Integrating React with Django – Complete Guide

# Objective

This document provides a step-by-step guide on how to integrate React with Django. It covers the   
implementation of a sidebar menu with two options: category upload and fish upload. The document includes both frontend   
(React) and backend (Django) code for each feature, along with an explanation of how React and Django communicate.

# React-Django Integration Overview

In this integration, React serves as the frontend for the user interface, while Django serves as the   
backend for handling data, processing requests, and providing the necessary API endpoints. This document focuses on   
creating two React components: one for uploading categories and another for uploading fish. Both components will interact   
with the Django backend to send and receive data through API calls.

# Backend Setup (Django)

On the Django side, we need to implement the necessary API views for handling the category and fish   
uploads. These views will receive data from the React components and store it in the database. Additionally, Django must   
serve static and media files (such as images) for the React frontend to access.  
  
First, let's ensure that Django is configured to handle media files by adding the following settings in `settings.py`:

MEDIA\_URL = '/media/'  
MEDIA\_ROOT = os.path.join(BASE\_DIR, 'media')

In your `urls.py`, include the media URL path to serve images:

from django.conf import settings  
from django.conf.urls.static import static  
  
urlpatterns = [  
 # Your API endpoints and other URLs  
] + static(settings.MEDIA\_URL, document\_root=settings.MEDIA\_ROOT)

Now, let's implement the Django views for handling category and fish uploads.

# Backend Code for Category and Fish Upload

In Django, we need to create models to store category and fish data, as well as serializers to   
convert data between Python objects and JSON. Finally, we'll create views that handle POST requests to add new categories   
and fish.

First, create the `models.py` for category and fish:

# models.py  
from django.db import models  
  
class Category(models.Model):  
 name = models.CharField(max\_length=255)  
 description = models.TextField()  
  
 def \_\_str\_\_(self):  
 return self.name  
  
class Fish(models.Model):  
 name = models.CharField(max\_length=255)  
 description = models.TextField()  
 price = models.DecimalField(max\_digits=10, decimal\_places=2)  
 category = models.ForeignKey(Category, on\_delete=models.CASCADE)  
 image = models.ImageField(upload\_to='fish\_images/')  
  
 def \_\_str\_\_(self):  
 return self.name

Next, create the `serializers.py` to serialize the category and fish models:

# serializers.py  
from rest\_framework import serializers  
from .models import Category, Fish  
  
class CategorySerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Category  
 fields = '\_\_all\_\_'  
  
class FishSerializer(serializers.ModelSerializer):  
 class Meta:  
 model = Fish  
 fields = '\_\_all\_\_'

Then, create the views in `views.py` to handle category and fish uploads:

# views.py  
from rest\_framework import viewsets  
from .models import Category, Fish  
from .serializers import CategorySerializer, FishSerializer  
from rest\_framework.decorators import action  
from rest\_framework.response import Response  
  
class CategoryViewSet(viewsets.ModelViewSet):  
 queryset = Category.objects.all()  
 serializer\_class = CategorySerializer  
  
class FishViewSet(viewsets.ModelViewSet):  
 queryset = Fish.objects.all()  
 serializer\_class = FishSerializer  
  
 @action(detail=False, methods=['post'])  
 def upload\_fish(self, request):  
 # Your custom fish upload logic, including saving the image  
 pass

Finally, add URLs to connect the API views in `urls.py`:

# urls.py  
from django.urls import path, include  
from rest\_framework.routers import DefaultRouter  
from .views import CategoryViewSet, FishViewSet  
  
router = DefaultRouter()  
router.register(r'categories', CategoryViewSet)  
router.register(r'fish', FishViewSet)  
  
urlpatterns = [  
 path('api/', include(router.urls)),  
]

# Frontend Code (React Components)

Now, let's focus on the frontend. We will create two React components: one for uploading categories   
and another for uploading fish. Both components will interact with the Django API to send data to the backend.  
  
First, we set up a basic React app with Axios to make HTTP requests. Here’s how to create a sidebar menu with options   
to upload categories and fish.

# Sidebar Menu

Here is an example of how to create a simple sidebar with two options: Category Upload and Fish Upload.

import React from 'react';  
import { Link } from 'react-router-dom';  
  
function Sidebar() {  
 return (  
 <div>  
 <nav>  
 <ul>  
 <li><Link to="/category-upload">Upload Category</Link></li>  
 <li><Link to="/fish-upload">Upload Fish</Link></li>  
 </ul>  
 </nav>  
 </div>  
 );  
}  
  
export default Sidebar;

In the above code, we create a sidebar with two links that lead to the respective upload pages   
for category and fish.

# CategoryUpload Component

The `CategoryUpload` component allows the user to upload a new category with a name and description.   
Here’s the code for the component.

import React, { useState } from 'react';  
import axios from 'axios';  
  
function CategoryUpload() {  
 const [name, setName] = useState('');  
 const [description, setDescription] = useState('');  
 const [successMessage, setSuccessMessage] = useState('');  
 const [errorMessage, setErrorMessage] = useState('');  
  
 const handleSubmit = async (e) => {  
 e.preventDefault();  
  
 try {  
 await axios.post(  
 'http://localhost:9000/admin-fish/api/categories/',  
 { name, description },  
 { headers: { 'Content-Type': 'application/json' } }  
 );  
 setSuccessMessage('Category successfully created!');  
 setErrorMessage('');  
 setName('');  
 setDescription('');  
 } catch (error) {  
 console.error('Error uploading category:', error);  
 setErrorMessage('Error uploading category');  
 setSuccessMessage('');  
 }  
 };  
  
 return (  
 <div>  
 <h2>New Category</h2>  
 <form onSubmit={handleSubmit}>  
 <div>  
 <label>Category Name</label>  
 <input type="text" value={name} onChange={(e) => setName(e.target.value)} required />  
 </div>  
 <div>  
 <label>Description</label>  
 <textarea value={description} onChange={(e) => setDescription(e.target.value)} required />  
 </div>  
 <button type="submit">Submit</button>  
 </form>  
 {successMessage && <p>{successMessage}</p>}  
 {errorMessage && <p>{errorMessage}</p>}  
 </div>  
 );  
}  
  
export default CategoryUpload;

# FishUpload Component

The `FishUpload` component handles the submission of fish data, including an image, name,   
description, price, and category. It also fetches available categories from the backend.

import React, { useState, useEffect } from 'react';  
import axios from 'axios';  
  
function FishUpload() {  
 const [image, setImage] = useState(null);  
 const [name, setName] = useState('');  
 const [description, setDescription] = useState('');  
 const [price, setPrice] = useState('');  
 const [category, setCategory] = useState('');  
 const [categories, setCategories] = useState([]);  
 const [successMessage, setSuccessMessage] = useState('');  
 const [errorMessage, setErrorMessage] = useState('');  
  
 useEffect(() => {  
 axios.get('http://localhost:9000/admin-fish/api/categories/')  
 .then((res) => setCategories(res.data))  
 .catch((err) => console.error('Error fetching categories:', err));  
 }, []);  
  
 const handleSubmit = async (e) => {  
 e.preventDefault();  
  
 const formData = new FormData();  
 formData.append('name', name);  
 formData.append('description', description);  
 formData.append('price', price);  
 formData.append('category', category);  
 formData.append('image', image);  
  
 try {  
 await axios.post('http://localhost:9000/admin-fish/api/fish/', formData, {  
 headers: { 'Content-Type': 'multipart/form-data' },  
 });  
 setSuccessMessage('Fish successfully uploaded!');  
 setErrorMessage('');  
 setName('');  
 setDescription('');  
 setPrice('');  
 setCategory('');  
 setImage(null);  
 } catch (error) {  
 console.error('Error uploading fish:', error);  
 setSuccessMessage('');  
 setErrorMessage('Error uploading fish');  
 }  
 };  
  
 return (  
 <div>  
 <h2>Add a New Fish</h2>  
 <form onSubmit={handleSubmit}>  
 <div>  
 <label>Fish Name</label>  
 <input type="text" value={name} onChange={(e) => setName(e.target.value)} required />  
 </div>  
 <div>  
 <label>Description</label>  
 <textarea value={description} onChange={(e) => setDescription(e.target.value)} required />  
 </div>  
 <div>  
 <label>Price</label>  
 <input type="number" value={price} onChange={(e) => setPrice(e.target.value)} required />  
 </div>  
 <div>  
 <label>Category</label>  
 <select value={category} onChange={(e) => setCategory(e.target.value)} required>  
 <option value="">Select Category</option>  
 {categories.map((cat) => (  
 <option key={cat.id} value={cat.id}>{cat.name}</option>  
 ))}  
 </select>  
 </div>  
 <div>  
 <label>Image</label>  
 <input type="file" onChange={(e) => setImage(e.target.files[0])} required />  
 </div>  
 <button type="submit">Submit</button>  
 {successMessage && <p>{successMessage}</p>}  
 {errorMessage && <p>{errorMessage}</p>}  
 </form>  
 </div>  
 );  
}  
  
export default FishUpload;