React – JSON-server and Firebase Real Time Database

1. What do you mean by RESTful web services?

- RESTful web services are web services that follow the REST (Representational State Transfer) architecture.
- REST is a set of principles for designing networked applications where clients communicate with servers using standard HTTP methods.

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

- JSON-Server is a lightweight package that allows you to create a fake REST API using a simple JSON file as a database.
- It is useful for prototyping, testing, and developing front-end applications without needing a backend server.

❖ How we use in React: -

Step 1: Install JSON-Server.

- Open your Terminal and Run
  - npm install -g json-server

Step 2: Create a db.json File.

  - {
  - "users": [
  - { "id": 1, "name": "John Doe", "email": "john@example.com" }

Step 3: Start JSON-Server

  - json-server --watch db.json --port 5000

Step 4: Fetch Data in React

  - Now, you can use fetch or axios in your React components to interact with 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.**

    - To fetch data from a JSON-Server API in React, you can use either fetch () (built-in JavaScript method) or axios (third-party library). Both help in making HTTP requests to interact with APIs.

    1. Using fetch () (Native JavaScript API)

    - fetch () is a built-in JavaScript function that allows us to make HTTP requests. It returns a Promise, which resolves to the response data.

    2. Using axios (Third-Party Library)

    - axios is a popular library that simplifies HTTP requests. It automatically parses JSON responses and handles errors better than fetch ().

4. **What is Firebase? What features does Firebase offer?**
    - Firebase is a Backend-as-a-Service (BaaS) platform developed by Google that provides a suite of tools to build and manage web and mobile applications without managing servers.
    - It helps developers with authentication, real-time databases, cloud functions, hosting, and more.

    ❖ **What Features Offer by Firebase: -**

    1. **Authentication**

        - Provides user authentication with Email/Password, Google, Facebook, GitHub,Twitter,Phonenumber,etc.Supports OAuth-based authentication and multi-factor authentication (MFA).Easy integration with Firebase Authentication SDK.

2. Firebase Firestore (NoSQL Database)

- A cloud-hosted NoSQL database that stores data in collections and documents.Supports real-time synchronization, meaning changes in data are instantly updated across all clients. Offline support allows apps to work without an internet connection.

3. Firebase Realtime Database

- A JSON-based NoSQL database that allows real-time updates across all connected devices. Best for chat applications, live notifications, and collaborative apps. Supports offline mode.

4. Firebase Cloud Storage

- Store and serve images, videos, and other files securely. Provides automatic scaling and fast CDN delivery.
- Supports Google Cloud Storage integration.

5. Firebase Hosting

- Fast, secure, and free hosting for web apps. Provides custom domains, SSL, and global CDN for fast performance.

6. Firebase Cloud Functions

- Serverless backend code execution triggered by Firebase events.
- Can handle authentication events, database updates, and HTTP requests.

7. Firebase Cloud Messaging (FCM)

- Allows sending push notifications to users on web and mobile apps. Supports targeted notifications, scheduling, and analytics tracking.

8. Firebase Analytics

- Tracks user engagement, retention, and in-app behavior. Helps in understanding how users interact with your app. Integrated with Google Analytics.

9. Firebase Remote Config

- Update app features dynamically without publishing a new version. Helps in A/B testing and feature rollouts.

10. Firebase ML (Machine Learning Kit)

- Pre-trained and custom ML models for face detection, text recognition, and more. Supports on-device ML and cloud-based processing.

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

- When working with APIs in React, it's essential to handle errors and loading states properly to provide a better user experience and ensure that your application is reliable and responsive.

1. Why Handle Loading States?

- Before an API request is completed, the application doesn't immediately have the data. Showing a loading indicator improves UX by informing users that data is being fetched.

2. Why Handle Errors?

- APIs can fail due to various reasons like network issues, server downtime, or invalid requests. Handling errors properly helps prevent application crashes and provides useful messages to users.