**Assignment 8 – Let’s get Classy**

Q1: How do you create Nested Routes react-router-dom configuration?

Q2: Read abt createHashRouter, createMemoryRouter from React Router docs. Q3: What is the order of life cycle method calls in Class Based Components?

Q4: Why do we use componentDidMount?

Q5: Why do we use componentWillUnmount? Show with example.

Q6: (Research) Why do we use super(props) in constructor?

Q7: (Research) Why can't we have the callback function of useEffect async?

In React, useEffect is used to handle side-effects in functional components. Can perform tasks such as external API call, timers etc.

UseEffect cannot have an async callback function directly. This is because useEffect expects a cleanup function to be returned synchronously. And making the callback function as async means it would return a promise thus violating the expected behaviour.

There are a couple of workarounds:

1. Using an IIFE :

const [books, setBooks] = useState([]);

useEffect(() => {

(async () => {

try {

const books = await fetchBooks();

setBooks(books);

} catch (err) {

console.log('Error occured when fetching books');

}

})(); //IIFE is also known as Self Executing Anonymous Function

return ()=>{

//optional – clean up code.Disconnecting from DB, clearing timers etc.This function is called when the component unmounts or when depencies change

}

}, []);

1. Using a named async function inside useEffect’s callback

const [books, setBooks] = useState([]);

useEffect(() => {

async function getBooks(){

try {

const books = await fetchBooks();

setBooks(books);

} catch (err) {

console.log('Error occured when fetching books');

}

}

getBooks();//Calling the named function

}, []);