Step 1: Project Setup

REST and `json-server`

- **`json-server`**: Provides a full fake REST API for testing and prototyping. It watches the db.json file and provides endpoints for CRUD operations.
- **Endpoints**:
  - `GET /products`: Fetch all products.
  - `POST /products`: Add a new product.
  - `PUT /products/:id`: Update an existing product.
  - `DELETE /products/:id`: Delete a product.

React Components

- **App Component**: Manages the state of the application (selected product, adding state) and conditionally renders `ProductList`, `ProductForm`, or `Product` based on the current state.
- **`ProductList` Component**: Fetches and displays the list of products, handles product selection for editing, and deletion.
- **`ProductForm` Component**: Handles the form for adding a new product and posts the new product to the REST API.
- **`Product Component`**: Handles the form for editing an existing product and updates the product via the REST API.

CSS (App.css)

- Provides styling for the application to ensure a clean, consistent look across all components.
- Styles include layout, typography, form elements, buttons, and hover effects.

This setup creates a full-featured, responsive, and visually appealing product management application that interacts seamlessly with a REST API provided by json-server.

1.1 Create a React Application

First, create a new React application if you haven't already:

```
npx create-react-app my-app
cd my-app
```

1.2 Set Up `json-server`

Install `json-server` globally, might have to use `sudo`:

```
npm install -g json-server
```

Create a `db.json` file in the root of your project to serve as the database for `json-server`:

```
{
  "products": [
    {
      "id": 1,
      "name": "React Book",
      "description": "Great book about React.",
      "price": "$120",
      "stock": 10
    },
    {
      "id": 2,
      "name": "ES6 Book",
      "description": "The basis of React.",
      "price": "$80",
      "stock": 20
    }
  ]
}
```

Start `json-server` to serve the database:

```
json-server --watch db.json --port 5000
```

This starts a RESTful API server at http://localhost:5000.

Step 2: Project Structure

Ensure your project structure looks like this:

```
my-app/
├── public/
├── src/
│   ├── components/
```

```
│   │   ├── Product.js
│   │   ├── ProductList.js
│   │   └── ProductForm.js
│   ├── App.js
│   ├── App.css
│   └── index.js
├── db.json
└── package.json
```

Step 3: CSS Styling

`src/App.css`

Create a CSS file to style your application:

```
/* src/App.css */

.App {
  font-family: 'Arial', sans-serif;
  padding: 20px;
  max-width: 800px;
  margin: 0 auto;
  background-color: #f7f7f7;
  border-radius: 10px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
}

h1 {
  font-size: 2em;
  margin-bottom: 20px;
  color: #333;
}

h2 {
  font-size: 1.5em;
  margin-bottom: 10px;
  color: #555;
}

ul {
  list-style-type: none;
  padding: 0;
```

```
}

ul li {
  padding: 10px;
  margin: 5px 0;
  border: 1px solid #ccc;
  background-color: white;
  cursor: pointer;
  display: flex;
  justify-content: space-between;
  align-items: center;
  border-radius: 5px;
}

ul li:hover {
  background-color: #f0f0f0;
}

button {
  padding: 10px 20px;
  font-size: 16px;
  cursor: pointer;
  background-color: #007bff;
  color: white;
  border: none;
  border-radius: 4px;
  transition: background-color 0.3s ease;
}

button:hover {
  background-color: #0056b3;
}

button:disabled {
  background-color: #cccccc;
  cursor: not-allowed;
}

form div {
  margin-bottom: 15px;
  display: flex;
  align-items: center;
}
```

```
form label {
  flex: 0 0 120px;
  margin-right: 10px;
  font-weight: bold;
  color: #555;
}

form input[type="text"], form input[type="number"] {
  flex: 1;
  padding: 10px;
  font-size: 16px;
  border: 1px solid #ccc;
  border-radius: 4px;
}

.form-container {
  background-color: #fff;
  padding: 20px;
  border: 1px solid #ccc;
  border-radius: 8px;
  box-shadow: 0 0 10px rgba(0, 0, 0, 0.1);
  margin-bottom: 20px;
}
```

Step 4: Components

src/components/ProductForm.js

Handles the form for adding new products:

```
import React, { useState } from 'react';

function ProductForm({ onAddProduct, onCancel }) {
  const [newProduct, setNewProduct] = useState({
    name: '',
    description: '',
    price: '',
    stock: ''
  });

  const handleChange = (e) => {
    const { name, value } = e.target;
```

```
    setNewProduct(prevState => ({
      ...prevState,
      [name]: value
    }));
  };

  const handleSubmit = (e) => {
    e.preventDefault();
    fetch('http://localhost:5000/products', {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(newProduct)
    })
      .then(response => response.json())
      .then(data => {
        console.log('Product added:', data);
        onAddProduct(data);
      })
      .catch(error => console.error('Error adding
product:', error));
  };

  return (
    <div className="form-container">
      <h2>Add New Product</h2>
      <form onSubmit={handleSubmit}>
        <div>
          <label>Name:</label>
          <input type="text" name="name"
value={newProduct.name} onChange={handleChange} required />
        </div>
        <div>
          <label>Description:</label>
          <input type="text" name="description"
value={newProduct.description} onChange={handleChange}
required />
        </div>
        <div>
          <label>Price:</label>
```

```
          <input type="text" name="price"
value={newProduct.price} onChange={handleChange} required /
>
        </div>
        <div>
          <label>Stock:</label>
          <input type="number" name="stock"
value={newProduct.stock} onChange={handleChange} required /
>
        </div>
        <button type="submit">Add Product</button>
        <button type="button" onClick={onCancel}>Cancel</
button>
      </form>
    </div>
  );
}

export default ProductForm;
```

src/components/Product.js

Handles the form for editing an existing product:

```
import React, { useState } from 'react';

function Product({ product, onBack }) {
  const [productData, setProductData] = useState(product);

  const handleSave = () => {
    fetch(`http://localhost:5000/products/${productData.id}
`, {
      method: 'PUT',
      headers: {
        'Content-Type': 'application/json'
      },
      body: JSON.stringify(productData)
    })
      .then(response => response.json())
      .then(data => {
        console.log('Product updated:', data);
        onBack();
```

```
      })
      .catch(error => console.error('Error updating
product:', error));
  };

  const handleChange = (e) => {
    const { name, value } = e.target;
    setProductData(prevState => ({
      ...prevState,
      [name]: value
    }));
  };

  return (
    <div className="form-container">
      <h2>Edit Product</h2>
      <form>
        <div>
          <label>Name:</label>
          <input type="text" name="name"
value={productData.name} onChange={handleChange} />
        </div>
        <div>
          <label>Description:</label>
          <input type="text" name="description"
value={productData.description} onChange={handleChange} />
        </div>
        <div>
          <label>Price:</label>
          <input type="text" name="price"
value={productData.price} onChange={handleChange} />
        </div>
        <div>
          <label>Stock:</label>
          <input type="number" name="stock"
value={productData.stock} onChange={handleChange} />
        </div>
        <button type="button" onClick={handleSave}>Save</
button>
        <button type="button" onClick={onBack}>Back to
list</button>
      </form>
    </div>
```

```
  );
}

export default Product;
```

src/components/ProductList.js

Displays the list of products:

```
import React, { useEffect, useState } from 'react';

function ProductList({ onSelectProduct }) {
  const [products, setProducts] = useState([]);

  useEffect(() => {
    fetchProducts();
  }, []);

  const fetchProducts = () => {
    fetch('http://localhost:5000/products')
      .then(response => response.json())
      .then(data => setProducts(data))
      .catch(error => console.error('Error fetching
products:', error));
  };

  const handleDelete = (productId) => {
    fetch(`http://localhost:5000/products/${productId}`, {
      method: 'DELETE',
    })
      .then(() => {
        fetchProducts();
      })
      .catch(error => console.error('Error deleting
product:', error));
  };

  return (
    <div>
      <h1>Product List</h1>
      <ul>
        {products.map((product) => (
```

```
          <li key={product.id}>
            <div className="product-details" onClick={() =>
onSelectProduct(product)}>
              {product.name}
            </div>
            <div className="product-actions">
              <button onClick={() =>
onSelectProduct(product)}>Edit</button>
              <button onClick={() =>
handleDelete(product.id)}>Delete</button>
            </div>
          </li>
        ))}
      </ul>
    </div>
  );
}

export default ProductList;
```

src/App.js

Main application component:

```
import React, { useState } from 'react';
import ProductList from './components/ProductList';
import Product from './components/Product';
import ProductForm from './components/ProductForm';
import './App.css';

function App() {
  const [selectedProduct, setSelectedProduct] =
useState(null);
  const [isAdding, setIsAdding] = useState(false);

  const handleSelectProduct = (product) => {
    setSelectedProduct(product);
  };

  const handleBack = () => {
    setSelectedProduct(null);
    setIsAdding(false);
```

```
  };

  const handleAddProduct = (product) => {
    setIsAdding(false);
    setSelectedProduct(null); // Clear selected product
  };

  const handleStartAdding = () => {
    setIsAdding(true);
  };

  return (
    <div className="App">
      {isAdding ? (
        <ProductForm onAddProduct={handleAddProduct}
onCancel={handleBack} />
      ) : selectedProduct ? (
        <Product product={selectedProduct}
onBack={handleBack} />
      ) : (
        <>
          <ProductList
onSelectProduct={handleSelectProduct} />
          <button onClick={handleStartAdding}>Add Product</
button>
        </>
      )}
    </div>
  );
}

export default App;
```

Running the Application

1. **Start `json-server` and leave it running**:

   ```
   json-server --watch db.json --port 5000
   ```

2. **Start the development server**: `npm start`