Assignment 6 - Exploring the world

- **What is Microservice?**
- Microservice is a service which is created for doing a specific functionality. In Microservice architecture there are different microservices which are created for providing a particular functionality. The microservices are loosely connected with each other and can interact with other microservices. Each microservice can have its own tech stack.

- **What is Monolith Architecture?**
- Monolith architecture is the traditional way of creating projects where all the services are written in the same project. The project contains logic for database connection, Auth, UI, backend, etc. For running the monolith architecture project, we need to compile and build the whole code which is not a good approach.

- **What is Difference between Monolith and Microservice architecture?**
- Here are the differences
    - In Monolith architecture, all the modules for different services are written in the same project whereas Microservice Architecture has different microservices for different service/functionality.
    - Monolith Architecture uses a single technology stack for the whole project, whereas Microservice architecture can have different tech stacks for different services.
    - In Monolith Architecture the components are tightly coupled whereas in Microservice architecture the services are loosely coupled.
    - For deploying the Monolith Architecture projects, we need to deploy the whole project as once which makes it harder. For Microservice architecture each service can be deployed separately.

- **Why do we need a useEffect hook?**
- In functional components, after rendering the Shimmer UI or initial UI for the user, we need the useEffect hook to call APIs or call some function where some of the variable value changes.
- To make API calls and to run functions which are required to execute when some of the dependency changes, we need to use the useEffect hook to re render the component based on the value of some variables.
- So to keep the UI updated with the different values present in the component, we need to use the useEffect hook for re rendering the component and keep the Component updated.

- **What is optional chaining?**
- Optional chaining is the way of checking if a particular property or a function exists for a variable. We can use “?” before the “.” to check if the value exists or not. **Optional chaining is a feature in JavaScript that allows you to safely access nested properties of an object without causing an error if any part of the chain is null or undefined.** It uses the ?. syntax to check each property along the chain before attempting to access it. If any part of the chain is not valid, the expression returns undefined instead of throwing an error.

- **What is Shimmer UI?**
- Shimmer UI is the default UI or a placeholder UI which the users sees while the data is getting being loaded/fetched. Instead of a traditional loading spinner or black screen, Shimmer UI is used to enhance the user experience by showing some placeholder UI so the user thinks that the app is loading faster.

- **What is difference between JS expression and JS Statement ?**

  JS Expression:
  - JS expression is an operation which returns a value.It can be a function call, operator combination, etc.

```
2 + 3              // Arithmetic expression
x * 5           // Variable and arithmetic expression
Math.sqrt(9)       // Function call expression
"Hello, " + name  // String concatenation expression
```

  JS Statement:
  - JS statement is a complete instruction in javascript that performs a specific task or action. Statements are used to control the flow of a program, manipulate data and define behavior. They don’t necessarily produce a value but execute an operation.

```
let x = 10;               // Variable declaration and
assignment statement
if (x > 5) {               // Conditional statement
    console.log("x is greater than 5");
} else {
    console.log("x is not greater than 5");
}

for (let i = 0; i < 5; i++) {  // Loop statement
    console.log(i);
}

function greet(name) {
    console.log("Hello, " + name);
}
```

- **What is conditional rendering, explain with a code example ?**
- Conditional rendering is the process of rendering a particular Component or a bunch of React element depending upon some conditions. We can use conditional statements like if…else or use ? with return to conditionally return a particular JSX based on a particular condition.

  Example:

```
if(loading){
      Return <Loading />;
}

return (
      <div>
            <h1> actual content </h1>
      </div>
)
```

OR

```
Return loading ? <loading /> :
      (<div>
            <h1> actual content </h1>
                  </div>);
```

- **What is CORS ?**
- CORS (Cross-Origin Resource Sharing) is a security feature in web browsers that controls and restricts web pages from making requests to domains other than the one that served the web page. It works through HTTP headers:

  1. **Same-Origin Policy:** Browsers normally block requests to different domains for security.

  2. **CORS Headers:** Servers can include specific headers in their responses to allow or deny cross-origin requests.

  - `Access-Control-Allow-Origin`: Specifies which origins are allowed.
  - `Access-Control-Allow-Methods`: Lists allowed HTTP methods.
  - `Access-Control-Allow-Headers`: Lists allowed headers.
  - `Access-Control-Allow-Credentials`: Determines if credentials (e.g., cookies) can be sent.
  - `Access-Control-Expose-Headers`: Lists headers that can be exposed.

3. **Browser Enforcement:** The browser checks these headers, allowing or blocking requests based on the server's settings.

In a nutshell, CORS is a security measure that enables safe cross-origin data sharing for web applications while protecting against potential threats.

- **What is `async and await`?**
- Async and await are the keyword used in javascript for performing asynchronous tasks. We can make any function asynchronous by using async keyword and at any point of executing, we want to wait for some asynchronous operation to be over and wait for the program to stay at the point before the async operation finishes, then we can use await keyword. Await will wait for the asynchronous operation to be over in the async function.

```
Const fetchData = async () => {
      Const response = await fetch(API);
      Const jsonResponse = await response.json();
      Return jsonResponse
}
```

- **What is the use of `const json = await data.json()` in getRestaurants()?**
- We use `.json()` on the response from a `fetch()` request to convert the JSON data in the response body into a JavaScript object. This is necessary because the response from the server is typically in text or binary format, and JavaScript needs data in object format to work with it effectively. `.json()` simplifies the process of parsing JSON data and makes it easier to use in your JavaScript code.