useEffect() ' takes two arguments .

1. Callback function.

2. Dependency Array.

```
// We passed Arrow function as callback function.
useEffect(() => {}, []);
```

When will the callback function get called inside the useEffect()?

Callback function is getting called **after the whole component get rendered**.

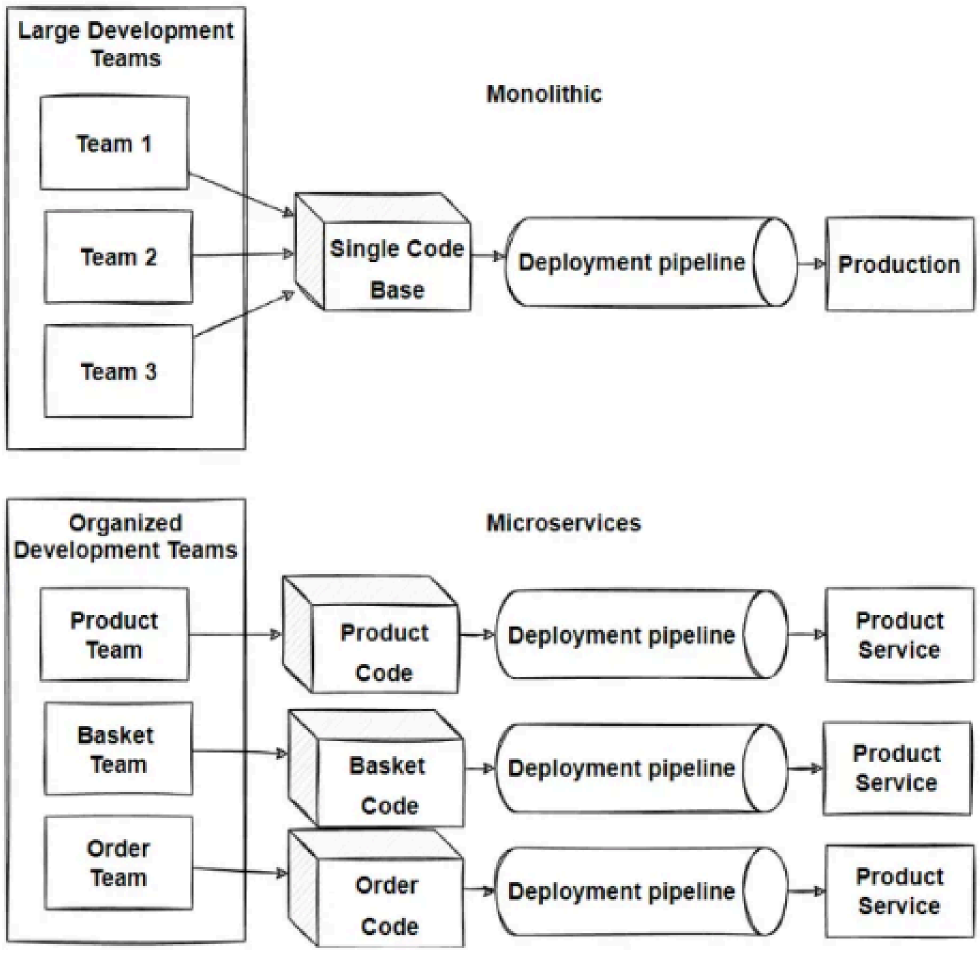
In our app we are using **' useEffect() '** inside Body component. So it will get called once Body component complete its render cycle.

If we have to do something after the rendercycle complets we can pass it inside the ' useEffect() ' . this is the actual use case of useEffect.

Where we fetch the data?

inside the 'useEffect() ' we use ' fetchData() ' function to fetch data from the external world

Monolithic vs. Microservices Architecture

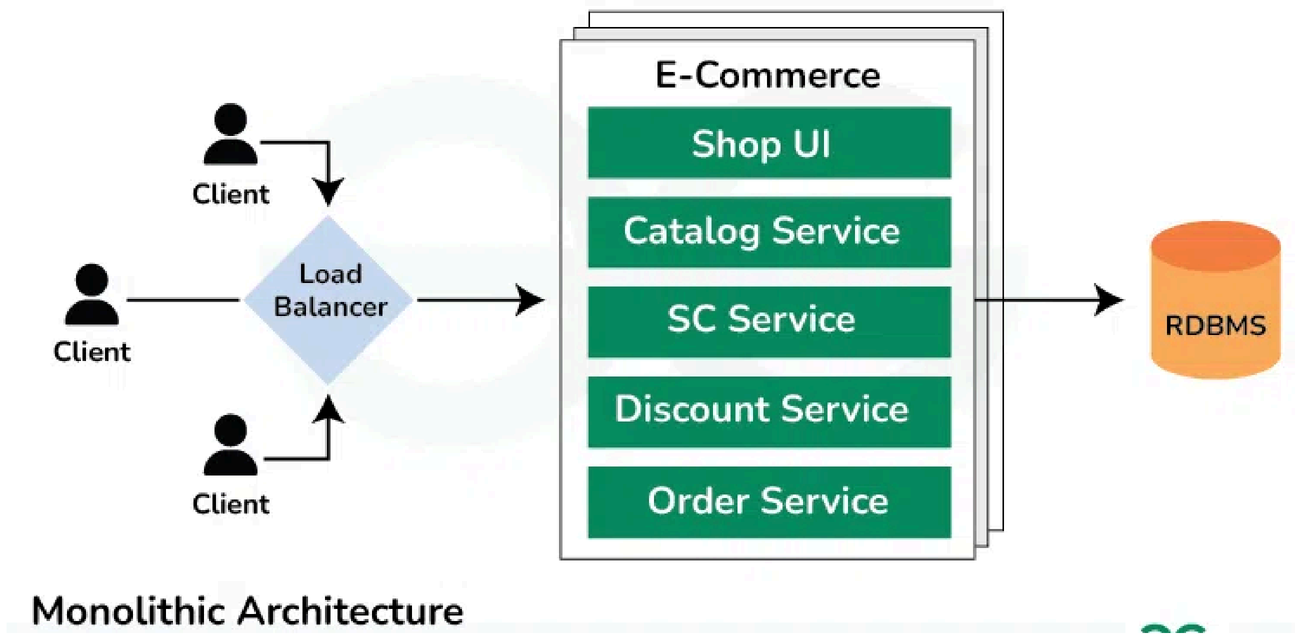


Monolithic Architecture (whole project in one service only)

A monolithic architecture means the entire software is built as one big unit.

Any update requires redeploying the whole application.

It’s simple and easy for small apps, but becomes hard to maintain and scale as the application grows.



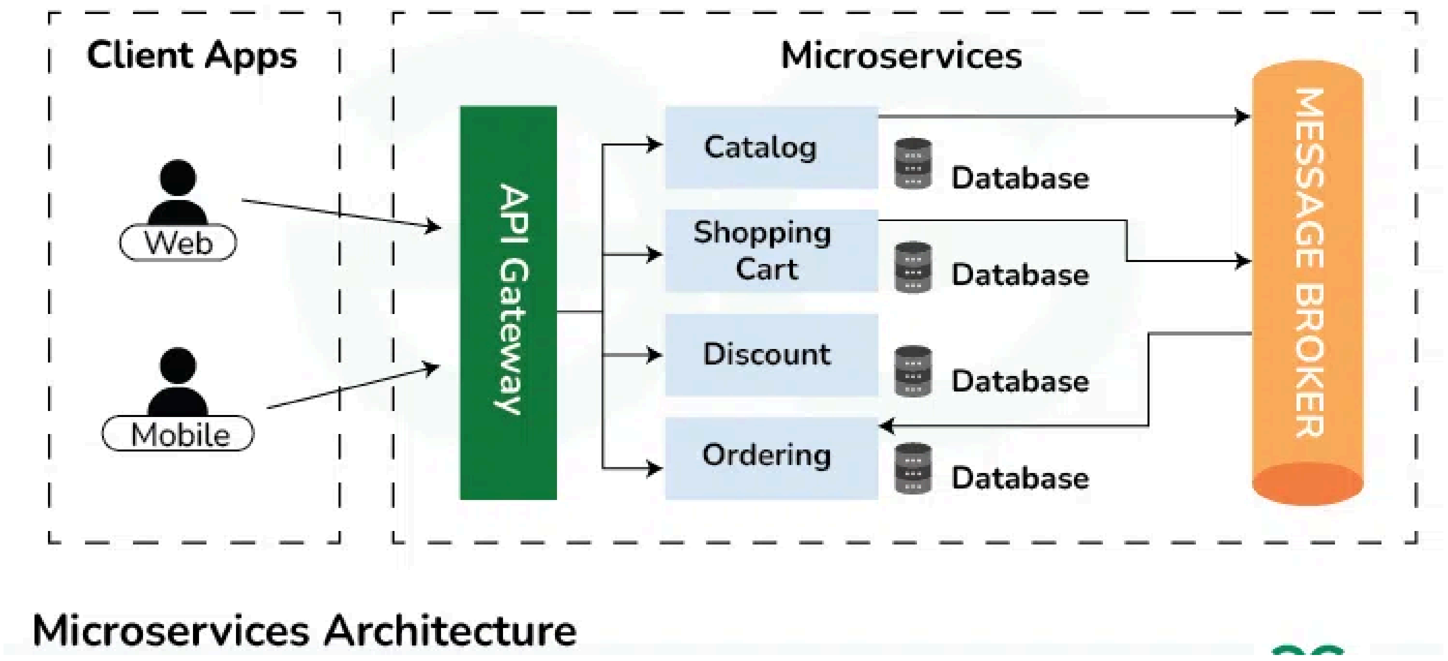
Monolithic Architecture

Microservices architecture (Different services for all)

Microservices architecture breaks an application into many small, independent services.

Each service handles one specific feature, can be built and deployed separately, and communicates with others over the network (like using HTTP or message queues).

Because services are independent, each one often manages its own database, changing how data is stored and accessed.



Microservices Architecture

Single Responsibility Principle

👉 **One service should do only one main job.**

Examples:

- **Catalog Service:** manages products
- **Shopping Cart Service:** manages cart items
- **Discount Service:** handles discount logic
- **Ordering Service:** handles order placement

No service does more than one job.

How webpages fetch data from backend

Load ➔ **API** ➔ **Render**

(webpage) (UI)

OR

Load ➜ **Render** ➜ **API** ➜ **Re-render**

(webpage) (UI) (UI)

second one gives Better UX

For this We can use ------------------ **UseEFFECT**

If you want to do something After render call in UseEffect

The fetch() function in JavaScript returns a Promise.

What is Shimmer UI

Shimmer UI is a **loading skeleton effect** that shows gray animated shapes while actual data is loading.

It gives the user a feeling that:

➡️ Content is coming
➡️ Page is not stuck
➡️ Better UI/UX than a normal spinner

Where you see it?

- Facebook feed loading
- Instagram loading posts

- YouTube video list before thumbnails show
- Swiggy / Zomato loading restaurant cards

Optional Chaining (?.)

Optional chaining is a safe way to access nested object properties without throwing an error if something is missing

`user?.address?.city`

🎯 When to use

- Accessing **API response**
- Accessing **deep nested objects**
- Accessing **props in React**
- Anytime you are *not sure* value exists
-

❗ When NOT to use

- When you know value always exists
- Inside loops where undefined breaks logic

CORS

CORS (Cross-Origin Resource Sharing) is a browser security feature that controls which frontend (origin) is allowed to access a backend API on a different domain.

Why does CORS exist?

Origin = protocol + domain + port

Example:
http://localhost:3000 → frontend
http://localhost:5000 → backend

Browsers block requests between different origins unless the server allows it using CORS headers.

⭐ Three Steps of CORS

Simple Request

- Methods: GET, POST, HEAD
- No custom headers
- Browser sends request directly
- Server must reply with:
  Access-Control-Allow-Origin

Preflight Request (OPTIONS)

- Happens when request is not simple
  (PUT/DELETE, custom headers, tokens, cookies)
- Browser sends OPTIONS first asking permission
- Server replies with allowed methods/headers
  (Access-Control-Allow-Methods,
  Access-Control-Allow-Headers,
  Access-Control-Allow-Origin)

Actual Request

- Sent only if preflight success

- **Browser allows response only if CORS headers match**