Lecture 6 – Exploring the world

Monolith Architecture:

- Everything is in the same project, like API, UI, auth, DB, SMS.
- If a change is made, the whole project must be tested, compiled, and built.

Microservices Architecture:

- Different services for different processes. All these services combine together to form a big app.
- It follows the single responsibility principle.
- This is known as separation of concerns.

Food Project (UI Microservice):

- The project is being built as a UI microservice.
- All services run on their own ports. For example, the UI is running at port 1234 and the backend might be running at port 5314.
- All of them can be mapped to a domain name and can be called with /ui, /api, /sms, etc.

React Application Communication:

- The React application needs to communicate with other services.
- To see how React talks with the backend API, you can use the `fetch` or `axios` libraries in your React code to make HTTP requests to your backend service.

Here's a basic example of how you might set up a call from your React app (running on port 1234) to your backend (running on port 5314):

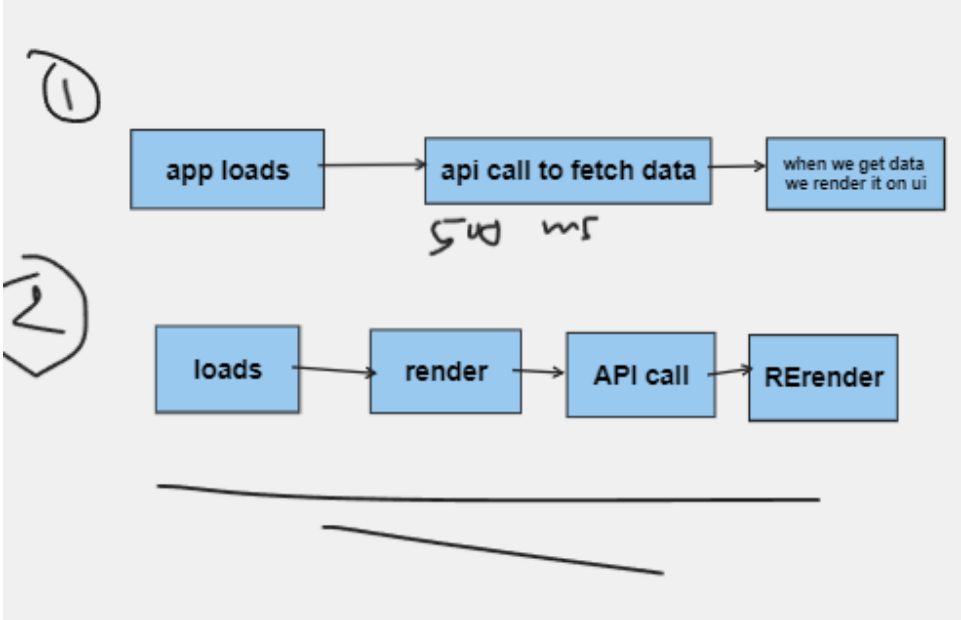
```
import axios from 'axios';

axios.get('http://localhost:5314/api')
  .then(response => {
    console.log(response.data);
  })
  .catch(error => {
    console.error('Error fetching data', error);
  });
```

This code sends a GET request to your backend and then logs the response data. You would replace '`http://localhost:5314/api`' with the actual URL of your backend service.

Please let me know if you want to add more notes or need information on another topic.

There are 2 ways we can make api calls



React Rendering Approach:

- The second approach is preferred in React because it provides a better user experience.
- Initially, the UI might not be visible as it needs to make an API call. But with the second approach, you can first see the skeleton of the UI, and then it gradually loads the full website.
- Also, React's render mechanism is very fast, so doing two renderings is not a big issue.

React UseEffect:

```
Syntax -   useEffect(()=>{} , []);
```

React's useEffect Hook:

- The `useEffect` hook in React has two parameters: a callback function and a dependency array.
- The callback function gets called as soon as React renders the component.
- If you need to do something after the component renders, you should put it inside the `useEffect` hook.

Fetching Data:

- As soon as the webpage renders (following the 2nd approach), an API call is made immediately. This process is illustrated in the diagram you mentioned.
- Therefore, the API call should be placed inside the `useEffect` hook.

```
const [listOfRestaurants , setListofRestaurant] = useState(resObj)

useEffect(()=>{
  fetchData();
} , []);
```

Example code:

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


function MyComponent() {
  const [data, setData] = useState(null);


  useEffect(() => {
    axios.get('http://localhost:5314/api')
      .then(response => {
        setData(response.data);
      })
      .catch(error => {
        console.error('Error fetching data', error);
      });
  }, []); // Empty dependency array means this effect runs once on mount


  return (
    <div>
      {/* Render your data here */}
    </div>
  );
}
```

In this code, `useEffect` runs once when the component mounts and fetches data from your backend. The fetched data is then stored in state with `setData`, and can be used in your component's render method.

Please let me know if you want to add more notes or need information on another topic

Now we will write the logic to fetch data.

Fetching Data in JavaScript

Introduction

- Fetching data in JavaScript is similar to other languages.
- We use a built-in browser function called `fetch`.

Fetch Function

- The `fetch` function is provided by the browser.
- It returns a Promise that resolves to the Response object representing the response to the request.

Async/Await

- Instead of using Promises directly, we can use `async/await` for cleaner and more readable code.

Optional Chaining

- With the help of optional chaining, we can prevent null errors.
- If the data is not present, it will show `undefined` instead of throwing a null error.

```
const Body = () => {
  // Local State Variable - Super powerful variable
  const [listOfRestaurants, setListOfRestraunt] = useState(resList);

  useEffect(()=>{
    fetchData();
  },[]);

  let fetchData = async () =>{
    const data = await fetch(
      "https://www.swiggy.com/dapi/restaurants/list/v5?lat=12.9715987&lng=
    )
    const json = await data.json();
    console.log(json)

    // optional chaining
    setListOfRestraunt(json?.data?.cards[2]?.data?.data?.cards);
  }
```

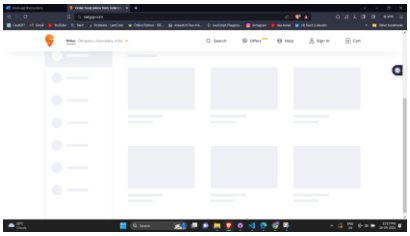
Even though we have fetched data first it will not render the Api , it will only give skeletons and after that it will load the data since we have given it inside useEffect().

The page rendered very fast even though it was empty and after that it fetched data and re rendered because we updated data with set method from use state()

But is that a good experience?? it will be showing loading till a point and suddenly the fetched data pops up.

Here shimmer ui comes into picture.

Kind of we load a fake page until we fetch actual json.



Here you can see some tile like structure right this is done by shimmer ui.

Now I will let you know something amazing , wht do we use starte variables instead of normal js variables and also we are creating state variables wirth const keyword ryt , thenhow can we update it??

1> js can not be used because even though we can change the value it will not re render itself

2> state variable will rerender the whole component once again so no need of manually doing it

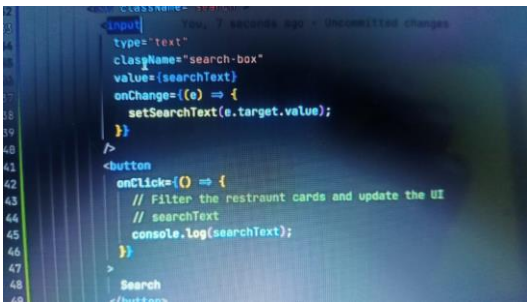
Okay now we are going to build a search text in the code , but there is one issue.

First I will create a search text state variable.

Once I have created the search text no matter what I write in it nothing is showing.

Why?

Actually the value wehave given is binded to the searchtext so its not updating , so we have to add onChange handler in it



Whenever something changes in input form we need to update it in state variable.

Why reatc is faster?? React has virtual dom and with reacts reconsilation mechanism that is react fiber , it differentiates the old V dom to new V dom and only updates the difference.