step-1

1st create a Laravel project by using the commend in cmd.  
composer create-project laravel/laravel example-app-name

Step -2

Create DB.  
go to the project and open .env file and make changes for DB connection.

Step -3

Go to migration folder and make change to the user table as you want. Or you can make new migration table for your need.

Schema::create('users', function (Blueprint $table) {

            $table->id();

            $table->string('name');

            $table->string('email')->unique();

            $table->string('mobile')->nullable();;

            $table->string('address')->nullable();;

            $table->timestamp('email\_verified\_at')->nullable();

            $table->string('password');

            $table->rememberToken();

            $table->timestamps();

        });

Make changes in the model too.

protected $fillable = [

        'name',

        'email',

        'password',

        'mobile',

        'address',

    ];

Step-4

Install Laravel api by artisan command

php artisan install:api

it also installs Laravel sanctum with it.

Step-5

Run the command php artisan migrate to create all the table in the DB.

Step-6

Import use Laravel\Sanctum\HasApiTokens; in user model.  
add HasApiTokens in User class

class User extends Authenticatable

{

    use HasApiTokens,HasFactory, Notifiable;

step-7

make a controller to control auth.  
php artisan make:controller Api/AuthController –api

step-8

go to authcontroller and make function to register user.

/\*\*

     \* function register new user in database storage.

     \* @param use App\http\Requests\RegisterRequest;

     \* @param json response

     \*/

    public function register(Request $request){}

step-9

make a new request class  
php artisan make:request RegisterRequest

make authorize function true in registerRequest

public function authorize(): bool

    {

        return true;

    }

**i/ Add rules to function rule()**

public function rules(): array

    {

        return [

            'name' => 'required|min:5|max:150',

            'email' => 'required|email|unique:users',

            'mobile' => 'required|digits:11|unique:users',

            'address' => 'required',

            'password' => 'required|min:3',

        ];

    }

**ii/ also add message for the validation to** function messages()

/\*\*

     \* Get the validation messages that apply to the request.

     \*

     \* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>

     \*/

    public function messages(): array

    {

        return [

            'name.required' => 'please enter your name.',

            'name.min' => 'Name must be atleast 5 charecter',

            'name.max' => 'Name must not be more than 150',

            'email.required' => 'please enter your email.',

            'email.email' => 'Enter a valid Email',

            'email.unique' => 'sorry! this mail is already been used',

            'mobile.required' => 'please enter your mobile number',

            'mobile.digits' => 'mobile number must be 11 digits',

            'mobile.unique' => 'The mobile number has already been used.',

            'address.required' => 'please enter your address',

            'password.required' => 'please enter Password',

            'password.min' => 'Password must be atleast 3 charecter.'

        ];

    }

**Step-10**

Go to authController and import the RegisterRequest  
use App\http\Requests\RegisterRequest;

Using RegisterRequest $request get data in function register

public function register(RegisterRequest $request)

    {

//chack using dd

dd($request->all());

}

**Step-11**

Use try catch and wright eloquent to insert data into database.

Import User model

use App\Models\User;

Also Import Hash

use Illuminate\Support\Facades\Hash;

and using hash function make password encrypted

 try{

         $user= User::create([

            'name' => $request->name,

            'email' => $request->email,

            'mobile' => $request->mobile ,

            'address' => $request->address ,

            'password' => Hash::make($request->password)

         ]);

      }

      catch(Exception $e){

      }

Step-12

Go to api route, import authController

use App\Http\Controllers\Api\AuthController;

make route group for authcontroller

route::controller(AuthController::class)->group(function(){

});

Add route to the group for register user

route::controller(AuthController::class)->group(function(){

    Route::post('register','register');

});

Test it using postman

**Step-13**

Make helper class using command

php artisan make:class Helper/ResponseHelper

mke two class in helper class one for success and one for error/fail

/\*\*

     \* common function to show success message tp json response

     \*  @param string $status

     \*  @param string $message

     \*  @param array $data

     \*  @param integer $statusCode

     \*  @return response

     \*\*/

    public static function success($status='success', $message=null, $data=[], $statusCode=200){

        return response()->json([

            'status'=> $status,

            'message'=>$message,

            'data'=>$data,

        ],$statusCode );

    }

    /\*\*

     \* common function to show error message tp json response

     \*  @param string $status

     \*  @param string $message

     \*  @param integer $statusCode

     \*  @return response

     \*\*/

    public static function error($status='error', $message=null, $statusCode=400){

        return response()->json([

            'status'=> $status,

            'message'=>$message,

        ],$statusCode );

    }

Import ResponseHelper to the authcontroller

use App\Helper\ResponseHelper;

#now if data stored successfully to the database than return response from try{} using responsehelper.

if($user){

            return ResponseHelper::success(message: 'User has been registerd successfully!', data: $user, statusCode:201);

         }

         return ResponseHelper::error(message: 'Unable to register User! try again.', statusCode:400);

#also return error response from the catch{} using responsehelper

catch(Exception $e){

        \log::error('unable to register user: '. $e->getMessage(). ' - line number '.$e->getLine() );

        return ResponseHelper::error(message: 'Unable to register User! try again.'.$e->getMessage(), statusCode:500);

      }

Then try to register using postman

In postman hit the link

<http://localhost:8000/api/register>

pass value in header

key “Accept”

value “application/json”

also pass rest of the value using key through body.

'name' => 'sunny'

            'email' => 'sunny@gmail.com'

            'mobile' => '01684091969'

            'address' => 'dhaka'

            'password' => 'sfhfkjhkj'

**Login**

Step-1

Make a login request

php artisan make:request LoginRequest

make authorize true

 public function authorize(): bool

    {

        return true;

    }

#make to two function for rules and message

/\*\*

     \* Get the validation rules that apply to the request.

     \*

     \* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>

     \*/

    public function rules(): array

    {

        return [

           'email' => 'required|email',

           'password'=> 'required'

        ];

    }

    /\*\*

     \* Get the validation error message that apply to the request.

     \*

     \* @return array<string, \Illuminate\Contracts\Validation\ValidationRule|array<mixed>|string>

     \*/

    public function message(): array

    {

        return [

           'email.required' => 'please Enter registered email',

           'email.email' => 'Enter your valid email',

           'password.required'=> 'please Enter your password'

        ];

    }

#import it to the authcontroller

use App\http\Requests\LoginRequest;

Also import auth   
chack the auth as user is valid or not using mail and password.

If user is valid create token else show error message.

/\*\*

     \* function login user

     \* @param use App\http\Requests\LoginRequest;

     \*/

    public function login(LoginRequest $request)

    {

        //dd($request->all());

        try{

            if(!Auth::attempt(['email'=>$request->email, 'password'=>$request->password])){

                return ResponseHelper::error(message: 'Unable to Login! Incurrect infomation', statusCode:400);

            };

            $user = Auth::user();

            $token = $user->createToken('My Api Token')->plainTextToken;

            $authUser = [

                'user'=> $user,

                'token' => $token

            ];

            return ResponseHelper::success(message: 'User loged in successfully!', data: $authUser, statusCode:200);

        }

        catch(Exception $e){

            \log::error('unable to Login: '. $e->getMessage(). ' - line number '.$e->getLine() );

            return ResponseHelper::error(message: 'Unable to Login! try again.'.$e->getMessage(), statusCode:500);

        }

    }

Step-2

Make api route

Route::post('login','login');

**Show data**

Make another function in authController to show the data using token.

/\*\*

     \* function auth user data, get user profile

     \* @param NA;

     \* @return JSON response

     \*/

    public function userProfile()

    {

        try{

           $user=auth::user();

           if($user){

            return ResponseHelper::success(message: 'User Data Fetch Successfully!', data: $user, statusCode:200);

         }

         return ResponseHelper::error(message: 'Fetch User Data Faild! try again.', statusCode:400);

        }

        catch(Exception $e){

            \log::error('unable to fetch user profile data: '. $e->getMessage(). ' - line number '.$e->getLine() );

            return ResponseHelper::error(message: 'unable to fetch user profile data! try again.'.$e->getMessage(), statusCode:500);

        }

    }

Make api for that and add middleware to the route for authenticate.

Route::get('profile','userProfile')->middleware('auth:sanctum');

Go to postman use the link

<http://localhost:8000/api/profile>

send authorization

token under the key bearer token.

**Log Out**

Make another function in authcontroller to logout the user and destroy the token.

 /\*\*

     \* function logout

     \* @param NA;

     \* @return JSON response

     \*/

    public function logOut()

    {

        try{

           $user=auth::user();

           if($user){

            $user->currentAccessToken()->delete();

            return ResponseHelper::success(message: 'Logout Successfully!', data: $user, statusCode:200);

         }

         return ResponseHelper::error(message: 'logout Faild! due to wrong token.', statusCode:400);

        }

        catch(Exception $e){

            \log::error('unable to logout: '. $e->getMessage(). ' - line number '.$e->getLine() );

            return ResponseHelper::error(message: 'unable to logout! try again.'.$e->getMessage(), statusCode:500);

        }

    }

#create route for the function   
Route::get('logout','logOut')->middleware('auth:sanctum');

Check it through postman by passing the token