**To build a portfolio with React.js, three.js (or Nest.js), and Node.js as requested, here’s a detailed documentation on how it can be done:**

**Project Overview:**

**You want to create a portfolio that showcases your skills, experiences, and social media content using modern web technologies. The front end will be built using React.js and three.js or Nest.js for 3D effects and interactions, while the backend will be powered by Node.js.**

**1. Tech Stack:**

**• Frontend:**

**• React.js: For building dynamic and interactive user interfaces.**

**• three.js or Nest.js: For 3D content and interactions (choose three.js if you want to add 3D graphics or animations, Nest.js can be used for the backend and API management if necessary).**

**• Backend:**

**• Node.js: To handle server-side logic, API routing, and database interactions.**

**• Database:**

**• MongoDB or PostgreSQL: For storing project and personal data.**

**• Hosting:**

**• Heroku/Vercel: For hosting the backend (Node.js).**

**• Netlify/Vercel: For deploying the front-end (React.js).**

**2. Project Setup:**

**2.1. Frontend (React.js & three.js):**

**1. Install Node.js and npm:**

**• Download and install the latest stable version of Node.js from the official site: https://nodejs.org/.**

**2. Create a React.js project:**

**npx create-react-app portfolio**

**cd portfolio**

**npm start**

**3. Install three.js (if you’re using it):**

**npm install three**

**4. Set up three.js:**

**• Create a new component for the 3D effects (e.g., src/components/ThreeDScene.js).**

**• Import three.js and set up a basic 3D scene (camera, lights, objects).**

**5. Set up routing:**

**• Install React Router for multiple pages (About, Projects, Contact):**

**npm install react-router-dom**

**• Configure the routes (e.g., src/App.js):**

**import { BrowserRouter as Router, Route, Routes } from 'react-router-dom';**

**import HomePage from './pages/HomePage';**

**import ProjectsPage from './pages/ProjectsPage';**

**import ContactPage from './pages/ContactPage';**

**function App() {**

**return (**

**<Router>**

**<Routes>**

**<Route path="/" element={<HomePage />} />**

**<Route path="/projects" element={<ProjectsPage />} />**

**<Route path="/contact" element={<ContactPage />} />**

**</Routes>**

**</Router>**

**);**

**}**

**export default App;**

**6. Styling the portfolio:**

**• Use CSS Modules, Styled Components, or SASS to make your portfolio visually appealing.**

**• Ensure responsiveness using media queries.**

**2.2. Backend (Node.js with Express.js or Nest.js):**

**1. Initialize the Node.js backend:**

**mkdir backend**

**cd backend**

**npm init -y**

**2. Install dependencies:**

**npm install express mongoose cors dotenv**

**3. Create a basic server:**

**• Create an index.js file:**

**const express = require('express');**

**const cors = require('cors');**

**const mongoose = require('mongoose');**

**const dotenv = require('dotenv');**

**dotenv.config();**

**const app = express();**

**app.use(cors());**

**app.use(express.json());**

**mongoose.connect(process.env.MONGO\_URI, { useNewUrlParser: true, useUnifiedTopology: true })**

**.then(() => console.log("MongoDB connected"))**

**.catch(err => console.log(err));**

**app.get('/', (req, res) => {**

**res.send('Portfolio API');**

**});**

**const PORT = process.env.PORT || 5000;**

**app.listen(PORT, () => {**

**console.log(Server running on port ${PORT});**

**});**

**4. Define API routes:**

**• Add a route to manage the projects or portfolio data:**

**app.get('/projects', (req, res) => {**

**// Retrieve projects from database**

**});**

**app.post('/projects', (req, res) => {**

**// Save new project data**

**});**

**5. Connect with the Frontend:**

**• Use axios or fetch in React to connect the frontend with this backend.**

**• Example:**

**import axios from 'axios';**

**useEffect(() => {**

**axios.get('http://localhost:5000/projects')**

**.then(response => setProjects(response.data))**

**.catch(error => console.log(error));**

**}, []);**

**2.3. Database Setup (MongoDB):**

**1. Create a MongoDB cluster:**

**• Go to MongoDB Atlas and create a free cluster.**

**2. Create a .env file for storing sensitive data:**

**MONGO\_URI=your-mongo-db-connection-string**

**3. Define the project schema:**

**• Create a models/Project.js file in the backend:**

**const mongoose = require('mongoose');**

**const ProjectSchema = new mongoose.Schema({**

**title: String,**

**description: String,**

**image: String,**

**url: String**

**});**

**module.exports = mongoose.model('Project', ProjectSchema);**

**3. Deployment:**

**3.1. Frontend Deployment:**

**• Use Netlify or Vercel to deploy your React application.**

**• Connect your GitHub repository and deploy the project.**

**3.2. Backend Deployment:**

**• Use Heroku to deploy the Node.js backend.**

**• Set up the environment variables for MONGO\_URI and any other necessary keys.**

**4. Advanced Features:**

**• Integrating Animations: Use three.js or any other animation library like GSAP to create interactive animations.**

**• Authentication: Add user authentication if you want private sections (admin login to update projects).**

**• CMS Integration: Optionally, use Contentful or Strapi as a headless CMS for managing content.**

**5. Conclusion:**

**This setup will allow you to build a fully functional portfolio with 3D animations using React.js, three.js, and Node.js. You can further customize the look and add features such as dynamic data fetching from the backend, portfolio item management, and deployment to the web.**

**Would you like more detailed guidance on any specific parts of this setup?**