**REACT DAY-4 ASSIGNMENT QUESTIONS (THEORY)**

**By- Shruti Gahlaut**

**Q-1 How to send security token in axios get request?**

Most APIs are not accessible without first authenticating. This helps to provide a safe and secure environment for the API users. After authentication, the user can safely make calls to the API. But how to authorize calls after authentication? One way of achieving it involves sending a bearer token with the client’s request to the API.

To send a bearer token with an axios request, pass an *Authorization* header to the request.

Here is the header you need to add to your request:

javascript{ Authorization: `Bearer ${token}` }

Here is an example of how to do this:

javascriptimport axios from 'axios';

const token = '{TOKEN}';

axios.request({

headers: {

Authorization: `Bearer ${token}`

},

method: "GET",

url: `https://jsonplaceholder.typicode.com/posts`

}).then(response => {

console.log(response.data);

});

As you can see, in this example we pass the bearer token to an axios GET request.

## Passing the bearer token to every request

To pass the bearer token to every axios request, add a default header option to the axios global object.

Here is how you do it:

javascriptimport axios from 'axios';

const token = '{TOKEN}';

axios.defaults.headers.common = {

'Authorization': `Bearer ${token}`

};

All the axios requests will now have the *Authorization* bearer header.

We can also pass the authorization token in an axios interceptor.

javascriptimport axios from 'axios';

const axiosInstance = axios.create({

baseURL: process.env.REACT\_APP\_BASE\_URL,

});

axiosInstance.interceptors.request.use(

(config) => {

const token = '{TOKEN}'

const auth = token ? `Bearer ${token}` : '';

config.headers.common['Authorization'] = auth;

return config;

},

(error) => Promise.reject(error),

);

In this case, only requests made with *axiosInstance* will have the *Authorization* bearer header.

**Q-2 How to create a class component in React JS?**

**React Components**

Components are independent and reusable bits of code. They serve the same purpose as JavaScript functions, but work in isolation and return HTML via a render() function.

Components come in two types, Class components and Function components.

**Creating a Class Component**

When creating a React component, the component's name must start with an upper-case letter.

The component must include the extends React.Component statement, this statement creates an inheritance to React.Component, and gives the class component access to React.Component's functions.

The component also requires a render() method and this method returns HTML.

**For Example -**

Create a Class component called Car

class Car extends React.Component {

render() {

return <h2>Hi, I am a Car!</h2>;

}

}

Now, the React application has a component called Car, which returns a <h2> element.

To use this component in the application, use similar syntax as normal HTML: <Car />

Example

Display the Car component in the "root" element:

const root = ReactDOM.createRoot(document.getElementById('root'));

root.render(<Car />);