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

**Answer:**  
RESTful web services are APIs that follow the principles of REST (Representational State Transfer). They use standard HTTP methods like GET, POST, PUT, and DELETE to interact with resources. These services are stateless, meaning each request contains all the information needed for the server to understand and respond. Data is usually returned in JSON or XML format.

**Example:**  
GET request to https://api.example.com/products returns a list of products.

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

**Answer:**  
JSON-Server is a tool that allows you to create a full fake REST API using a simple JSON file. It is useful for testing and prototyping frontend applications without needing a real backend server.

**How to use in React:**

1. Install JSON-Server:  
   npm install -g json-server
2. Create a db.json file:

{

"users": [

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

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

]

}

1. Run the server:  
   json-server --watch db.json --port 3001
2. Use it in React with fetch or axios:

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

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

.then(data => console.log(data));

**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.**

**Answer:**  
To fetch data from a JSON-Server API in React, you can use fetch() or axios(). Both are used to make HTTP requests.

* fetch() is a built-in JavaScript API that returns a Promise and helps make asynchronous calls.
* axios() is a popular external library that provides advanced features like automatic JSON conversion, request cancellation, and better error handling.

**Example using fetch() in React:**

useEffect(() => {

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

.then(res => res.json())

.then(data => setUsers(data))

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

}, []);

**Example using axios() in React:**

import axios from 'axios';

useEffect(() => {

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

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

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

}, []);

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

**Answer:**  
Firebase is a Backend-as-a-Service (BaaS) platform developed by Google. It provides tools and infrastructure to build, improve, and grow web and mobile apps. It helps developers build applications faster without managing servers.

**Key Features of Firebase:**

* **Authentication:** Easy user sign-up/sign-in with email, Google, Facebook, etc.
* **Cloud Firestore & Realtime Database:** NoSQL databases to store and sync data in real-time.
* **Firebase Hosting:** Fast and secure static web hosting.
* **Cloud Functions:** Serverless backend code triggered by events.
* **Cloud Storage:** Store and serve user-generated content like images and videos.
* **Analytics & Crashlytics:** App usage tracking and real-time crash reporting.
* **Push Notifications:** Send notifications to web and mobile users.

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

**Answer:**  
Handling errors and loading states is crucial when working with APIs in React to provide a smooth user experience and avoid app crashes.

**Why it’s important:**

* **User Feedback:** Shows users a loading spinner or message while data is being fetched.
* **Error Handling:** Gracefully handles issues like network failures, wrong endpoints, or server errors.
* **Prevents Crashes:** Avoids trying to render incomplete or missing data.
* **Better UX:** Keeps users informed and improves trust in your app.

**Example:**

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

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

useEffect(() => {

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

.then(res => {

if (!res.ok) throw new Error("Network error");

return res.json();

})

.then(data => setUsers(data))

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

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

}, []);