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

**RESTful web services** are APIs (Application Programming Interfaces) that follow the principles of REST (Representational State Transfer). REST is an architectural style that uses standard HTTP methods like GET, POST, PUT, DELETE, etc., to perform operations on resources (like data objects).

**Key characteristics of RESTful services:**

* **Stateless:** Each request from client to server must contain all the necessary information.
* **Client-Server architecture:** The frontend (client) and backend (server) are separated.
* **Use of standard HTTP methods.**
* **Resource-based URLs:** Each piece of data is treated as a resource (e.g., /users, /products/1).
* **Response formats:** Typically JSON or XML.

**Question 2: What is JSON-Server? How do we use it in React?**

**JSON-Server** is a lightweight, fake REST API that lets you simulate a real backend using a simple JSON file. It’s often used for development or testing purposes.

**How to use JSON-Server in React:**

1. **Install JSON-Server:**

bash

CopyEdit

npm install -g json-server

1. **Create a db.json file** (your mock database):

json

CopyEdit

{

"users": [

{ "id": 1, "name": "John" },

{ "id": 2, "name": "Jane" }

]

}

1. **Start the JSON Server:**

bash

CopyEdit

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

1. **Access the API:** You can now make requests to http://localhost:3001/users.
2. **Use it in your React app** to fetch or modify data using fetch() or axios().

**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.**

In React, you can fetch data from a JSON-Server API using either fetch() (built-in) or axios (external library).

**Using fetch():**

jsx

CopyEdit

useEffect(() => {

fetch('http://localhost:3001/users')

.then(response => response.json())

.then(data => setUsers(data))

.catch(error => console.error('Error:', error));

}, []);

**Using axios:**

jsx

CopyEdit

import axios from 'axios';

useEffect(() => {

axios.get('http://localhost:3001/users')

.then(response => setUsers(response.data))

.catch(error => console.error('Error:', error));

}, []);

**Role of fetch() and axios():**

* Both are used to send HTTP requests to APIs.
* fetch() is built into modern browsers but requires manual JSON parsing and error handling.
* axios simplifies the process, handles JSON automatically, and provides better error messages.

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

**Firebase** is a Backend-as-a-Service (BaaS) platform developed by Google that helps you build and scale web and mobile apps quickly.

**Key features of Firebase:**

* **Authentication:** Provides easy sign-in options (email/password, Google, Facebook, etc.).
* **Realtime Database & Firestore:** Cloud-hosted NoSQL databases.
* **Firebase Hosting:** Fast and secure static hosting.
* **Cloud Functions:** Write backend code that responds to events.
* **Cloud Storage:** For storing user-generated content like images and videos.
* **Analytics & Performance Monitoring:** Helps track user behavior and app performance.
* **Crashlytics:** Real-time crash reporting.

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

**Handling errors and loading states** is critical for providing a smooth user experience and avoiding crashes.

**Importance:**

* **User Feedback:** Shows a loading spinner so users know data is being fetched.
* **Error Transparency:** Helps users understand if something went wrong (e.g., “Network Error”).
* **Avoid Broken UI:** Prevents rendering issues when data is not yet available.
* **Debugging:** Helps developers detect and fix issues with API calls.

**Example:**

jsx

CopyEdit

const [loading, setLoading] = useState(true);

const [error, setError] = useState(null);

const [data, setData] = useState([]);

useEffect(() => {

fetch('http://localhost:3001/users')

.then(res => {

if (!res.ok) throw new Error('Failed to fetch data');

return res.json();

})

.then(data => setData(data))

.catch(err => setError(err.message))

.finally(() => setLoading(false));

}, []);

**UI Feedback:**

jsx

CopyEdit

if (loading) return <p>Loading...</p>;

if (error) return <p>Error: {error}</p>;

return <UserList data={data} />;