details. The bottom screenshot shows a 'POST' request to 'localhost:8000/api/login'. The 'Body' tab is selected, displaying form-data parameters: email (tegar@gmail.com) and password (123456). The response body is shown in JSON format, returning a token. Both screenshots show the Postman sidebar with a collection of API endpoints and the top navigation bar with options like 'New', 'Import', 'Runner', and 'My Workspace'.