**JAVASCRIPT Assignment**

**MODULE: 18 (REACT – JSON-server and Firebase Real Time Database)**

1. What do you mean by RESTful web series?

Ans. A RESTful web service, or RESTful API, follows the principles of REST (Representational State Transfer), which is an architectural style for designing networked applications.

* **Resource-Based**: REST treats every component of a web service as a resource, which can be accessed and manipulated using standard HTTP methods (GET, POST, PUT, DELETE).
* **Stateless Communication**: Each request from a client to a server must contain all the information necessary to understand the request. The server should not store any client state between requests.
* **Uniform Interface**: Resources are manipulated through a common set of methods (HTTP verbs) and standardized responses. This simplifies the architecture and allows for decoupling between the client and server.
* **Client-Server Architecture**: The client and server are independent of each other and communicate through a uniform interface (HTTP).
* **Layered System**: A client can interact with the server without knowing the underlying details of how the server handles its requests and serves resources.

1. What is JSON server? How we use in React?

Ans. JSON-Server is a lightweight and quick way to set up a mock REST API using a simple JSON file as a database. It allows developers to simulate API responses without needing a backend, making it ideal for prototyping and testing frontend applications.

Following is the way how is JSON-server is used in a React Application:

1. **Install JSON-Server globally or locally in your project.**

# Install globally

npm install -g json-server

# OR install locally

npm install json-server --save-dev

1. **Create a db.json file in your project root directory.**

{ "posts": [ { "id": 1, "title": "Post 1", "content": "This is post 1" }, { "id": 2, "title": "Post 2", "content": "This is post 2" } ], "users": [ { "id": 1, "name": "John Doe" }, { "id": 2, "name": "Jane Smith" } ] }

1. **Run JSON-Server and point it to the db.json file.**
2. **Use fetch or a library like axios to make requests to the JSON-Server API.**

import React, { useEffect, useState } from "react";

import axios from "axios";

const App = () => {

const [posts, setPosts] = useState([]);

useEffect(() => {

// Fetch data from JSON-Server

axios.get("http://localhost:3001/posts")

.then((response) => setPosts(response.data))

.catch((error) => console.error("Error fetching data:", error));

}, []);

return (

<div>

<h1>Posts</h1>

<ul>

{posts.map(post => (

<li key={post.id}>{post.title}: {post.content}</li>

))}

</ul>

</div>

);

};

export default App;

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

Ans. Fetching data from a JSON-Server API in React involves making HTTP requests to the server endpoints to retrieve, send, update, or delete data. This is done using tools like fetch() or axios().

1. **Start JSON-Server:**
2. **Fetch data using fetch()**

import React, { useState, useEffect } from "react";

const FetchExample = () => {

const [posts, setPosts] = useState([]);

useEffect(() => {

// Fetch data from JSON-Server

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

.then((response) => response.json()) // Parse JSON response

.then((data) => setPosts(data)) // Set fetched data in state

.catch((error) => console.error("Error fetching data:", error)); // Handle errors

}, []);

return (

<div>

<h1>Posts</h1>

<ul>

{posts.map((post) => (

<li key={post.id}>{post.title}: {post.content}</li>

))}

</ul>

</div>

);

};

export default FetchExample;

1. **Fetch data using axios()**

import React, { useState, useEffect } from "react";

import axios from "axios";

const AxiosExample = () => {

const [posts, setPosts] = useState([]);

useEffect(() => {

// Fetch data from JSON-Server

axios.get("http://localhost:3001/posts")

.then((response) => setPosts(response.data)) // Set fetched data in state

.catch((error) => console.error("Error fetching data:", error)); // Handle errors

}, []);

return (

<div>

<h1>Posts</h1>

<ul>

{posts.map((post) => (

<li key={post.id}>{post.title}: {post.content}</li>

))}

</ul>

</div>

);

};

export default AxiosExample;

**# Role of fetch() and axios()**

**fetch()**

* Native JavaScript API for making HTTP requests.
* Returns a Promise that resolves to a Response object.
* Requires manual parsing of the response (e.g., .json() for JSON data).

**axios()**

* A **third-party library** that simplifies HTTP requests.
* Automatically parses the response into JSON.
* Handles errors better than fetch().
* Supports additional features like interceptors, timeouts, and request cancellation.

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

Ans. Firebase is a Google-backed platform that provides tools and services to help developers build, manage, and scale web and mobile applications efficiently. It offers features like Realtime Database and Firestore for real-time data synchronization, Authentication for secure user login, Cloud Functions for serverless backend logic, Hosting for deploying web apps, Cloud Storage for file storage, Analytics for tracking user behavior, Crashlytics for crash reporting, and Cloud Messaging for sending notifications. Firebase simplifies development with cross-platform support and scalability.

1. Discuss the importance of handling errors and loading states when working withAPIs in React.

Ans. Handling errors and loading states is crucial when working with APIs in React to ensure a seamless user experience. Proper management of loading states helps users understand that data is being fetched or processed, preventing confusion or repeated actions. Meanwhile, handling errors ensures that users receive clear feedback if something goes wrong, like network issues or API failures, enabling them to retry or take corrective action. Together, these practices make applications more reliable, user-friendly, and professional.