How to Create Dynamic Breadcrumb Navigation in React.js with Laravel Backend

In modern web applications, breadcrumb navigation helps users understand their location within a site and easily navigate between sections. This tutorial will guide you through building dynamic breadcrumb navigation using React.js for the frontend and Laravel for the backend API.

# Step 1: Setting Up Laravel Backend

Before we dive into the React part, you need to have a working Laravel backend that serves the necessary routes and handles page requests.

## 1.1 Create Laravel API for Routes

In your Laravel backend, create an API endpoint to provide the breadcrumb data for each route. This could look something like:

// routes/api.php  
Route::get('/breadcrumbs/{route}', [BreadcrumbController::class, 'getBreadcrumb']);

## 1.2 Create the Breadcrumb Controller

In the BreadcrumbController, create a method to return breadcrumb data based on the route passed:

// app/Http/Controllers/BreadcrumbController.php  
namespace App\Http\Controllers;  
  
use Illuminate\Http\Request;  
  
class BreadcrumbController extends Controller  
{  
 public function getBreadcrumb($route)  
 {  
 // For simplicity, you can hardcode route mappings or retrieve them from a database  
 $breadcrumbs = [  
 'home' => ['Home'],  
 'dashboard' => ['Home', 'Dashboard'],  
 'profile' => ['Home', 'Profile'],  
 ];  
  
 return response()->json($breadcrumbs[$route] ?? ['Home']);  
 }  
}

# Step 2: Setting Up React.js Frontend

## 2.1 Install Axios

To fetch breadcrumb data from the Laravel backend, install Axios in your React project:

npm install axios

## 2.2 Create the Breadcrumb Component

In your React project, create a Breadcrumb component that fetches and displays the breadcrumb trail dynamically based on the current route:

// src/components/Breadcrumb.js  
import React, { useEffect, useState } from 'react';  
import axios from 'axios';  
  
const Breadcrumb = ({ route }) => {  
 const [breadcrumbs, setBreadcrumbs] = useState([]);  
  
 useEffect(() => {  
 axios.get(`/api/breadcrumbs/${route}`)  
 .then(response => setBreadcrumbs(response.data))  
 .catch(error => console.error('Error fetching breadcrumbs:', error));  
 }, [route]);  
  
 return (  
 <nav aria-label="breadcrumb">  
 <ol className="breadcrumb">  
 {breadcrumbs.map((crumb, index) => (  
 <li key={index} className={`breadcrumb-item ${index === breadcrumbs.length - 1 ? 'active' : ''}`}>  
 {index !== breadcrumbs.length - 1 ? <a href={`/${crumb.toLowerCase()}`}>{crumb}</a> : crumb}  
 </li>  
 ))}  
 </ol>  
 </nav>  
 );  
};  
  
export default Breadcrumb;

## 2.3 Use Breadcrumb Component in Header

Now, include the Breadcrumb component in your header or wherever you want to display the navigation:

// src/components/Header.js  
import React from 'react';  
import Breadcrumb from './Breadcrumb';  
  
const Header = ({ currentRoute }) => {  
 return (  
 <header>  
 <Breadcrumb route={currentRoute} />  
 {/\* Other header content \*/}  
 </header>  
 );  
};  
  
export default Header;

## 2.4 Update Routes to Pass Correct Route to Breadcrumb

Make sure you pass the correct route to the Breadcrumb component. This could be based on your routing setup, for example using React Router:

// src/App.js  
import React from 'react';  
import { BrowserRouter as Router, Route, Switch } from 'react-router-dom';  
import Header from './components/Header';  
  
const App = () => {  
 return (  
 <Router>  
 <Header currentRoute={window.location.pathname.substring(1)} />  
 <Switch>  
 <Route path="/dashboard" component={Dashboard} />  
 <Route path="/profile" component={Profile} />  
 {/\* Other routes \*/}  
 </Switch>  
 </Router>  
 );  
};  
  
export default App;

# Conclusion

By following these steps, you can create a dynamic breadcrumb navigation system in React.js, powered by a Laravel backend. This setup allows for flexibility in adding or updating routes while keeping the breadcrumb data in sync with your application's navigation flow.