- **Write the Unit Test cases by using Jest for your Front-End [React].**

Jest

Login Page
Code

```
// LoginComponent.test.js
import React from 'react';
import { render, fireEvent } from '@testing-library/react';
import axios from 'axios';
import { screen } from '@testing-library/react';
import { BrowserRouter, MemoryRouter, Routes, useNavigate, Route,Router }
from 'react-router-dom';
import  LoginComponent  from '../Component/LoginComponent';

// mock axios and useNavigate
jest.mock('axios');
jest.mock('react-router-dom', () => ({ useNavigate: jest.fn() }));



test('RENDER THE LOGIN FORM CHECK LABEL', async () => {
    const { getByLabelText } =
        render(<LoginComponent />);
    expect(getByLabelText('Enter a email')).toBeInTheDocument();
    expect(getByLabelText('Enter a password')).toBeInTheDocument();
});
test('RENDER THE LOGIN FORM CHECK BUTTON EXISTS', async () => {
    const { getByText } =
        render(<LoginComponent />);
   expect(getByText('Login')).toBeInTheDocument();
    expect(getByText('Register')).toBeInTheDocument();


});

  test('shows error messages if email or password is empty', () => {
    // render the component
    const { getByText ,getByTestId} = render(<LoginComponent/>);
    // get the elements
```

```
    const loginButton = getByText('Login');
    // click the login button
    fireEvent.click(loginButton);
    // check if the error messages are shown
    expect(getByTestId('useremailerror')).toHaveTextContent('Please enter
a email');
    expect(getByTestId('userpwderror')).toHaveTextContent('Please enter a
password');
  });


  test('calls post axios if the login is successful', async () => {
    // render the component
    const { getByText,getByRole } = render(<LoginComponent/>);
    // get the elements
    const emailInput = getByRole('logininput1', { id: 'UserEmail' });
    const passwordInput = getByRole('logininput2', { id: 'UserPassword'
});
    const loginButton = getByText('Login');
    // enter some valid data
    fireEvent.change(emailInput, { target: { value: 'Karthick@gmail.com' }
});
    fireEvent.change(passwordInput, { target: { value: 'Karthick123' } });
    // mock the axios.post response
    axios.post.mockResolvedValue({ data: { account: true, emailstatus:
true, passwordstatus: true } });
    // mock the useNavigate function
    const navigate = useNavigate();
    // click the login button
    fireEvent.click(loginButton);
    // wait for the axios.post call
    await expect(axios.post).toHaveBeenCalled();

  });
```

Register Page Code

```
import React from 'react'
import axios from 'axios';
import { render, fireEvent } from '@testing-library/react';
import { useNavigate } from 'react-router-dom';
import RegisterComponent from '../Component/RegisterComponent';

jest.mock('axios');
jest.mock('../Style/CreateUpdateComponent.css');
jest.mock('react-router-dom');

test('RENDER THE REGISTER FORM CHECK LABEL', async () => {
  const { getByLabelText } =
      render(<RegisterComponent />);
  expect(getByLabelText('Enter a email')).toBeInTheDocument();
  expect(getByLabelText('Enter a name')).toBeInTheDocument();
  expect(getByLabelText('Enter a password')).toBeInTheDocument();
});
test('RENDER THE REGISTER FORM CHECK BUTTON EXISTS', async () => {
  const { getByText } =
      render(<RegisterComponent />);
 expect(getByText('Login')).toBeInTheDocument();
  expect(getByText('Register')).toBeInTheDocument();


});




  test('shows error messages if email or password is empty', () => {
    // render the component
    const { getByText ,getByTestId} = render(<RegisterComponent/>);
    // get the elements

    const RegisterButton = getByText('Register');
    // click the login button

    fireEvent.click(RegisterButton);
    // check if the error messages are shown
```

```
    expect(getByTestId('useremailerror')).toHaveTextContent('Please enter
a email');
    expect(getByTestId('userpwderror')).toHaveTextContent('Please enter a
password');
    expect(getByTestId('usernameerror')).toHaveTextContent('Please enter a
username');
  });

  test('calls post axios if the login is successful', async () => {
    // render the component
    const { getByText,getByRole } = render(<RegisterComponent/>);
    // get the elements
    const nameInput = getByRole('registerinput1', { id: 'UserName' });
    const emailInput = getByRole('registerinput2', { id: 'UserEmail' });
    const passwordInput = getByRole('registerinput3', { id: 'UserPassword'
});
    const registerButton = getByText('Register');
    // enter some valid data
    fireEvent.change(emailInput, { target: { value: 'Karthick@gmail.com' }
});
    fireEvent.change(passwordInput, { target: { value: 'Karthick123' } });
    fireEvent.change(nameInput, { target: { value: 'Karthick' } });
    // mock the axios.post response
    axios.post.mockResolvedValue({ data: { account: true, emailstatus:
true, passwordstatus: true } });
    // mock the useNavigate function
    const navigate = useNavigate();
    // click the login button
    fireEvent.click(registerButton);
    // wait for the axios.post call
    await expect(axios.post).toHaveBeenCalled();

  });
```

CreateUpdatePage

Code

```
import { fireEvent, render, screen,getByRole } from
"@testing-library/react";
import "@testing-library/jest-dom";
import { BrowserRouter, useLocation } from "react-router-dom";
import CreateUpdateMovie from "../Component/CreateUpdateMovie";

import axios from "axios";
describe(" Render CreateUpdate Component  ", () => {
  it("check exist of View button", async () => {
    const { getByRole } = render(
      <BrowserRouter>
      <CreateUpdateMovie/>
      </BrowserRouter>
    );
    const button=getByRole('link', { id: "ViewButton" })
    expect(button).toBeInTheDocument();
  });
});
describe("Render CreateUpdate Component ", () => {
    it("Check click event of view button", async () => {
      const { getByRole } = render(
      <BrowserRouter>
      <CreateUpdateMovie/>
      </BrowserRouter>
      );

      const button=getByRole('link', { id: "ViewButton" })

      fireEvent.click(button);


    });
  });
  test('shows error messages if fields is empty', () => {
    // render the component
    const { getByText ,getByTestId} = render(<BrowserRouter>
    <CreateUpdateMovie/>
    </BrowserRouter>);
```

```
    // get the elements

    const AddMovie = getByTestId('AddMovie');
    // click the login button

    fireEvent.click(AddMovie);
    // check if the error messages are shown
    expect(getByTestId('nameerror')).toHaveTextContent('**Required ! Must
fill the field **');
    expect(getByTestId('typeerror')).toHaveTextContent('**Required ! Must
fill the field **');
    expect(getByTestId('langerror')).toHaveTextContent('**Required ! Must
fill the field **');
    expect(getByTestId('duraerror')).toHaveTextContent('**Required ! Must
fill the field **');
  });
```

List Page
Code

```
import { fireEvent, render, screen } from "@testing-library/react";
import "@testing-library/jest-dom";
import { BrowserRouter } from "react-router-dom";
import axios from "axios";

import { ReactDOM } from "react-dom";
import ListMovie from '../Component/ListMovie';


describe("Check exist of Add Movie button ", () => {
  it("Renders the List component", async () => {
    const { getByRole } = render(
    <BrowserRouter>
    <ListMovie/>
    </BrowserRouter>
    );

    const button=getByRole('link', { name: "Add Movie" })
```

```
    expect(button).toBeInTheDocument();
  });
});

describe("Check Redirection to CreateUpdateComponent after clicking the
add movie button ", () => {
  it("Renders the List component", async () => {
    const { getByRole } = render(<BrowserRouter>
        <ListMovie/>
        </BrowserRouter>);

    const button=getByRole('link', { name: "Add Movie" })

    fireEvent.click(button);
    const title = global.window.location.href;
    expect(title).toBe("http://localhost/create");
  });
});
describe("Check exist of Table column movie name ", () => {
    it("Renders the List component", async () => {
      const { getByText } = render(
      <BrowserRouter>
      <ListMovie/>
      </BrowserRouter>
      );

      const MovieNameTitle=getByText("Movie Name" )

      expect(MovieNameTitle).toBeInTheDocument();
    });
  });

  describe("Check exist of Table column movie language ", () => {
    it("Renders the List component", async () => {
      const { getByText } = render(
      <BrowserRouter>
      <ListMovie/>
      </BrowserRouter>
      );
```

```
    const MovieLanguageTitle=getByText("Movie Language" )

    expect(MovieLanguageTitle).toBeInTheDocument();
  });
});
describe("Check exist of Table column movie type ", () => {
  it("Renders the List component", async () => {
    const { getByText } = render(
    <BrowserRouter>
    <ListMovie/>
    </BrowserRouter>
    );

    const MovieTypeTitle=getByText("Movie Type" )

    expect(MovieTypeTitle).toBeInTheDocument();
  });
});
describe("Check exist of Table column durations ", () => {
  it("Renders the List component", async () => {
    const { getByText } = render(
    <BrowserRouter>
    <ListMovie/>
    </BrowserRouter>
    );

    const MovieDurationsTitle=getByText("Movie Durations" )

    expect(MovieDurationsTitle).toBeInTheDocument();
  });
});
describe("Check exist of Table column action ", () => {
  it("Renders the List component", async () => {
    const { getByText } = render(
    <BrowserRouter>
    <ListMovie/>
    </BrowserRouter>
    );
```

```
      const Action=getByText("Action" )

      expect(Action).toBeInTheDocument();
    });
  });
```

Layout page
Code

```
import { render, screen } from '@testing-library/react';
import LayoutComponent from '../Component/LayoutComponent';

test('renders check the title of the sites', () => {
  render(<LayoutComponent />);
  const linkElement = screen.getByText(/movie center/i);
  expect(linkElement).toBeInTheDocument();
});
```

Home Page
Code

```
import { fireEvent, render, screen } from "@testing-library/react";
import "@testing-library/jest-dom";
import {  BrowserRouter,useLocation,useNavigate,Route, Routes } from
"react-router-dom";
import HomeComponent from "../Component/HomeComponent";
import ListMovie from '../Component/ListMovie'


describe(" Check exist of Total movies label ", () => {
  it("Renders the Home component", async () => {
    const { getByText } = render(<BrowserRouter>
      <HomeComponent/>
    </BrowserRouter>,);

    const label = getByText(/Total Movies/i);
```

```
    expect(label).toBeInTheDocument();
  });
});
describe(" Check exist of Add movie button ", () => {
    it("Renders the Home component", async () => {
      const { getByRole } = render(<BrowserRouter>
        <HomeComponent/>
      </BrowserRouter>,);

      const button=getByRole('link', { name: "Add Movie" })

      await expect(button).toBeInTheDocument();
    });
  });

  describe(" Check redirection after click add movie button ", () => {
    it("Renders the Home component", async () => {
      const { getByRole } = render(<BrowserRouter>
        <HomeComponent/>
      </BrowserRouter>);

      const button=getByRole('link', { name: "Add Movie" })

      fireEvent.click(button);
      const title = global.window.location.href;
      expect(title).toBe("http://localhost/create");
    });
  });
  describe(" Check whether the list button is exists", () => {
    it("Renders the Home component", async () => {
      const { getByRole } = render(<BrowserRouter>
        <HomeComponent/>
      </BrowserRouter>);

      const button=getByRole('link', { name: "List Movie" })

      expect(button).toBeInTheDocument();
    });
  });
```

```
  describe("Check whether the list button redirects to list component ",
() => {
    it("Renders the Home component", async () => {
      const { getByRole } = render(<BrowserRouter>
        <HomeComponent/>
      </BrowserRouter>,);

      const button=getByRole('link', { name: "List Movie" })

      fireEvent.click(button);
      const title = global.window.location.href;
      expect(title).toBe("http://localhost/list");
    });
  });
```

Result of above 24 valid test case

Passed

```
Test Suites: 6 passed, 6 total
Tests:       24 passed, 24 total
Snapshots:   0 total
Time:        26.654 s
Ran all test suites related to changed files.
```