World of react js app

Monolith and microservice architecture :

Monolith project 🡪 huge large of code and we have UI code,authentication , api code,notification,DB inside the project both frontend and backend is in single application ..if we need to change small button .. we need to build whole application 🡪 this is called monolith project

Microservice architecture :

We have different service for ui , notification , authentication , DB ,api etc and it combine together to form entire app .. separate project for each service and they interact with each other

**This is called separation of concern and single responsibility principle**

All the developers works in same repo .. backend is separate and frontend is separate

How the services interact with each other?

The frontend need to talk to backend and backend need to talk to db and notification etc

**How the services are deployed and how they are managed?**

In microservice we can have any language in separate microservices eg:python , .net etc

It is not necessary to have entire application with react alone

**All the services run on their own specific port eg: ui service in 1234 and backend in 2344**

**All the ports are mapped to main url**

Eg: we need ui service as /frontend and backend as /backend which is mapped to main url

How these service are interacted?

1.They are interacted through service urls

**How to get data from dynamic api?**

**There are 2 ways that are followed**

1.as soon as page load we can call api and render it to UI .. the ui will load after the api call is completed

**2.The UI is rendered and then after the response is got we will again re-render the application with data 🡪 ui rendered first then api called**

In react we will use 2nd … it is better UI or UX.. we can see skelton alone initially and then we display data

We are rendering 2 times that’s ok ..because react render UI very very fast

**useEffect?**

**useEffect(()=>{},[])**

callback function and dependency array as parameters

1.when the callback function is called?? ..> **The callback function is called after the component is completed rendered**

**2.if we need to do something after the body is loaded**

 // it is similar to settimeout which is asycn i.e ueseEffect is called after component ui rendered and commit phase is over in react

  useEffect(() => {

    console.log("call useeffect");

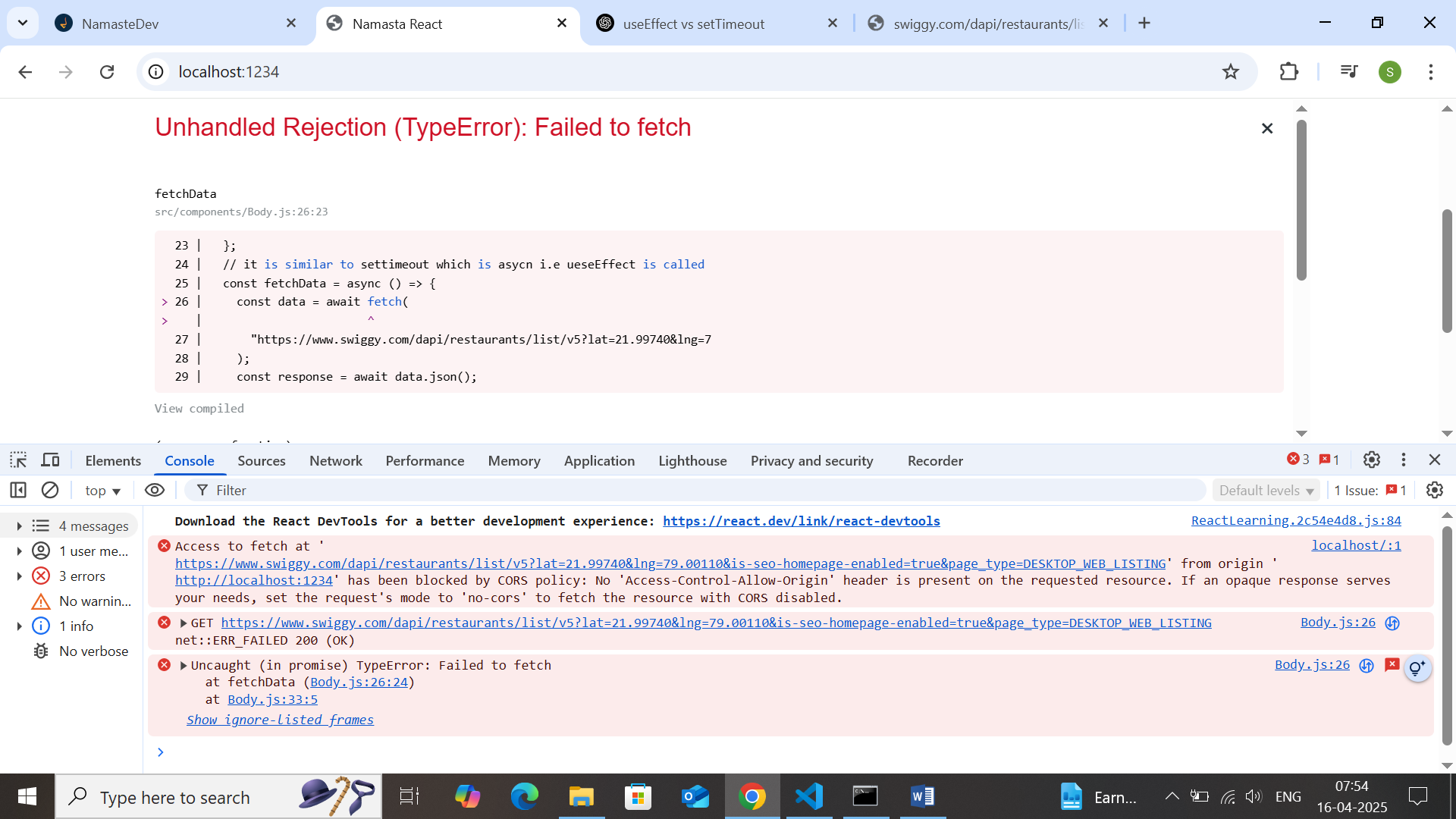
  }, []);

**The second approach is used here where ui rendered first and data from api fetched later**

Fetch method for getting data

**Fetch() is given by web apis in js engine that is used to get data from api**

When we try to access the swiggy api eg: <https://www.swiggy.com/dapi/restaurants/list/v5?lat=21.99740&lng=79.00110&is-seo-homepage-enabled=true&page_type=DESKTOP_WEB_LISTING> from react it is throwing error

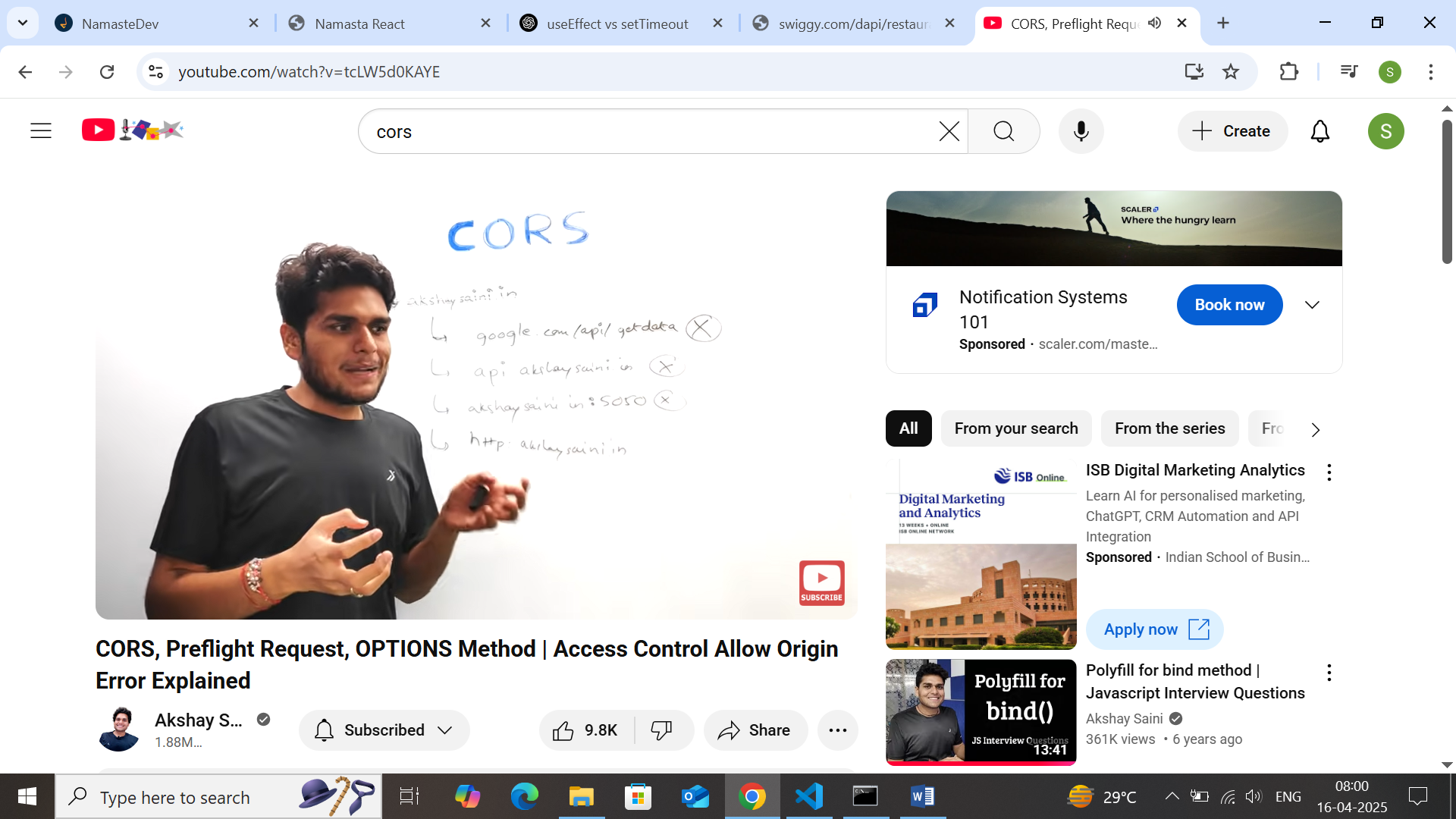


What does cross origin error means?

The browser is blocking us to call one origin to another origin .. this is called cors policy

**What is cors?**

* It is mechanism which uses additional http headers to tell browser whether specific web app can share the resource with other web app .. if the web apps have 2 different url eg: swiggy and local ,, it has to follow cors mechanism
* Eg: https:swiggy need to get data from swiggy to http:swiggy etc



If we have 2 apps on different domines how they share resources?

* A cross pre-flight call options call is made before api call is made
* Preflight call ..which is called options called .. and if the call is valid the second origin set some additional http request and then the next api call is called
* 
* The additional headers from origin B contains some headers as follows
  + [Accept-control-allow-origin:\*] .. this means the domain B is fulfilled i.e we can access any data or post it from any domain
  + [Accept-control-allow-origin:swiggy,Sangeetha.com] etc
  + Not everytime the pre-flight is called in all apis
  + Best way is to following cors mechanism to solve cross origin issue

**Why Does CORS Exist?**

It’s **not** a bug — it’s a **security feature** to prevent:

* Malicious sites from reading sensitive info from other domains
* CSRF-like attacks by limiting cross-origin access

Awesome question! CORS errors are super common when working with APIs, especially in frontend development. Let me break it down for you in a simple, developer-friendly way.

## 🚫 What is a CORS Error?

**CORS** stands for **Cross-Origin Resource Sharing**. It’s a **security feature** implemented by browsers to prevent a web page from making requests to a different domain (origin) than the one it was loaded from — **unless that server says it’s okay.**

### 🤔 Example of a CORS Error:

Let’s say your React app is running on:

http://localhost:3000

And you try to fetch from:

https://api.example.com/data

Boom 💥 — different origins (different protocol + domain + port). If https://api.example.com **doesn't explicitly allow requests from your origin**, the browser blocks the response, and you see:

**Access to fetch at ‘**[**https://api.example.com/data’**](https://api.example.com/data%E2%80%99) **from origin ‘**[**http://localhost:3000’**](http://localhost:3000%E2%80%99/) **has been blocked by CORS policy**

## 🔒 Why Does CORS Exist?

It’s **not** a bug — it’s a **security feature** to prevent:

* Malicious sites from reading sensitive info from other domains
* CSRF-like attacks by limiting cross-origin access

## ⚙️ How CORS Works Behind the Scenes

Let’s say you call fetch('https://api.example.com/data'):

1. **Browser sends a request** with an Origin header:
2. Origin: http://localhost:3000
3. **Server checks** if that origin is allowed. If yes, it responds with:
4. Access-Control-Allow-Origin: http://localhost:3000
5. If the response **contains the right CORS headers**, the browser allows it.
6. If not — ❌ CORS error. The **browser blocks the response**, even though the request reaches the server.

### 🕵️‍♂️ Preflight Requests (OPTIONS)

For some requests (like POST, or requests with custom headers), the browser first sends a **preflight** request:

OPTIONS /data HTTP/1.1

Origin: http://localhost:3000

Access-Control-Request-Method: POST

The server needs to respond with:

Access-Control-Allow-Origin: http://localhost:3000

Access-Control-Allow-Methods: POST

Only then will the actual request be made.

## 🛠️ How to Fix / Bypass CORS Errors

### ✅ **Option 1: Server-side fix (Best)**

Update the API server to include the appropriate CORS headers:

Access-Control-Allow-Origin: \*

Or more securely:

Access-Control-Allow-Origin: http://localhost:3000

In Node.js (Express):

const cors = require('cors');

app.use(cors({ origin: 'http://localhost:3000' }));

### 🧪 **Option 2: Use a Proxy (Dev Only)**

In React (Vite/CRA), set up a proxy in vite.config.js or package.json:

// vite.config.js

server: {

proxy: {

'/api': 'https://api.example.com'

}

}

This avoids the CORS check because the browser thinks the request is coming from the **same origin**.

### 🔀 **Option 3: Use a CORS proxy (Not recommended for prod)**

Example:

fetch('https://cors-anywhere.herokuapp.com/https://api.example.com/data')

This routes your request through a proxy that adds CORS headers. Good for quick tests, not for real apps.

### 🚨 Important:

CORS is enforced **only by browsers**. If you're using tools like Postman or cURL, they won’t block you. It’s purely a browser security policy.

adding loading is not good way why?? .. there is concept of shimmer ui where

  // adding loading is not good way why?? .. there is concept of shimmer ui where we load fake page till we load ui

  if (listRestorent.length === 0) {

    return <p>Loading...</p>;

  }

It is not good ui experience to show loading first and then load the data

Instead of showing loading we can show fake ui how to do it?

Shimmer ui:

## Shimmer UI: A Better Way to Show Loading States

If you have ever used a web or mobile app that takes some time to load data from a server, you might have seen a loading spinner or a progress bar that indicates that something is happening. While these are common ways to show loading states, they are not very engaging or informative for the user. They don’t tell the user what kind of content is being loaded, how long it will take, or what to expect next.

A better way to show loading states is to use a shimmer UI. A shimmer UI is a version of the UI that doesn’t contain actual content, but instead mimics the layout and shapes of the content that will eventually appear. It uses a gradient animation that creates a shimmering effect over the placeholders, giving the user a sense of motion and progress.

Why we need useState?

What is super power in useState that make the data to re-render and make component re-render?

When we use normal variable in jsx the react will not know when there is change in data and it wont update ui with latest data

**To achieve this we use useState.. to manage data in ui and tell react to re-render we use useState**

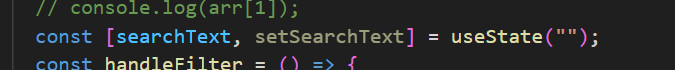
useState gives a state variable and gives the function to set the state variable

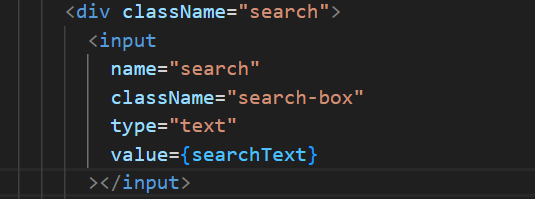
why react need to do it?

1.there Is no way for react to know whether the variable value is changed or not..when we do setBtn function .. react will re-render the component again

2.react trigger re-consollation process again and updates the ui

Is react re-freshing whole application or modifiying the button?

* React will re-render the whole react component again and react will update the variable with latest data
* Re-rendering 🡪it will trigger the component once again
* How const variable is modified in useState??
  + When ever we update the const variable it will re-render the component again which means new const variable comes with last updated value ..
  + React will compare the diff b/t old component and new component and changes only the btn if it is changed not entire component
  + 



When I type the content in input it wont be reflected in input because the input value is binded to searchText but it is not bind to value change so we can do with onChange

When ever state variable update react trigger reconsliation cyle i.e render the react component

How it is happening

The react is finding the diff b/t old virtual DOM and new virtual DOM and if there is input value change the react is re-rendering the entire component but change only the input value in DOM

**People who code slow does not need to debug the code so code slow**

**How useEffect hooks works?**

**1.when** Is useEffect called?

1.it is called everytime the component is rendered … because of dependency array It changes

2.if no dependency array the useEffect calls on each render of component

  useEffect(() => {

    console.log(

      "useEffect without dependency calls on every re-render of component"

    );

  });

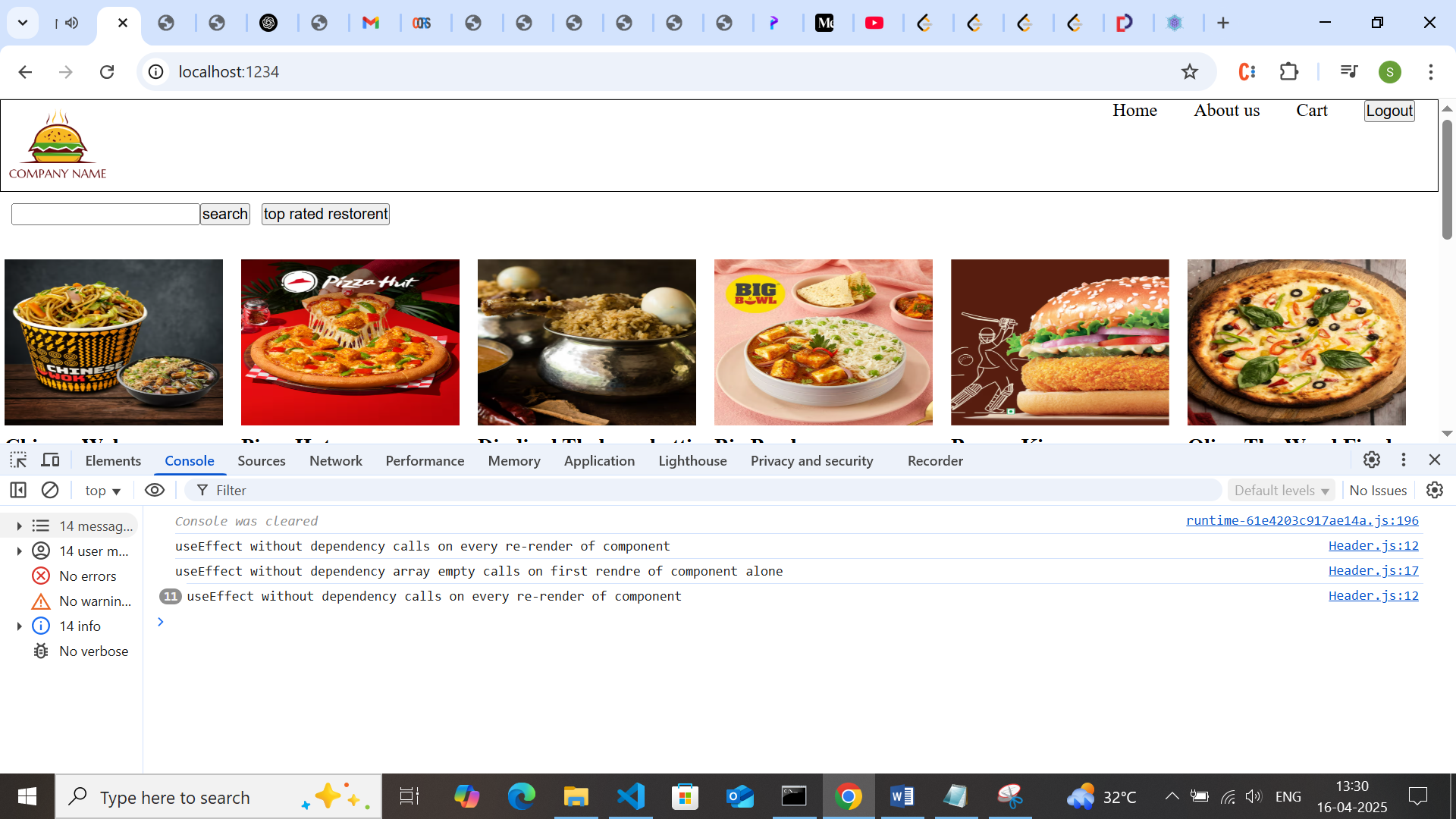
  useEffect(() => {

    console.log(

      "useEffect without dependency array empty calls on first rendre of component alone"

    );

  }, []);



The basic nature of useEffect is to call each render but with empty [] it calls only one time

1.If there is dependency array with some value then useEffet is called everytime when the dependency changes it calls the useEffect

 useEffect(() => {

    console.log(

      "useEffect with depdency array calls when there is change in dependency"

    );

  }, [isLoggedIn]);

**Best practices for using useState and useEffect**

1.should not call useState outside of functional component … local state variable inside functional component

2.should have useState at top of component

3.never create useState inside if else --sometimes the if condition will be there or may not

**4.dont create state variables inside functions and no where else except inside functional component**

5.dont use useState with for conditions