React – JSON-server and Firebase Real Time Database

THEORY EXERCISE:

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

RESTful web services are web services that follow the principles of **REST (Representational State Transfer)**, an architectural style used for designing networked applications. In REST, resources (like data or objects) are identified using **URLs**, and operations on these resources are performed using standard **HTTP methods**:

- **GET** – to retrieve data
- **POST** – to create new data
- **PUT** – to update existing data
- **DELETE** – to remove data

RESTful services are **stateless**, meaning each request from the client contains all the information needed for the server to understand and respond. They often use **JSON** or **XML** for data exchange and are known for being **simple, scalable, and easy to use** in web applications.

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

JSON-Server is a tool that allows developers to create a **fake REST API** using a simple **JSON file**. It is mainly used for testing and prototyping front-end applications without needing a real backend.

In React, we use JSON-Server to simulate backend data. By creating a JSON file with sample data and running the server, we get a fully functional REST API. React can then make **HTTP requests** (like GET, POST, PUT, DELETE) to this API using tools like fetch or axios, allowing us to test data fetching, state management, and UI behavior during development.

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.

To fetch data from a JSON-Server API in React, we use **HTTP request methods** like GET, POST, PUT, or DELETE to communicate with the fake backend. This is commonly done inside the **useEffect** hook to load data when the component mounts.

fetch() and **axios()** are tools used to make these API requests:

- **fetch()** is a built-in JavaScript method for making network requests. It returns a promise and requires manual handling of JSON conversion.
- **axios()** is an external library that simplifies HTTP requests. It automatically handles JSON data and provides a cleaner syntax.

Both are used to send requests to the JSON-Server and receive responses, which can then be stored in state and used in the UI.

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

Firebase is a **Backend-as-a-Service (BaaS)** platform developed by **Google** that helps developers build web and mobile applications faster without managing complex backend infrastructure.

Features offered by Firebase:

1. **Authentication** – Easy-to-use sign-in methods (email, password, Google, Facebook, etc.).
2. **Cloud Firestore & Realtime Database** – NoSQL databases to store and sync data in real time.
3. **Cloud Storage** – Store and serve user-generated files like images, videos, etc.
4. **Hosting** – Fast and secure static website hosting.
5. **Cloud Functions** – Write backend code that runs on Firebase servers.
6. **Analytics** – Track user behavior with Google Analytics integration.
7. **Push Notifications** – Send messages to users using Firebase Cloud Messaging (FCM).

Firebase is widely used because it is scalable, secure, and easy to integrate into React and other front-end frameworks.

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

Handling **errors** and **loading states** is very important when working with APIs in React to ensure a smooth and user-friendly experience.

Importance of Loading State:

- Shows the user that the data is **being fetched**.

- Prevents confusion by displaying a **spinner or "Loading..." message** until the data is ready.
- Improves **user experience** and avoids showing empty or broken UI.

Importance of Error Handling:

- Catches issues like **network errors**, **server failures**, or **invalid responses**.
- Allows you to show **friendly error messages** instead of crashing the app.
- Helps developers **debug problems** easily.

In short, managing these states makes your React app more **reliable, responsive, and user-friendly**.