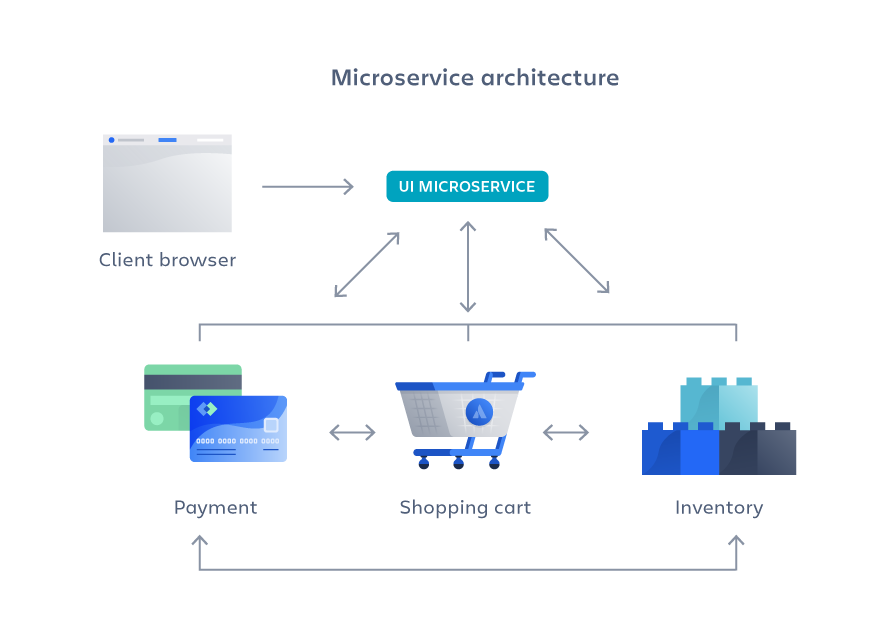
1. **What is Microservice?**

**Ans:** Microservice is a software development architecture pattern where each part of the software is developed and deployed individually as a service. They interact with each other which collectively works as software. Ex: UI is a separate service, Backend is a different service, Payment is an extra service, and Authentication is an additional service so these collectively communicate with each other.



1. **What is Monolith architecture?**

**Ans:** Monolith architecture is a software development architecture pattern where every part of the software is developed in one project and deployed as software. Ex: The project includes the frontend part, backend part, ML part, and payment part so they communicate with each other internally.



1. **What is the difference between microservice and monolith architecture?**

**Ans:** **Monolith Advantages & disadvantages:**

**Advantages:**

* Easy deployment – One executable file or directory makes deployment easier.
* Development – When an application is built with one code base, it is easier to develop.
* Performance – In a centralized code base and repository, one API can often perform the same function that numerous APIs perform with microservices.
* Simplified testing – Since a monolithic application is a single, centralized unit, end-to-end testing can be performed faster than with a distributed application.

**Disadvantages**

* Slower development speed – A large, monolithic application makes development more complex and slower.
* Scalability – You can’t scale individual components.
* Reliability – If there’s an error in any module, it could affect the entire application’s availability.
* The barrier to technology adoption – Any changes in the framework or language affects the entire application, making changes often expensive and time-consuming.
* Lack of flexibility – A monolith is constrained by the technologies already used in the monolith.
* Deployment – A small change to a monolithic application requires the redeployment of the entire monolith.

**Microservice Advantages and Disadvantages:**

**Advantages**

* Agility – Promote agile ways of working with small teams that deploy frequently.
* Flexible scaling – If a microservice reaches its load capacity, new instances of that service can rapidly be deployed to the accompanying cluster to help relieve pressure. We are now multi-tenant and stateless with customers spread across multiple instances. Now we can support much larger instance sizes.
* Continuous deployment – We now have frequent and faster release cycles. Before we would push out updates once a week and now we can do so about two to three times a day.
* Highly maintainable and testable – Teams can experiment with new features and roll back if something doesn’t work. This makes it easier to update code and accelerates time-to-market for new features. Plus, it is easy to isolate and fix faults and bugs in individual services.
* Independently deployable – Since microservices are individual units they allow for fast and easy independent deployment of individual features.
* Technology flexibility – Microservice architectures allow teams the freedom to select the tools they desire.

**Disadvantages**

* Development sprawl – Microservices add more complexity compared to a monolith architecture since there are more services in more places created by multiple teams. If development sprawl isn’t properly managed, it results in slower development speed and poor operational performance.
* Exponential infrastructure costs – Each new microservice can have its own cost for the test suite, deployment playbooks, hosting infrastructure, monitoring tools, and more.
* Added organizational overhead – Teams need to add another level of communication and collaboration to coordinate updates and interfaces.
* Debugging challenges – Each microservice has its own set of logs, which makes debugging more complicated. Plus, a single business process can run across multiple machines, further complicating debugging.

1. **Why do we need useEffect hook in react?**

**Ans:** React components have their lifecycle stages like mounting, unmounting, and state resetting so to handle all these functional components react introduced the use effect hook which is mainly used to execute side effects at certain times of the component life cycle. generally during the network request use effect will be used to set the state whenever we get the data from the server, when we want to execute some side effect like calculating the total product price of all products in the cart we can use the use effect with dependency array set with the state of the products and when we added the setTimeout,setInterval to build the timer we can remove these timers when the component unmounts.

1. **What is Optional Chaining?**

**Ans:** In javascript, we can use optional chaining to handle the error when some variables don’t include what we are accessing to expect. using we can prevent unexpected errors.

1. **What is shimmer UI?**

**Ans:** In web development fetching data is very common in older times loading spinners are used to handle the loading scenario of the data so that the user is indicated that the data is coming instead of showing a blank screen. But as web development improves the handing of the loading state is shifted to shimmer UI which means showing the UI with the same width and height as the component which showed after the data is populated With the background color of light grey animation. Which improves the user experience such the actual UI is appearing.

1. **What is a js expression and js statement?**

**Ans:** Js Expression is a unit of code that resolves to a value whereas the Js statement is the unit of code that resolves to action.

Ex: // Statements

let x = 0;

function add(a, b) { return a + b; }

if (true) { console.log('Hi'); }

// Expressions

x; // Resolves to 0

3 + x; // Resolves to 3

add(1, 2); // Resolves to 3

1. **What is conditional rendering?**

**Ans:** Conditional rendering based on the value of some variable we are rendering different components. Suppose we are fetching the data of the user and we are managing the loading state we want to show the shimmer divs if loading is true and if loading we show the actual div with the fetched data this can be achieved using conditional rendering. Using? : statement.

Ex: loading?<Shimmerdiv/>:<ProductContainer products={products}/>

1. **What is Cors?**

**Ans:** Cros is cross-origin resource sharing in the modern web the website need to fetch data from other origins the cors mechanism indicates it is allowed or it all happens in the preflight request where the server validates the request is coming from the same origin or not if it is not then it will throw an error because generally, browser prevent the cross-origin request to handle this the server should have access-control-allow-origin header with the conditions, origins which allowed to request.

1. **What is async and await?**

**Ans:** In javascript, asynchronous programming can be handled using promises and async and await both serve the kind of same work. A function can be turned into an async function in that function we can handle the asynchronous code. using await to wait for the async code to finish it will wait until it executes that code.

Ex:



1. **What is the use of `const json=await data.json()` in getRestaurants?**

**Ans:** we used wait because data.json returns the promise so to handle that we used await and we used .json because we need to convert the response to a JSON object so that it will be easier to handle.