import React from 'react';

function Header() {

return (

<header>

<h1>Welcome to My Portfolio</h1>

<nav>

<ul>

<li><a href="#about">About</a></li>

<li><a href="#projects">Projects</a></li>

</ul>

</nav>

</header>

);

}

export default Header;

import React from 'react';

function About() {

return (

<section id="about">

<h2>About Me</h2>

<p>Hello! I'm a software engineer with a passion for building web applications.</p>

</section>

);

}

export default About;

import React from 'react';

function Projects() {

return (

<section id="projects">

<h2>My Projects</h2>

<ul>

<li>Project One: A web application for managing tasks</li>

<li>Project Two: A portfolio website built with React</li>

</ul>

</section>

);

}

export default Projects;

import React from 'react';

function Footer() {

return (

<footer>

<p>&copy; 2024 My Portfolio</p>

</footer>

);

}

export default Footer;

import React from 'react';

import Header from './components/Header';

import About from './components/About';

import Projects from './components/Projects';

import Footer from './components/Footer';

import './App.css';

function App() {

return (

<div className="App">

<Header />

<About />

<Projects />

<Footer />

</div>

);

}

export default App;

body {

font-family: Arial, sans-serif;

margin: 0;

padding: 0;

background-color: #f4f4f4;

color: #333;

}

header {

background-color: #333;

color: white;

padding: 10px 0;

}

header h1 {

margin: 0;

text-align: center;

}

nav ul {

list-style: none;

padding: 0;

text-align: center;

}

nav ul li {

display: inline;

margin: 0 10px;

}

section {

padding: 20px;

}

footer {

background-color: #333;

color: white;

text-align: center;

padding: 10px 0;

position: fixed;

bottom: 0;

width: 100%;

}

from django.db import models

class About(models.Model):

name = models.CharField(max\_length=100)

bio = models.TextField()

def \_\_str\_\_(self):

return self.name

class Project(models.Model):

title = models.CharField(max\_length=200)

description = models.TextField()

def \_\_str\_\_(self):

return self.title

from rest\_framework import serializers

from .models import About, Project

class AboutSerializer(serializers.ModelSerializer):

class Meta:

model = About

fields = ['name', 'bio']

class ProjectSerializer(serializers.ModelSerializer):

class Meta:

model = Project

fields = ['title', 'description']

from rest\_framework import generics

from .models import About, Project

from .serializers import AboutSerializer, ProjectSerializer

class AboutList(generics.ListAPIView):

queryset = About.objects.all()

serializer\_class = AboutSerializer

class ProjectList(generics.ListAPIView):

queryset = Project.objects.all()

serializer\_class = ProjectSerializer

from rest\_framework import generics

from .models import About, Project

from .serializers import AboutSerializer, ProjectSerializer

class AboutList(generics.ListAPIView):

queryset = About.objects.all()

serializer\_class = AboutSerializer

class ProjectList(generics.ListAPIView):

queryset = Project.objects.all()

serializer\_class = ProjectSerializer

from django.urls import path

from .views import AboutList, ProjectList

urlpatterns = [

path('about/', AboutList.as\_view(), name='about-list'),

path('projects/', ProjectList.as\_view(), name='project-list'),

]

from django.contrib import admin

from django.urls import path, include

urlpatterns = [

path('admin/', admin.site.urls),

path('api/', include('portfolio.urls')),

]

from django.contrib import admin

from .models import About, Project

admin.site.register(About)

admin.site.register(Project)

python manage.py runserver

three-tier architecture would be implemented based on the React frontend and Django backend we discussed:

**1. Presentation Layer (Frontend)**

**Technology Used**: React.js

* **Purpose**: This layer is responsible for presenting the user interface and interacting with the user. It sends requests to the backend and displays the responses.
* **Components**:
  + **React Components**: Components like Header, About, Projects, and Footer are responsible for rendering the user interface.
  + **Data Fetching**: Uses fetch or axios to send HTTP requests to the Django backend to retrieve data for About and Projects.

**Flow**:

* The user interacts with the React application.
* The React application sends HTTP requests to the backend to fetch the required data.

**2. Application Logic Layer (Backend)**

**Technology Used**: Django (with Django REST Framework)

* **Purpose**: This layer contains the business logic of the application. It processes client requests, performs the necessary computations or database queries, and returns the processed data to the frontend.
* **Components**:
  + **Django Views**: Views like AboutList and ProjectList are responsible for handling incoming HTTP requests and returning serialized data.
  + **Serializers**: Converts Django models into JSON to be sent as a response.
  + **API Endpoints**: /api/about/ and /api/projects/ are provided for the frontend to retrieve data.

**Flow**:

* The backend receives HTTP requests from the frontend.
* The Django views handle the requests, interact with the data layer (database) to fetch or manipulate data, and return JSON responses.

**3. Data Layer (Database)**

**Technology Used**: SQLite (or any other relational database like PostgreSQL, MySQL, etc.)

* **Purpose**: This layer is responsible for data storage and management. It handles the CRUD (Create, Read, Update, Delete) operations on the data.
* **Components**:
  + **Django Models**: Represent the data structure and are used to interact with the database.
  + **Database**: Stores data for About and Project entities.

**Flow**:

* Django ORM (Object-Relational Mapping) interacts with the database to perform operations such as querying About and Project records.
* The data is then serialized by Django serializers and returned to the application logic layer.

**Summary of the Three-Tier Architecture**

1. **Presentation Layer (React.js)**:
   * Handles user interaction and presentation.
   * Sends requests to the Django backend and displays the data returned.
2. **Application Logic Layer (Django)**:
   * Contains the business logic and serves as an intermediary between the frontend and the database.
   * Processes requests, interacts with the database, and returns the necessary data to the frontend.
3. **Data Layer (Database)**:
   * Manages the storage and retrieval of application data.
   * Interacts with Django models to store and fetch data.

**Example Workflow in the Three-Tier Architecture**

1. **User Interaction**: The user visits the React.js frontend.
2. **Data Request**: The React app requests data from the Django backend via /api/about/ or /api/projects/ endpoints.
3. **Business Logic Processing**: Django receives the request, processes it, queries the database, and retrieves the required data.
4. **Data Retrieval**: The database returns the data to Django, which then serializes it into JSON.
5. **Response**: Django sends the JSON data back to the React app.
6. **Display**: The React app receives the data and updates the UI accordingly.