**Assignment 7 – Finding the path**

Q1: What are various ways to add images into our App? Explain with code examples?

React with its component-based architecture, offers a flexible way to handle assets, allowing our applications to be visually appealing and performance optimized.

1. Importing and using an image in a React Component

Most straightforward way. The file should be located within the project directory.

import errorImg from "../images/404-error.jpg";

const Error = () => {

   return (

    <div className="error-page">

      <img src={errorImg} alt="Food Image" />

    </div>

  );

};

1. Importing an SVG image

Importing an SVG(Scalable Vector Graphics) image in the application is slightly different from importing a regular image file. SVG images are XML-based vector images that can be scaled without losing quality making them suitable for web applications.

import { ReactComponent as MyLogo } from './my\_logo.svg';

const MyComponent = () => {

return (

<div>

<MyLogo />

</div>

);

};

1. Importing an image from a public folder

You can use an image from the public folder without using the import statement. This is useful for handling large image assets or for dynamic image URLs.

The image must be in the public folder.

<img src={process.env.PUBLIC\_URL + '/image.jpg'} alt="My Image" />

1. Rendering an externally hosted image

const MyComponent = () => {

const imgUrl = = 'https://example.com/my\_image.jpg';

return (

<div>

<img src={imgUrl} alt="My Image" />

</div>

);

};

Q2: What would happen if we do console.log(useState())?

An array containing 2 items is returned. The first item is a state variable with value undefined and the 2nd item is a setter function to set value to that state variable.

Console.log(useState()) – o/p : [undefined, Function]

Q3: How will useEffect behave if we don't add a dependency array?

Dependency Array: The role of dependency array is to control when the effect should run.

Reactive values include props and all variables and functions declared directly inside of your component.

1. If you specify the dependencies, your Effect runs **after the initial render** *and* **also re-renders with changed dependencies.**

useEffect(() => {

// ...

}, [a, b]); // Runs again if a or b are different.

1. An Effect with empty dependencies doesn’t re-run when any of your component’s props or state change.

If your Effect truly doesn’t use any reactive values, it will only run after the initial render.

useEffect(() => {

// ...

}, []); // Does not run again (except once in development)

1. If you pass no dependency array at all, your Effect runs after every single render (and re-render) of your component.

useEffect(() => {

// ...

}); // Always runs again and again.

Q4: What is SPA?

According to MDN, an SPA(Single Page Application) is a web-app implementation that loads only a single web document, and then updates the body content of that same single document dynamically via Javascript fetch API.

This enables the users to use websites without loading new pages from the server. There is only one page that is updated with data thereby eliminating the need for re-loading the entire page.

Q5: What is difference between Client-Side Routing and Server-Side Routing?

There are 2 types of routing in any web application: Client-side routing (CSR) and Server-side routing (SSR).

1. Execution Logic – In CSR, the routing logic is executed within the browser on the client side. In SSR, the routing logic is on the server. The server determines how to respond to the request URL and sends the response accordingly.
2. Page Rendering – In CSR, this happens typically within the browser. When the user navigates to a different route, the client side JS framework (React)fetches the data and updates the DOM to render a new view without reloading the entire page.

In SSR, this happens at the server. Each time the user navigates to a new route, the server generates the HTML template response and sends it to the client thereby reloading the entire page.

1. Initial Page load- In CSR, Initial page is minimal (Shimmer UI) containing only the basic structure. It is only after this that the data is fetched and rendered.

In SSR, the initial HTML page contains the entire content of the app.

1. Performance – CSR, provides faster and smoother user experience after the initial load.

SSR, may result in slower navigations between routes due to the round trips between client and server.