THEORY EXERCISE

12. React – JSON-server and Firebase Real Time Database

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

- RESTful web services are APIs (ways for apps to talk to each other) that follow **REST** principles.
- **REST** stands for **Representational State Transfer** – it's a set of rules for building web APIs.
- REST APIs use **HTTP methods** like GET, POST, PUT, DELETE to interact with data.
- Example:
    - GET → get data
    - POST → send new data
    - PUT → update data
    - DELETE → delete data
- RESTful APIs are **stateless**, meaning every request is independent.

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

- **Json-Server** is a tool to create a **fake REST API** quickly.
- It uses a JSON file as a database – so you can build & test APIs without a real backend.
- In React, we use Json-Server for testing or demos.

How to use:

1. Install it:

npm install -g json-server

2. Create a db.json file (like a fake database).
3. Start the server:

CopyEdit

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

4. Now your API is running at http://localhost:3001!

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

- We use tools like **fetch()** (built-in JavaScript function) or **axios** (popular library) to get data from the API.

Example with fetch():

jsx

CopyEdit

```
useEffect(() => {
  fetch("http://localhost:3001/posts")
    .then((res) => res.json())
    .then((data) => {
      console.log(data); // show data
    })
    .catch((err) => console.log(err)); // handle error
}, []);
```

Example with axios:

```
import axios from "axios";
useEffect(() => {
  axios.get("http://localhost:3001/posts")
    .then((res) => {
      console.log(res.data); // show data
    })
    .catch((err) => console.log(err)); // handle error
}, []);
```

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

- **Firebase** is a platform by Google for building web and mobile apps.
- It's **free** to start and easy to use.

Key features:

✔ **Authentication** – Login, sign-up with Google, Facebook, email, etc.

✔ **Firestore Database** – Real-time database.

- ✔ **Storage** – Save files (images, videos).
- ✔ **Hosting** – Deploy your app for free.
- ✔ **Cloud Functions** – Write server code.
- ✔ **Analytics** – See how your app is used.
- ✔ **Push Notifications** – Send messages to users.

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

- **When you call an API**, it can take time to load data, and sometimes it can fail.
- That's why you need to:
  Show a **loading message** or spinner while the data is being fetched.
  Catch errors – so your app doesn't crash and shows a nice error message.

Example:

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

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

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

useEffect(() => {

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

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

    .then((data) => {

      setData(data);

      setLoading(false);

    })

    .catch((err) => {

      setError(err.message);

      setLoading(false);

    });

}, []);

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

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