Sigin And Sigup

Sure! Let's summarize the step-by-step process to set up registration and login functionality in Laravel using Laravel Sanctum for API authentication, including handling potential errors.

**Step-by-Step Guide**

**Step 1: Install Laravel**

Create a new Laravel project if you haven't already:

sh

Copy code

composer create-project --prefer-dist laravel/laravel your\_project\_name

**Step 2: Set Up Database Configuration**

Configure your database in the .env file:

env

Copy code

DB\_CONNECTION=mysql

DB\_HOST=127.0.0.1

DB\_PORT=3306

DB\_DATABASE=your\_database\_name

DB\_USERNAME=your\_username

DB\_PASSWORD=your\_password

**Step 3: Run Migrations**

Run the built-in migrations to create the necessary database tables:

sh

Copy code

php artisan migrate

**Step 4: Install Laravel Sanctum**

Install Sanctum for API authentication:

sh

Copy code

composer require laravel/sanctum

**Step 5: Publish Sanctum Configuration**

Publish the Sanctum configuration file:

sh

Copy code

php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"

**Step 6: Run Sanctum Migrations**

Run the Sanctum migrations:

sh

Copy code

php artisan migrate

**Step 7: Configure Sanctum Middleware**

1. **Open app/Http/Kernel.php**.
2. **Add Sanctum Middleware**:

php

Copy code

namespace App\Http;

use Illuminate\Foundation\Http\Kernel as HttpKernel;

use Laravel\Sanctum\Http\Middleware\EnsureFrontendRequestsAreStateful;

class Kernel extends HttpKernel

{

protected $middleware = [

// Other middleware

];

protected $middlewareGroups = [

'web' => [

\App\Http\Middleware\EncryptCookies::class,

\Illuminate\Cookie\Middleware\AddQueuedCookiesToResponse::class,

\Illuminate\Session\Middleware\StartSession::class,

\Illuminate\View\Middleware\ShareErrorsFromSession::class,

\App\Http\Middleware\VerifyCsrfToken::class,

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

'api' => [

EnsureFrontendRequestsAreStateful::class,

'throttle:api',

\Illuminate\Routing\Middleware\SubstituteBindings::class,

],

];

}

**Step 8: Configure User Model**

Add the HasApiTokens trait to your User model:

php

Copy code

namespace App\Models;

use Laravel\Sanctum\HasApiTokens;

use Illuminate\Foundation\Auth\User as Authenticatable;

class User extends Authenticatable

{

use HasApiTokens, Notifiable;

// other model properties and methods

}

**Step 9: Create Auth Controller**

Create a controller to handle registration and login:

sh

Copy code

php artisan make:controller Api/AuthController

**Step 10: Implement Register and Login Methods**

In the AuthController, add methods for registration and login:

php

Copy code

namespace App\Http\Controllers\Api;

use App\Models\User;

use Illuminate\Http\Request;

use App\Http\Controllers\Controller;

use Illuminate\Support\Facades\Hash;

use Illuminate\Support\Facades\Validator;

class AuthController extends Controller

{

public function register(Request $request)

{

$validator = Validator::make($request->all(), [

'name' => 'required|string|max:255',

'email' => 'required|string|email|max:255|unique:users',

'password' => 'required|string|min:8',

]);

if ($validator->fails()) {

return response()->json($validator->errors(), 422);

}

$user = User::create([

'name' => $request->name,

'email' => $request->email,

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

]);

$token = $user->createToken('auth\_token')->plainTextToken;

return response()->json(['token' => $token], 200);

}

public function login(Request $request)

{

$credentials = $request->only('email', 'password');

if (!auth()->attempt($credentials)) {

return response()->json(['error' => 'Unauthorised'], 401);

}

$user = User::where('email', $request->email)->firstOrFail();

$token = $user->createToken('auth\_token')->plainTextToken;

return response()->json(['token' => $token], 200);

}

}

**Step 11: Define API Routes**

In the routes/api.php file, add routes for registration and login:

php

Copy code

use App\Http\Controllers\Api\AuthController;

Route::post('register', [AuthController::class, 'register']);

Route::post('login', [AuthController::class, 'login']);

**Step 12: Clear Configuration and Route Cache**

Run these commands to ensure that your changes are applied:

sh

Copy code

php artisan config:cache

php artisan config:clear

php artisan route:cache

php artisan route:clear

**Step 13: Test the API**

Run the Laravel development server:

sh

Copy code

php artisan serve

Use a tool like Postman to test your API endpoints:

* **Register**: POST /api/register
  + Headers: Content-Type: application/json
  + Body:

json

Copy code

{

"name": "John Doe",

"email": "john@example.com",

"password": "password"

}

* **Login**: POST /api/login
  + Headers: Content-Type: application/json
  + Body:

json

Copy code

{

"email": "john@example.com",

"password": "password"

}

**Error Handling**

* **Validation Errors**: Handled by returning a JSON response with a 422 status code.
* **Unauthorized Errors**: Handled by returning a JSON response with a 401 status code if authentication fails.
* **General Errors**: Ensure to return appropriate status codes and error messages to help with debugging.

**Summary**

1. **Install Laravel**: composer create-project --prefer-dist laravel/laravel your\_project\_name
2. **Configure Database**: Update .env with your database settings.
3. **Run Migrations**: php artisan migrate
4. **Install Sanctum**: composer require laravel/sanctum
5. **Publish Sanctum Config**: php artisan vendor:publish --provider="Laravel\Sanctum\SanctumServiceProvider"
6. **Run Sanctum Migrations**: php artisan migrate
7. **Configure Middleware**: Add EnsureFrontendRequestsAreStateful::class in app/Http/Kernel.php
8. **Update User Model**: Add HasApiTokens trait.
9. **Create Auth Controller**: php artisan make:controller Api/AuthController
10. **Implement Methods**: Add register and login methods.
11. **Define Routes**: Add routes in routes/api.php.
12. **Clear Cache**: php artisan config:cache and php artisan route:cache
13. **Test API**: Use Postman or another tool to test endpoints.

This setup will provide a complete registration and login API using Laravel Sanctum, with proper error handling included.

4o

swagger