Got it! Let's take it step by step, no rush. Let me know which part you'd like to focus on next or if you want to review any of the previous steps in more detail. I'm here to assist whenever you're ready!

Perfect! Let's start by setting up the **React 19 frontend** for the **login** functionality, step by step. We’ll first create the React application, then integrate the login flow with **Sanctum** authentication.

**Step 1: Set up the React 19 Application**

1. **Create the React app:**

Run the following command to create a new React 19 application:

npx create-react-app react-frontend

1. **Navigate to the project directory:**
2. cd react-frontend
3. **Install Axios for API requests:**

Axios will help in making HTTP requests to your Laravel backend. You can install it by running:

npm install axios

1. **Install React Router DOM** (for handling routing in React):
2. npm install react-router-dom

**Step 2: Configure Axios**

1. **Create an Axios instance** in the src directory to handle requests to your Laravel backend.

Create a new file src/axios.js:

import axios from 'axios';

const axiosInstance = axios.create({

baseURL: 'http://localhost', // Update with your backend API URL

headers: {

'Content-Type': 'application/json',

},

});

export default axiosInstance;

**Step 3: Create the Login Component**

1. **Create a Login component** where users can input their credentials.

Inside src folder, create a new folder called components and a file Login.js inside it.

import React, { useState } from 'react';

import axios from '../axios'; // Import the axios instance

import { useHistory } from 'react-router-dom';

const Login = () => {

const [email, setEmail] = useState('');

const [password, setPassword] = useState('');

const [error, setError] = useState('');

const history = useHistory();

const handleSubmit = async (e) => {

e.preventDefault();

try {

// Make the login request to the Laravel backend

const response = await axios.post('/api/login', {

email,

password,

});

// If login is successful, check role and redirect

const user = response.data.user;

if (user.role === 'superadmin') {

history.push('/admin');

} else {

setError('You are not a superadmin!');

}

} catch (err) {

setError('Invalid credentials!');

}

};

return (

<div>

<h2>Login</h2>

<form onSubmit={handleSubmit}>

<div>

<label>Email</label>

<input

type="email"

value={email}

onChange={(e) => setEmail(e.target.value)}

required

/>

</div>

<div>

<label>Password</label>

<input

type="password"

value={password}

onChange={(e) => setPassword(e.target.value)}

required

/>

</div>

{error && <p style={{ color: 'red' }}>{error}</p>}

<button type="submit">Login</button>

</form>

</div>

);

};

export default Login;

**Step 4: Set Up Routing**

1. **Add routing** to the application by modifying src/App.js to handle the login route.
2. import React from 'react';
3. import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';
4. import Login from './components/Login';
5. const App = () => {
6. return (
7. <Router>
8. <Switch>
9. <Route exact path="/" component={Login} />
10. {/\* Add routes for the admin section once created \*/}
11. </Switch>
12. </Router>
13. );
14. };
15. export default App;

**Step 5: Create Laravel Routes for Login (Backend)**

Now, we need to create an API route in Laravel to handle the login request and return the user information along with the role.

In your routes/api.php file, add a login route like this:

use Illuminate\Support\Facades\Route;

use App\Http\Controllers\AuthController;

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

**Step 6: Create the AuthController for Login**

Create a new controller AuthController to handle the login process. The controller will validate the user, issue a token, and return the user role.

namespace App\Http\Controllers;

use Illuminate\Http\Request;

use App\Models\User;

use Illuminate\Support\Facades\Auth;

use Illuminate\Support\Facades\Hash;

use Laravel\Sanctum\HasApiTokens;

class AuthController extends Controller

{

public function login(Request $request)

{

$credentials = $request->validate([

'email' => 'required|email',

'password' => 'required',

]);

if (Auth::attempt($credentials)) {

$user = Auth::user();

// Create a token for the user

$token = $user->createToken('React App')->plainTextToken;

return response()->json([

'token' => $token,

'user' => $user,

]);

}

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

}

}

**Step 7: Update the User Model to Use Sanctum**

In your User.php model, ensure you are using the HasApiTokens trait from Sanctum:

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Foundation\Auth\User as Authenticatable;

use Illuminate\Notifications\Notifiable;

use Laravel\Sanctum\HasApiTokens;

class User extends Authenticatable

{

use HasApiTokens, HasFactory, Notifiable;

// Other code

}

**Step 8: Testing the Login Flow**

1. **Run the Laravel server**:
2. php artisan serve
3. **Run the React app**:
4. npm start
5. Test the **login flow** by visiting the React frontend and logging in with the superadmin credentials. If successful, it will redirect to the /admin page.

Let me know if you need assistance with any of these steps! We will continue from here once the login functionality is set up.