**React JS**

Module-16) React – Json-Server and Firebase Real Time Database

**THEORY EXERCISE**

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

**RESTful web services** are APIs that follow the **REST (Representational State Transfer)** architectural style. These services use standard **HTTP methods** like GET, POST, PUT, DELETE to perform operations on data.

**Key principles:**

* **Stateless**: Each request is independent.
* **Resources**: Represented by URLs (e.g., /users/1)
* **Use of HTTP verbs**:
  + GET: Read data
  + POST: Create data
  + PUT/PATCH: Update data
  + DELETE: Remove data

**Question 2: What is JSON-Server? How do we use it in React?**

**JSON-Server** is a fake REST API server for frontend development. It allows you to mock a backend using a simple JSON file.

**Use in React:**

1. Install it:
2. npm install -g json-server
3. create db.json file:
4. {
5. "users": [
6. { "id": 1, "name": "John" },
7. { "id": 2, "name": "Alice" }
8. ]
9. }
10. Start server:
11. json-server --watch db.json --port 3001

we can access data at http://localhost:3001/users.

**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.**

You can use either the built-in fetch() or external library axios().

**Example using fetch():**

useEffect(() => {

fetch('http://localhost:3001/users')

.then(response => response.json())

.then(data => setUsers(data))

.catch(error => console.log(error));

}, []);

**Example using axios:**

import axios from 'axios';

useEffect(() => {

axios.get('http://localhost:3001/users')

.then(res => setUsers(res.data))

.catch(err => console.log(err));

}, []);

**Role:**

* fetch() is native JavaScript.
* axios() is simpler for complex requests (auto JSON conversion, better error handling).

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

**Firebase** is a **Backend-as-a-Service (BaaS)** platform by Google. It provides tools for building web and mobile apps quickly.

**Features:**

* **Realtime Database / Firestore**: NoSQL cloud databases
* **Authentication**: Google, Email/Password, Facebook, etc.
* **Hosting**: Deploy static websites
* **Cloud Functions**: Run backend code
* **Storage**: Upload and serve files
* **Analytics**, **Push Notifications**, and more

Firebase is widely used with React for authentication and database.

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

Proper handling ensures **user experience** and **debugging** is smooth.

**Why important?**

* **Loading state** helps indicate progress (e.g., spinner).
* **Error state** helps users and developers identify problems.

**Example:**

const [loading, setLoading] = useState(true);

const [error, setError] = useState(null);

useEffect(() => {

fetch('http://localhost:3001/users')

.then(res => {

if (!res.ok) throw new Error("Network Error");

return res.json();

})

.then(data => setUsers(data))

.catch(err => setError(err.message))

.finally(() => setLoading(false));

}, []);

**LAB EXERCISE**

• Task 1:

• Create a React component that fetches data from a public API (e.g., a list of

users)and displays it in a table format.

• Create a React app with Json-server and use Get , Post , Put , Delete & patch

method on Json-server API.

• Task 2:

• Create a React app crud and Authentication with firebase API.

• Implement google Authentication with firebase API.

• Task 3:

• Implement error handling and loading states for the API call. Display a

loadingspinner while the data is being fetched.