**React – Json-Server And Firebase Real Time Database**

Here's a full breakdown of your **theory and lab exercises** related to **RESTful APIs, JSON Server, Firebase, and API handling in React**:

**THEORY EXERCISE**

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

* **REST (Representational State Transfer)** is an architecture style for designing networked applications.
* RESTful web services use **HTTP methods** (GET, POST, PUT, DELETE) to perform operations on resources (data) via URLs.
* It's stateless and uses standard web protocols, making it simple, scalable, and fast.

**Question 2: What is JSON-Server? How is it used in React?**

* **JSON Server** is a tool that allows you to create a full fake REST API using a simple db.json file.
* It is used in React for **mocking backend APIs** during development.
* You can perform **CRUD operations** using HTTP methods.

**Installation:**

npm install -g json-server

**Usage:**

json-server --watch db.json --port 3001

**Question 3: How do you fetch data from a JSON-server API in React?**

**Role of fetch() or axios()**

* fetch() and axios() are used to make **HTTP requests** in React.
* You can use them inside useEffect() to call APIs from a JSON server.

**Example:**

useEffect(() => {

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

.then(res => res.json())

.then(data => setUsers(data))

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

}, []);

* axios is preferred for its simplicity and better error handling.

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

* **Firebase** is a platform by Google for building web and mobile applications.
* Key Features:
  + Realtime Database / Firestore
  + Authentication (Email, Google, Facebook, etc.)
  + Hosting
  + Cloud Functions
  + Cloud Storage
  + Analytics and Crash Reporting

**Question 5: Why is handling errors and loading states important?**

* Ensures a **better user experience**.
* Prevents the app from crashing or showing empty screens.
* A **loading spinner** indicates that something is happening.
* **Error messages** help inform users about problems (e.g., "No Internet").

**LAB EXERCISE**

**Task 1: Fetch Data from a Public API and Display in Table**

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

function UserTable() {

const [users, setUsers] = useState([]);

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

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/users')

.then(res => res.json())

.then(data => {

setUsers(data);

setLoading(false);

});

}, []);

if (loading) return <p>Loading...</p>;

return (

<table border="1">

<thead>

<tr><th>Name</th><th>Email</th><th>City</th></tr>

</thead>

<tbody>

{users.map(user => (

<tr key={user.id}>

<td>{user.name}</td><td>{user.email}</td><td>{user.address.city}</td>

</tr>

))}

</tbody>

</table>

);

}

export default UserTable;

**JSON Server CRUD App with GET, POST, PUT, DELETE, PATCH**

1. db.json

{

"posts": [

{ "id": 1, "title": "First Post", "content": "Hello World" }

]

}

1. Sample API requests in React (with axios)

npm install axios

import axios from 'axios';

const api = 'http://localhost:3001/posts';

// GET

axios.get(api).then(res => console.log(res.data));

// POST

axios.post(api, { title: "New Post", content: "Post content" });

// PUT

axios.put(`${api}/1`, { title: "Updated", content: "Updated content" });

// PATCH

axios.patch(`${api}/1`, { title: "Partially Updated" });

// DELETE

axios.delete(`${api}/1`);

**Task 2: CRUD App + Firebase Authentication**

1. **Install Firebase**

npm install firebase

1. **Firebase Config Setup**

// firebase.js

import { initializeApp } from "firebase/app";

import { getAuth, GoogleAuthProvider } from "firebase/auth";

const firebaseConfig = {

apiKey: "...",

authDomain: "...",

projectId: "...",

// rest of config

};

const app = initializeApp(firebaseConfig);

export const auth = getAuth(app);

export const googleProvider = new GoogleAuthProvider();

1. **Google Authentication Button**

import { auth, googleProvider } from './firebase';

import { signInWithPopup } from 'firebase/auth';

function Login() {

const handleLogin = () => {

signInWithPopup(auth, googleProvider)

.then(res => console.log(res.user))

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

};

return <button onClick={handleLogin}>Login with Google</button>;

}

1. Use Firestore for storing todo items (optional).

**Task 3: Loading Spinner and Error Handling**

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

function FetchComponent() {

const [data, setData] = useState([]);

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

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

useEffect(() => {

fetch('https://jsonplaceholder.typicode.com/posts')

.then(res => {

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

return res.json();

})

.then(data => setData(data))

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

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

}, []);

if (loading) return <p> Loading...</p>;

if (error) return <p> Error: {error}</p>;

return (

<ul>

{data.slice(0, 5).map(post => <li key={post.id}>{post.title}</li>)}

</ul>

);

}

export default FetchComponent;