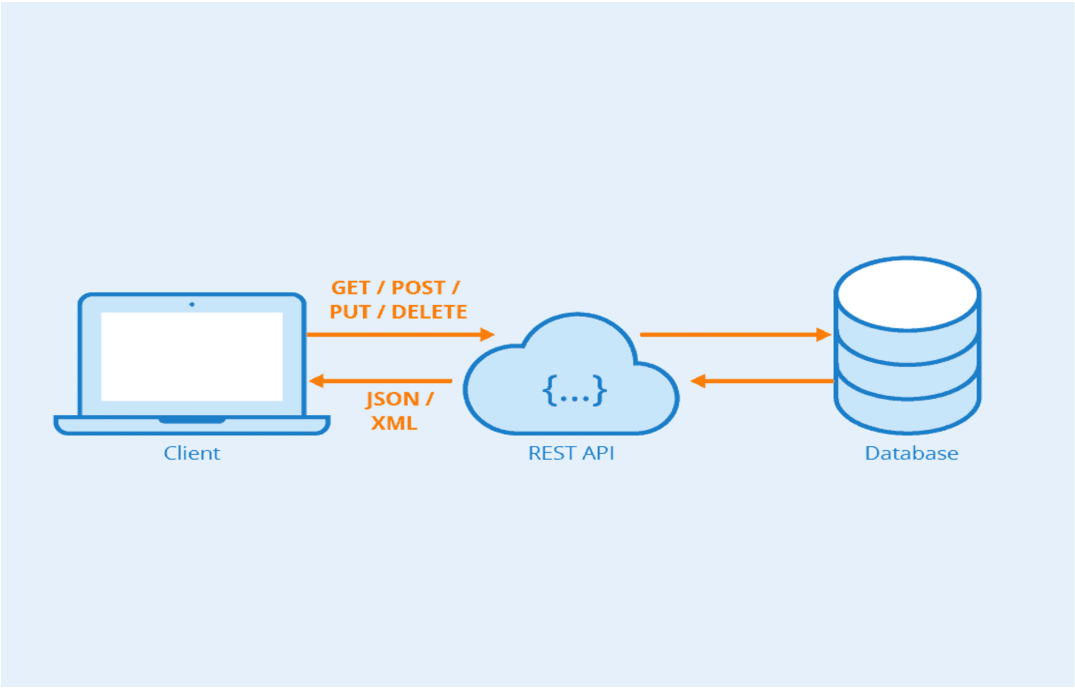
# React – JSON-server and Firebase Real Time Database

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

**Ans 1:** RESTful web services are a way of building web-based applications that use Representational State Transfer (REST) principles. They allow communication between different systems using standard web protocols like HTTP. In REST, each resource (such as a user, product, or post) is identified by a unique URI (Uniform Resource Identifier), and clients can perform operations on these resources using standard HTTP methods — GET (retrieve), POST (create), PUT (update), and DELETE (remove).

RESTful web services are lightweight, scalable, and stateless, meaning each request from a client contains all the information needed to process it, and the server doesn't store any client context. They commonly exchange data in JSON or XML format, making them simple to understand and integrate. Due to their flexibility and efficiency, RESTful APIs are widely used in modern web and mobile applications for smooth communication between frontend and backend systems.

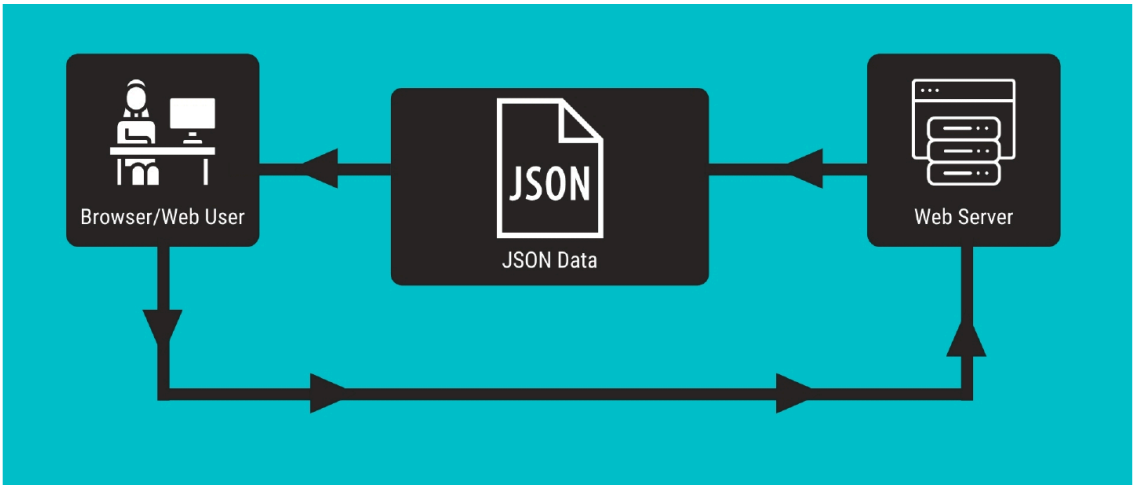


# React – JSON-server and Firebase Real Time Database

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

**Ans 2:** JSON-Server is a lightweight and easy-to-use tool that allows developers to create a mock RESTful API using a simple JSON file as a database. It helps simulate a real backend server without actually building one. With JSON-Server, you can perform all CRUD (Create, Read, Update, Delete) operations on data stored in the JSON file, making it very useful for frontend developers to test and develop applications quickly.

In React, we use JSON-Server by first installing it using npm install -g json-server. Then, we create a file like db.json to store sample data and start the server with json-server --watch db.json --port 3001. After that, we can fetch data in React using Axios or the Fetch API from the provided URL (for example, http://localhost:3001/posts). This helps simulate real API calls while developing and testing React components.



# React – JSON-server and Firebase Real Time 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.**

**Answer 3:**

To fetch data from a JSON-Server API in React, you can use either the `fetch()` function or the Axios library inside a component, usually within the `useEffect()` hook. This ensures the API request runs when the component mounts. For example, using `fetch()`,

useEffect(() => {

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

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

.then(data => setPosts(data))

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

}, []);

The `fetch()` function is a built-in JavaScript method that sends HTTP requests and returns a promise containing the response, while Axios is an external library that simplifies requests and automatically converts JSON data. Both are used to retrieve, send, update, or delete data from an API, helping React components interact with backend data dynamically and display it in the UI.

# React – JSON-server and Firebase Real Time Database

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

**Answer 4 :**

Firebase is a Backend-as-a-Service (BaaS) platform developed by Google that provides developers with a variety of tools and services to build, manage, and scale web and mobile applications easily. It helps reduce the need for complex backend coding by offering ready-to-use features like authentication, databases, hosting, and analytics — all integrated through a simple interface and SDKs for multiple platforms.

Firebase offers many powerful features, including:

- Realtime Database and Cloud Firestore for storing and syncing data across users instantly.
- Firebase Authentication for easy login using email, Google, Facebook, or other providers.
- Cloud Storage for storing images, videos, and files securely.
- Firebase Hosting for fast and secure web app deployment.
- Cloud Functions for running serverless backend code.
- Firebase Analytics for tracking user behavior and app performance.

  These features make Firebase an all-in-one solution for developing modern, scalable applications.

# React – JSON-server and Firebase Real Time Database

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

**Answer 5 :**

Handling errors and loading states in React when working with APIs is crucial for providing a smooth and user-friendly experience. When data is being fetched from an API, there's often a short delay before the data arrives. During this time, showing a loading indicator (like a spinner or "Loading..." message) informs users that the app is working and prevents confusion or repeated requests.

Error handling is equally important because network requests can fail due to issues like a poor internet connection, invalid endpoints, or server errors. By using try-catch blocks (in async functions) or .catch() with promises, developers can catch these errors and display helpful messages to users instead of crashing the app. Properly managing loading and error states makes React applications more reliable, responsive, and professional.