Web service documentation

We are using Laravel 9 for jwt authentication based login and signup API.

To protect user authentication API in Laravel we will use **tymondesigns/jwt-auth** a third-party jwt-auth library.

Install Laravel Application
Let us install a brand new laravel project to formulate the Laravel REST API project using JWT authentication.
Run following command to install a fresh Laravel project.

```
:- composer create-project laravel/laravel laravel-jwt-auth --prefer-dist
```

Database Connection
We have created a brand new Laravel application from scratch, now to store the user registration data, we have to create a database in MySQL.

Our env. File should look like this…

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=Laravel-and-react-case
DB_USERNAME=root
DB_PASSWORD=
```

Add User into MySQL Database

In this step, we will learn how to register a user table in MySQL database. Laravel fresh installation offers a default user table that we can register into the database using migration.

```
:-  php artisan migrate
```

The above command has created a users table inside the database. (we can edit the model, controller and user_table to add and remove login/signup fields for user).

Install & Configure JWT Authentication Package
Execute the following command to install tymondesigns/jwt-auth, It is a third-party JWT package and allows user authentication using JSON Web Token in Laravel & Lumen securely.

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

Above command installed the jwt-auth package in the vendor folder, now we have to go to **config/app.php** file and include the laravel service provider inside the providers array.
Also include the **JWTAuth** and **JWTFactory** facades inside the aliases array.

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

In the next step, we have to publish the package’s configuration, following command copy JWT Auth files from vendor folder to **config/jwt.php** file.

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

For handling the token encryption, generate a secret key by executing the following command.

```
:-  php artisan jwt:secret
```

We have successfully generated the JWT Secret key, and you can check this key inside the .env file.

Now you have to

- **Set -User Model.**
- **Configure Auth guard**
- **Build Authentication Controller**
  with command :- php artisan make:controller AuthController

Add Authentication Routes:- in routes/api.php

```
   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']);
```

We are using only two API in our project that are login and register and for logout we have created a seprate component in react frontend.

/**api/auth/register** :- this is a POST API which takes parameters = first_name, last_name, phone, email, password, password_confirmation.

We have added some validations in our controller for these parameters which are
 The first_name and last_name should have 2-20 characters
 The phone should have 5-12 digits
 The email should be unique not already registered
 The password should be minimum 6 characters
 The password_confirmation should be same as password

And in response we get message “User successfully registered” and the user data we has been stored in database

/**api/auth/login:-** this is a POST API which takes paremeters = email and password

And in response we get we get Acces_token, token_type, expires_in and the user data that is stored in database

/**api/auth/user-profile:-** this is a GET API which takes access token as parameter give the all user that is stored in database as response.

/**api/auth/refresh:-** this is a POST API which takes access_token as parameter and refreshes the access_token and gives same response as /user-profile API with new access_token

/**api/auth/logout:-** this is a POST API which takes access_token as parameter and logout the user (cleares data) and give message “user successfully signed out” as response.