React – JSON-server and Firebase Real Time Database

Question 1: What do you mean by RESTful web services?

RESTful web services are web APIs that follow the REST (Representational State Transfer) architecture style. They use standard HTTP methods like GET, POST, PUT, DELETE to perform CRUD operations on resources (like users, products, etc.).

Key Points:

- **REST uses URLs to access resources.**
- **Data is usually sent and received in JSON or XML format.**
- **It is stateless – every request is independent.**
- **Scalable and widely used for web and mobile apps.**

Example:

sql

CopyEdit

```
GET    /products      → Get all products
POST   /products      → Add a new product
PUT    /products/1    → Update product with ID 1
DELETE /products/1    → Delete product with ID 1
```

RESTful web services are APIs that use HTTP methods to perform operations on resources in a simple, scalable way.

Question 2: What is Json-Server? How we use in React ?

Ans:- 📌 **What is JSON Server?**

JSON Server is a lightweight Node.js tool that allows you to create a fake REST API using a simple JSON file (db.json).
It is commonly used for frontend development and testing without needing a real backend server.

🔍 **Key Features:**

- **Create a full REST API with zero coding.**

- Supports GET, POST, PUT, DELETE.
- Perfect for local testing during React or frontend development.

🛠 How to Use JSON Server in React (Step-by-Step)

✅ Step 1: Install JSON Server

bash

CopyEdit

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

✅ Step 2: Create a db.json file

json

CopyEdit

```
{
 "products": [
   { "id": 1, "name": "Shirt", "price": 500 },
   { "id": 2, "name": "Pant", "price": 700 }
 ]
}
```

✅ Step 3: Start JSON Server

bash

CopyEdit

```
json-server --watch db.json --port 3001
```

➡ Your fake API is now live at:
http://localhost:3001/products

✅ Step 4: Use it in React with Axios

js

CopyEdit

```
import axios from "axios";
import { useEffect, useState } from "react";

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

```
  useEffect(() => {
    axios.get("http://localhost:3001/products")
      .then(res => setProducts(res.data))
      .catch(err => console.error(err));
  }, []);

  return (
    <div>
      {products.map((item) => (
        <p key={item.id}>{item.name} - ₹{item.price}</p>
      ))}
    </div>
  );
}
```

✅ One-Line Summary:

JSON Server is a fake REST API used for testing React apps locally using a JSON file as a database.

Question 3: How do you fetch data from a Json-server API in React? Explain the role of fetch() or axios() in making API requests.

Ans:- Explain the role of fetch() or axios() in making API requests.

📌 How to Fetch Data from JSON Server in React?

To fetch data from a JSON Server in React, you typically use either:

- fetch() – a built-in JavaScript method
- axios – a third-party HTTP client library

Both are used to send HTTP requests (GET, POST, PUT, DELETE) to the API.

▯ Example using fetch():

js

CopyEdit

```
import { useEffect, useState } from "react";
```

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

  useEffect(() => {
    fetch("http://localhost:3001/products")
      .then((response) => response.json())
      .then((data) => setProducts(data))
      .catch((error) => console.error("Error:", error));
  }, []);

  return (
    <div>
      {products.map((item) => (
        <p key={item.id}>{item.name} - ₹{item.price}</p>
      ))}
    </div>
  );
}
```

Example using axios():

js

CopyEdit

```
import axios from "axios";
import { useEffect, useState } from "react";

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

  useEffect(() => {
    axios.get("http://localhost:3001/products")
      .then((response) => setProducts(response.data))
      .catch((error) => console.error("Error:", error));
  }, []);
  return (
    <div>
      {products.map((item) => (
        <p key={item.id}>{item.name} - ₹{item.price}</p>
      ))}
    </div>
  );
}
```

🔍 Role of fetch() and axios() in API Requests:

Feature	fetch()	axios()
Type	Built-in JavaScript function	External library (npm install axios)
Syntax	Promise-based	Promise-based (simpler syntax)
JSON parsing	Manual (.json()) needed	Auto-parses JSON
Error handling	Slightly manual	Built-in .catch() and better support
Headers, etc.	More verbose	Easier and cleaner

✅ One-Line Summary:

You can fetch data from a JSON Server API in React using fetch() or axios(), which send HTTP requests to retrieve and display data from the backend.

Question 4: What is Firebase? What features does Firebase offer?

Ans:- Firebase is a Backend-as-a-Service (BaaS) platform developed by Google, which helps developers build and scale web and mobile applications quickly and easily.

It offers real-time databases, authentication, hosting, storage, and many other backend services — so you don't need to build a backend from scratch.

🚀 Key Features of Firebase:

🔢 Feature	Description
1️⃣ Realtime Database	NoSQL cloud database that syncs data in real-time across clients.
2️⃣ Firestore (New DB)	Scalable NoSQL database with powerful querying and offline support.

🔢 Feature	Description
3️⃣ Authentication	Easy login via Email/Password, Google, Facebook, Phone, etc.
4️⃣ Firebase Hosting	Free and fast hosting for static sites and SPAs like React apps.
5️⃣ Cloud Storage	Store user-generated files like images, videos, PDFs securely.
6️⃣ Firebase Functions	Write backend logic with serverless cloud functions (like Node.js APIs).
7️⃣ Analytics & Crashlytics	Monitor app usage, user behavior, and crashes in real-time.
8️⃣ Cloud Messaging (FCM)	Send push notifications to Android, iOS, and web apps.
9️⃣ Performance Monitoring	Track app performance, speed, and bottlenecks.

□ Why Use Firebase?

- No backend server needed
- Real-time sync (great for chat, live apps)
- Easy integration with React, Flutter, Android
- Secure and scalable

✅ One-Line Summary:

Firebase is a cloud platform by Google that offers real-time databases, authentication, hosting, and many powerful backend tools to build full-stack apps quickly.

Question 5: Discuss the importance of handling errors and loading states when working with APIs in React

Ans:- When you make API requests in React (using fetch() or axios()), the data doesn't arrive instantly. During this waiting period, and if something goes wrong, it's essential to:

- Show users that something is happening (Loading...)
- Handle any failures properly (Error messages)

This improves:

- ✅ User experience (UX)
- ✅ App reliability
- ✅ Debugging and maintenance

🎯 3 Key States to Handle:

State	Meaning	What to Show?
loading	Data is being fetched	Show a spinner or "Loading..."
error	Something went wrong (network, 404, etc.)	Show error message to user
data	API call successful	Show the actual content/data

 React Example (With axios):

jsx

CopyEdit

```
import { useEffect, useState } from "react";
import axios from "axios";

function Users() {
  const [users, setUsers] = useState([]);
  const [loading, setLoading] = useState(true);  // 🌀 loading state
  const [error, setError] = useState(null);      // ❌ error state

  useEffect(() => {
    axios.get("https://jsonplaceholder.typicode.com/users")
      .then((res) => {
        setUsers(res.data);
        setLoading(false);
      })
      .catch((err) => {
        setError("Failed to fetch users!");
        setLoading(false);
      });
  }, []);

  if (loading) return <p>Loading users...</p>;
  if (error) return <p>{error}</p>;

  return (
    <ul>
      {users.map(user => <li key={user.id}>{user.name}</li>)}
    </ul>
  );
}
```

✅ **Benefits of Handling Loading & Errors:**

- **▯ Guides users: Shows progress and status.**
- **🛑 Avoids crashes: Prevents the app from breaking if the API fails.**
- **🎨 Professional UX: Smooth experience even during slow network.**
- **▯ Clean fallback: You can retry or show a backup UI.**

✅ **One-Line Summary:**

Handling loading and error states in React ensures users know what's happening while data loads or when something fails — leading to a smoother, more reliable app experience.