React – JSON-server and Firebase Real Time Database

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

ANS:- RESTful web services are web services built according to the Representational State Transfer (REST) architectural style. They leverage standard HTTP methods (GET, POST, PUT, DELETE) to interact with resources identified by URIs (Uniform Resource Identifiers) and utilize various representations (like JSON or XML) for data transfer. This approach promotes simplicity, scalability, and modifiability, making them well-suited for web-based applications.

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

ANS:- JSON Server is a Node.js module that enables the rapid creation of a mock RESTful API using a simple JSON file as the data source. It is particularly useful for front-end development, allowing developers to simulate a backend API for prototyping, testing, and building applications without the need for a full-fledged server or 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:- To fetch data from a JSON-server API in React, we typically use fetch() or axios() inside a useEffect() hook to make asynchronous API calls. fetch() is a built-in JavaScript method to make HTTP requests, while axios() is a third-party library that provides a simpler syntax and automatic JSON parsing. Both allow React apps to retrieve or modify data from the API and update the component state accordingly.

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

ANS:- Firebase is a comprehensive Backend-as-a-Service (BaaS) platform developed by Google that provides developers with a variety of tools and services to help build high-quality applications quickly and efficiently. It allows developers to build web, Android, and iOS apps without managing server infrastructure.

Firebase is especially popular for real-time applications, thanks to its powerful database and hosting features. It supports both frontend and backend development by offering ready-to-use backend services and APIs.

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

ANS:- Handling errors and loading states is crucial when working with APIs in React to provide a good user experience and ensure app stability. Without proper loading indicators, users may be confused during network delays, and unhandled errors (like failed requests) can break the UI or result in incorrect data being shown. Using conditional rendering and try-catch or .catch() blocks helps manage these states effectively.

Context API

Question 1: What is the Context API in React? How is it used to manage global state across multiple components?

ANS:- The React Context API is a feature that allows you to share data across components without having to manually pass props down through every level of the component tree. It provides a way to manage global state and make it accessible to any component that needs it, regardless of its position in the component hierarchy. This helps avoid "prop drilling," where data is passed through intermediate components that don't actually use it.

Question 2: Explain how createContext() and useContext() are used in React for sharing state.

ANS:- In React, `createContext()` and `useContext()` are part of the Context API, which is used to share data (or state) across components without having to pass props manually through every level of the component tree. This is especially useful for global state, such as user authentication status, theme settings, or language preferences.

`createContext()`

- The `createContext()` function is used to create a new Context object.
- This object comes with two components:
  - **Provider:** Used to wrap the part of the component tree that needs access to the context. It holds the actual value that will be shared.
  - **Consumer:** (less commonly used with modern React) Used to access the context value. However, `useContext()` is preferred in functional components.