React – JSON-server and Firebase Real Time Database

1. What do you mean by RESTful web services?

Representational State Transfer, or REST, is a design pattern for interacting with resources stored in a server. Each resource has an identity, a data type, and supports a set of actions. The RESTful design pattern is normally used in combination with HTTP, the language of the internet.

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

json-server is a lightweight, mock REST API server that allows developers to quickly create a fake backend using a simple JSON file. It is useful for prototyping, front-end development, and testing without needing a real backend.

USE :

Step 1: Install JSON-Server

Step 2: Create a Fake Database (db.json)

Step 3: Start JSON-Server

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

Create a React Application.

Change your Project Directory.

Access the API Endpoint.

Import the useState() Hook and Set it to Hold Data.

Create a FetchInfo() Callback Function to Fetch and Store Data. Output.

Create a React Application.

Axios automatically transforms the data to and from JSON, while fetch() requires you to call response.

json() to parse the data to a JavaScript object.

Axios provides the data in the response object, while fetch() provides the response object itself, which contains other information such as status, headers, and url.

4. What is Firebase? What features does Firebase offer?

Firebase is a platform by Google that helps developers build and manage apps easily.

It provides tools for database storage, user authentication, hosting, notifications, and more—so you don’t have to build everything from scratch.

It's like a ready-made backend for web and mobile apps! All popular features are supported: Cloud Firestore, Cloud Functions, Authentication, Realtime Database, Cloud Storage, AdMob, AppCheck, Google Analytics, Cloud Messaging, Remote Config, Dynamic Links, Crashlytics, and Performance.

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

Handling errors and loading states in React when working with APIs is crucial for a smooth user experience.

Loading State: Show a spinner or message ("Loading...") while data is being fetched to prevent UI flickering.

Error Handling: Catch errors and display friendly messages ("Something went wrong") instead of breaking the app.