Project Setup Documentation

# 1. Backend: Laravel API Setup

Step 1: Install Laravel  
First, create a Laravel project if it’s not already set up.  
  
```bash  
composer create-project --prefer-dist laravel/laravel backend  
cd backend  
```

Step 2: Configure Environment Variables  
Copy the `.env.example` file to `.env`:  
  
```bash  
cp .env.example .env  
```  
  
Update the `.env` file with your database configuration:  
  
```env  
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  
If you have migrations set up for categories and products, run the following command to migrate the database:  
  
```bash  
php artisan migrate  
```

Step 4: Create API Routes  
In `routes/api.php`, define the routes for CRUD operations on categories and products:  
  
```php  
use App\Http\Controllers\CategoryController;  
  
Route::resource('categories', CategoryController::class);  
Route::resource('products', ProductController::class); // Add product routes if required  
```

Step 5: Create Category and Product Controllers  
Create controllers for Category and Product to handle CRUD operations. Example for Category:  
  
```bash  
php artisan make:controller CategoryController --resource  
```  
  
Then, add the logic for each CRUD method in `CategoryController.php`:  
  
```php  
namespace App\Http\Controllers;  
  
use App\Models\Category;  
use Illuminate\Http\Request;  
  
class CategoryController extends Controller  
{  
 public function index()  
 {  
 return Category::all();  
 }  
  
 public function store(Request $request)  
 {  
 $category = Category::create($request->all());  
 return response()->json($category, 201);  
 }  
  
 public function update(Request $request, $id)  
 {  
 $category = Category::find($id);  
 $category->update($request->all());  
 return response()->json($category);  
 }  
  
 public function destroy($id)  
 {  
 Category::destroy($id);  
 return response()->json(null, 204);  
 }  
}  
```

Step 6: Start Laravel Server  
Run the Laravel development server:  
  
```bash  
php artisan serve  
```  
  
The backend should now be running at `http://localhost:8000`.

Step 7: Test Laravel API  
Make sure to test the API routes using Postman or a browser.  
  
For example, check if the categories endpoint is working:  
  
```  
GET http://localhost:8000/api/categories  
```

# 2. Frontend: Next.js Setup

Step 1: Install Next.js  
Create a Next.js project using the following commands:  
  
```bash  
npx create-next-app@latest frontend  
cd frontend  
yarn install  
```

Step 2: Install Axios and React Icons  
To make API calls, install `axios` and `react-icons` for UI components:  
  
```bash  
yarn add axios react-icons  
```

Step 3: Configure Axios for API Calls  
In `services/categoryService.js`, configure Axios to point to your Laravel backend:  
  
```js  
import axios from 'axios';  
  
const api = axios.create({  
 baseURL: 'http://localhost:8000/api', // Ensure this is the correct URL  
});  
  
export const getCategories = async () => {  
 return api.get('/categories');  
};  
  
export const createCategory = async (data) => {  
 return api.post('/categories', data);  
};  
  
export const updateCategory = async (id, data) => {  
 return api.put(`/categories/${id}`, data);  
};  
  
export const deleteCategory = async (id) => {  
 return api.delete(`/categories/${id}`);  
};  
```

Step 4: Create Category Form Component  
In `components/CategoryForm.jsx`, create a reusable form component for adding or editing categories:  
  
```js  
import { useState, useEffect } from 'react';  
  
const CategoryForm = ({ onSubmit, initialData }) => {  
 const [name, setName] = useState('');  
 const [description, setDescription] = useState('');  
  
 useEffect(() => {  
 if (initialData) {  
 setName(initialData.name);  
 setDescription(initialData.description);  
 }  
 }, [initialData]);  
  
 const handleSubmit = (e) => {  
 e.preventDefault();  
 onSubmit({ name, description });  
 };  
  
 return (  
 <form onSubmit={handleSubmit}>  
 <div>  
 <label>Name</label>  
 <input  
 type="text"  
 value={name}  
 onChange={(e) => setName(e.target.value)}  
 required  
 />  
 </div>  
 <div>  
 <label>Description</label>  
 <input  
 type="text"  
 value={description}  
 onChange={(e) => setDescription(e.target.value)}  
 required  
 />  
 </div>  
 <button type="submit">Submit</button>  
 </form>  
 );  
};  
  
export default CategoryForm;  
```

Step 5: Create Category List Page  
In `app/settings/category/page.jsx`, create a page to list all categories:  
  
```js  
import { useEffect, useState } from 'react';  
import { getCategories } from '../../../services/categoryService';  
import Link from 'next/link';  
  
const CategoryListPage = () => {  
 const [categories, setCategories] = useState([]);  
  
 useEffect(() => {  
 const fetchCategories = async () => {  
 const response = await getCategories();  
 setCategories(response.data);  
 };  
  
 fetchCategories();  
 }, []);  
  
 return (  
 <div>  
 <h1>Category List</h1>  
 <Link href="/settings/category/create">Create New Category</Link>  
 <ul>  
 {categories.map((category) => (  
 <li key={category.id}>  
 {category.name} - {category.description}  
 <Link href={`/settings/category/update/${category.id}`}>Edit</Link>  
 </li>  
 ))}  
 </ul>  
 </div>  
 );  
};  
  
export default CategoryListPage;  
```

Step 6: Create Category Create Page  
In `app/settings/category/create/page.jsx`, create a page for creating a category:  
  
```js  
import CategoryForm from '../../components/CategoryForm';  
import { createCategory } from '../../../services/categoryService';  
import { useRouter } from 'next/router';  
  
const CreateCategoryPage = () => {  
 const router = useRouter();  
  
 const handleSubmit = async (data) => {  
 await createCategory(data);  
 router.push('/settings/category');  
 };  
  
 return (  
 <div>  
 <h1>Create Category</h1>  
 <CategoryForm onSubmit={handleSubmit} />  
 </div>  
 );  
};  
  
export default CreateCategoryPage;  
```

Step 7: Create Category Update Page  
In `app/settings/category/update/page.jsx`, create a page to update an existing category:  
  
```js  
import CategoryForm from '../../components/CategoryForm';  
import { useRouter } from 'next/router';  
import { useEffect, useState } from 'react';  
import { getCategories, updateCategory } from '../../../services/categoryService';  
  
const UpdateCategoryPage = () => {  
 const router = useRouter();  
 const { id } = router.query;  
 const [category, setCategory] = useState(null);  
  
 useEffect(() => {  
 if (id) {  
 const fetchCategory = async () => {  
 const response = await getCategories();  
 const foundCategory = response.data.find((cat) => cat.id === id);  
 setCategory(foundCategory);  
 };  
  
 fetchCategory();  
 }  
 }, [id]);  
  
 const handleSubmit = async (data) => {  
 await updateCategory(id, data);  
 router.push('/settings/category');  
 };  
  
 if (!category) return <div>Loading...</div>;  
  
 return (  
 <div>  
 <h1>Update Category</h1>  
 <CategoryForm onSubmit={handleSubmit} initialData={category} />  
 </div>  
 );  
};  
  
export default UpdateCategoryPage;  
```

Step 8: Add Sidebar and Menu  
Create a basic sidebar in `components/Sidebar.jsx` and use it in your main layout:  
  
```js  
import Link from 'next/link';  
import { menuItems } from '../constants/menu';  
  
const Sidebar = () => {  
 return (  
 <div>  
 <h2>Sidebar</h2>  
 <ul>  
 {menuItems.map((item) => (  
 <li key={item.name}>  
 <Link href={item.link}>{item.name}</Link>  
 </li>  
 ))}  
 </ul>  
 </div>  
 );  
};  
  
export default Sidebar;  
```  
  
Wrap your layout with the sidebar:  
  
```js  
const Layout = ({ children }) => {  
 return (  
 <div style={{ display: 'flex' }}>  
 <Sidebar />  
 <main style={{ marginLeft: '200px', padding: '20px' }}>{children}</main>  
 </div>  
 );  
};  
  
export default Layout;  
```

# 3. Running the Project

Step 1: Run the Backend (Laravel)  
1. \*\*Start the Laravel server\*\*:  
  
```bash  
cd backend  
php artisan serve  
```  
  
Your backend should be available at `http://localhost:8000`.

Step 2: Run the Frontend (Next.js)  
1. \*\*Start the Next.js server\*\*:  
  
```bash  
cd frontend  
yarn dev  
```  
  
Your frontend should be available at `http://localhost:3000`.

Step 3: Test the Full Application  
1. Ensure the \*\*API\*\* is working by accessing `http://localhost:8000/api/categories` and testing CRUD operations.  
2. Test the \*\*frontend\*\* by visiting `http://localhost:3000/settings/category`.