Assignment 8

Owner - Mohit R

1. **How do you create Nested Routes react-router-dom configuration?**

```
const appRouter = createBrowserRouter([
{
        path: "/",
        element: <AppLayout />,
        children: [
        {
                path: "/",
                element: <Body />,
        },
        {
                path: "/about",
                element: <About />,
        },
        {
                path: "/contact",
                element: <Contact />,
        },
        {
                path: "/resturants/:resId",
                element: <ResturantMenu />,
        },
        ],
        errorElement: <Error />,
},
]);
```

2. **Read about createHashRouter, createMemoryRouter from React Router docs.**

- **createHashRouter**
    - This router is useful if you are unable to configure your web server to direct all traffic to your React Router application.
    - Instead of using normal URLs, it will use the hash (#) portion of the URL to manage the "application URL".
- **createMemoryRouter**
    - Instead of using the browser's history, a memory router manages its own history stack in memory.
    - It's primarily useful for testing and component development tools like Storybook but can also be used for running React Router in any non-browser environment.

3. **What is the order of life cycle method calls in Class Based Components**

- Class based components are executed in two phases: Render phase & commit phase.

- The render phase is pure and has no side effects. It may be paused, restarted or aborted by React (when child component is created for eg).
- The constructor(), render() and componentDidMount() happens in this phase.
- In constructor, the props are passed to its parents.
- These methods are called in the following order when an instance of a component is being created and inserted into the DOM:
- Mounting:
    - constructor -
        - The constructor for a React component is called before it is mounted.
        - When implementing the constructor for a React.Component subclass, you should call super(props) before any other statement.
        - Otherwise, this.props will be undefined in the constructor, which can lead to bugs.
        - Initializing local state by assigning an object to this.state
        - Binding event handler methods to an instance.
        - Constructor is the only place where you should assign this.state directly. In all other methods, you need to use this.setState() instead.
        - componentDidMount() - componentDidMount() is invoked immediately after a component is mounted (inserted into the tree). You may call setState() immediately in componentDidMount() so that it triggers re-render before the browser updates the screen.
- Updating : 3. componentDidUpdate() - componentDidUpdate() is invoked immediately after updating occurs. This method is not called for the initial render.
- Unmounting : 4. componentWillUnmount() -componentWillUnmount() is invoked immediately before a component is unmounted and destroyed. Perform any necessary cleanup in this method, such as invalidating timers, canceling network requests, or cleaning up any subscriptions that were created in componentDidMount().

4. Why do we use componentDidMount?

- If you need to load data from a remote endpoint (Calling an API), this is a good place to instantiate the network request.
- This method is a good place to set up any subscriptions.
- You may call setState() immediately in componentDidMount(). It will trigger an extra rendering, but it will happen before the browser updates the screen.

5. Why do we use componentWillUnmount? Show with example

- componentWillUnmount is used to cleanup any function/subscriptions that will be running even after the component is unmounted.
- For example, in Repo class, during componentDidMount() a timer is set with an interval of every one second to print in console. When the component is unmounted (users moves to a different page), the timer will be running in the background, which we might not even realise and causing huge performance issue. To avoid such situations the cleanup function can be done in componentWillUnmount, in this example clearInterval(timer) to clear the timer interval before unmounting Repo component.

```
componentDidMount() {
        const timer = setInterval(() => {
        },1000);
}

ComponentWillUnmount() {
        clearInterval(timer);
}
```

6. **Why do we use super(props) in constructor?**

- super() is used inside constructor of a class to derive the parent's all properties inside the class that extended it.
- If super() is not used, then Reference Error : Must call super constructor in derived classes before accessing 'this' or returning from derived constructor is thrown in the console.
- A component that extends React.Component must call the super() constructor in the derived class since it’s required to access this context inside the derived class constructor.
- When you try to use props passed on parent to child component in child component using this.props.name, it will still work without super(props).
- Only super() is also enought for accessing props in render method.
- The main difference between super() and super(props) is the this.props is undefined in child's constructor in super() but this.props contains the passed props if super(props) is used.

7. **Why can't we have the callback function of useEffect async?**

- useEffect expects it's callback function to return nothing or return a function (cleanup function that is called when the component is unmounted).
- If we make the callback function as async, it will return a promise which is not expected. Solution to this is not making the callback function async but create another async function inside callback function of useEffect()