Name: Nandhakumaran H

Employee ID: 12225

Case Study: Online Charity Management System

W3H Analysis:

Case Study: Online Charity Management System	
1. What?	2. How?
What are the modules required for online charity management System? **Ans:** 1. Admin Module 2. Employee Module 3. User Module **What are the functionalities needed for Admin module?** **Ans:** 1. Register and Login to the portal 2. Do CRUD operations on All Data **What are the functionalities needed for Employee Module?** **Ans:** 1. Register and Login to the portal 2. Do CRUD operation on Employee Data 3. Handle requests of users and response accordingly **What are the functionalities needed for User Module?** **Ans:** 1. Register and Login to the portal 2. Do CRUD operation on User Data 3. Raise Request	**Admin:** **Register and Login:** **Method 1:** Register and Login by Admin Id and password **Method 2:** Register and Login by email id and password **Method 3:** Register and Login by Admin Name and Password **CRUD:** **Method 1:** Do Crud operations by unique values **Method 2:** Do Crud Operations by Duplicable values **Method 3:** Do Crud Operations by Manually search the Users and Employees **Method 4:** Do Crud Operations by Selecting the Employee and User in Dropdown Menu **Employee:** **Register and Login:** **Method 1:** Register and Login by Employee Name and password **Method 2:** Register and Login by email id and password

	Method 3: Register and Login by Employee id and password. **Method 4:** Register and Login by Social Media Accounts **CRUD:** **Method 1:** Do Crud operations by unique values like primary keys **Method 2:** Do Crud Operations by Duplicable values **Method 3:** Do Crud Operations by Manually search the Requests **Method 4:** Do Crud Operations by Selecting the Request in Dropdown Menu **Handle requests and Response:** **Method 1:** Handle the requests and responses of users by their User Id and donation type **Method 2:** Handle the requests and responses of users by their addresses **Method 3:** Handle the requests and responses by users by their frequent donation history and any other third-party data. **User:** **Register and Login:** **Method 1:** Register and Login by Username and password **Method 2:** Register and Login by email id and password **Method 3:** Register and Login by User Id and password **CRUD:**

	Method 1: Do Crud operations by unique values like User Ids. **Method 2:** Do Crud Operations by Duplicable values like Addresses. **Raise Request:** **Method 1:** Raise Requests by typing what type of donation they want to donate **Method 2:** Raise requests by selecting the existing donation formats **Method 3:** Raise Requests by copying other user's donation type or sample donation type formats
Admin: **Register and Login:** **Method 1:** Register and Login by Admin Id and password **(**Unique values like Admin Id is most preferrable for Register and Login for Admin part of side) **CRUD:** **Method 1:** Do Crud operations by unique values (Unique values are the best way handle data anywhere) **Employee:** **Register and Login:** **Method 1:** Register and Login by Employee Name and password **(**Usernames are most preferable for Register and Login for Employee side)	**Admin, Employee and User:** • Register and Login by email id and social media account is not a best approachable way for the trustable website like charity websites. • CRUD Operation by duplicable values will lead to Loss of data • CRUD Operations by manually searching the data without using primary key is not an effective way to do crud operations. • CRUD Operations by Dropdown menu by selecting the data is not an efficient way to find or select the data. • Handling requests and responses by duplicable values is the inappropriate way to handle the data. • Handling requests and responses by previous donation history to any

CRUD: **Method 1:** Do Crud operations by unique values like Employee Ids. (Unique values are the best way handle data anywhere) **Handle requests and Response:** **Method 1:** Handle the requests and responses of users by their User Id and donation types (Unique values like User Ids will avoid the confusion among user during updating the donation status) **User:** **Register and Login:** **Method 1:** Register and Login by Username and password (Unique values like usernames are most preferable for Register and Login for Users Side) **CRUD:** **Method 1:** Do Crud operations by unique values like primary keys (Unique values are the best way handle data anywhere) **Raise Request:** **Method 1:** Raise Requests by typing what type of donation they want to donate **(**Type the donation type will be the most efficient way to raise a request**)**	other third-party data is the inappropriate way to handle the data. • Raise requests by selecting the existing donation formats or copying other user's donation formats will make the user lose their genuine interest to select the things which they want to mention or donate
3. Why?	4. Why Not?

Junit Test cases:

```
package com.assessment.codequality.serviceimplementation;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import com.assessment.codequality.model.Registration;

@SpringBootTest
class RegistrationServiceImplementationTest {

@Autowired
private RegistrationServiceImplementation registrationService;


@Test
void testAddRegistration() {
Registration registration = new Registration(0, "suriya",
"donator", "sss@gmail.com", "suriya123");
assertEquals("success",
registrationService.addRegistration(registration));
}
@Test
void testAddRegistrationFailure() {
Registration registration = null;
assertEquals("failure",
registrationService.addRegistration(registration));
}


@Test
void testUpdateRegistration() {
Registration obj = new Registration(6, "nandha", "employee",
"nnn@gmail.com", "nandha123");
registrationService.updateRegistration(obj);
obj.setName("Suriyakumaran");
assertEquals("success",
registrationService.updateRegistration(obj));
Registration retrievedRegistration =
registrationService.getRegistration(6);
assertNotNull(retrievedRegistration);
assertEquals("Suriyakumaran", retrievedRegistration.getName());
}

@Test
void testUpdateRegistrationNull() {
Registration obj = null;
assertEquals("failure",
registrationService.updateRegistration(obj));
}

@Test
void testGetRegistration() {
Registration retrievedRegistration =
registrationService.getRegistration(8);
assertNotNull(retrievedRegistration);
assertEquals("ponraj", retrievedRegistration.getName());
assertEquals("nandha123", retrievedRegistration.getPassword());
}

@Test
void testGetRegistration_NotFound() {
 Registration retrievedRegistration =
registrationService.getRegistration(99);
assertNull(retrievedRegistration);
}

@Test
void testDeleteRegistration() {
assertEquals("success",
registrationService.deleteRegistrationById(9));
}

@Test
void testDeleteRegistrationNotFound() {
assertEquals("failure",
registrationService.deleteRegistrationById(100));
}
```

```
}


package com.assessment.codequality.serviceimplementation;


import com.assessment.codequality.model.Request;
import com.assessment.codequality.model.User;
import com.assessment.codequality.repository.RequestRepository;
import com.assessment.codequality.service.RequestService;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;
import static org.junit.jupiter.api.Assertions.assertEquals;

@SpringBootTest
class RequestServiceImplementationTest {

@Autowired
RequestRepository requestRepository;
@Autowired
RequestService requestService;

@Test
void testSaveAddRequest() {
User user = new User(7, "ponraj", "nnn@gmail.com", "Madurai",
123456789, "Dress", "Pending");
Request obj = new Request(0,user);
assertEquals("success", requestService.saveAddRequest(obj));
}

@Test
void testSaveAddRequest2() {
Request obj = null;
assertEquals("failure", requestService.saveAddRequest(obj));
}
```

```
@Test
void testGetRequestById() {
int id = 1;
Request request = new Request();
requestRepository.saveRequest(request);

String result = requestService.getRequestById(id);

assertEquals("success", result);
}

@Test
void testGetRequestByIdWithNullRequest() {
int id = 99;

String result = requestService.getRequestById(id);

assertEquals("failure", result);
}


@Test
void testUpdateIdRequest() {

User user = new User(7, "ponraj", "nnn@gmail.com", "Madurai",
123456789, "Dress", "Pending");
Request obj = new Request(19,user);

String result = requestService.updateIdRequest(obj);

assertEquals("success", result);
}

@Test
void testUpdateIdRequestWithNullRequest() {
String result = requestService.updateIdRequest(null);

assertEquals("failure", result);
}
```

```java
@Test
void testDeleteIdRequest() {
int id = 18;
Request request = new Request();
requestRepository.saveRequest(request);

String result = requestService.deleteIdRequest(id);

assertEquals("success", result);
}

@Test
void testDeleteIdRequestWithNonExistingRequest() {
int id = 100;

String result = requestService.deleteIdRequest(id);

assertEquals("failure", result);
}
}
```

package com.assessment.codequality.serviceimplementation;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.Test;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.boot.test.context.SpringBootTest;

import com.assessment.codequality.model.User;

import com.assessment.codequality.service.UserService;

```
@SpringBootTest
class UserServiceImplementationTest {

  @Autowired
  private UserService userService;


  @Test
  void testSaveUser() {
    User obj = new User(0, "Nagarjun", "naga@gmail.com", "Madurai", 123456789,
"Money", "Pending");
    assertEquals("success", userService.saveAddUser(obj));
  }

  @Test
  void testSaveUserNull() {
    User obj = null;
    assertEquals("failure", userService.saveAddUser(obj));
  }

  @Test
  void testUpdateUser() {
        User obj = new User(6, "nandha", "nnn@gmail.com", "Madurai", 123456789,
"Dress", "Pending");
        userService.updateIdUser(obj);
        obj.setName("Gokul");
        assertEquals("success", userService.updateIdUser(obj));

        User retrievedUser = userService.getUserById(6);
        assertEquals("Gokul", retrievedUser.getName());
    }

    @Test
    void testUpdateUserNull() {
        User obj = null;
        assertEquals("failure", userService.updateIdUser(obj));
    }

    @Test
    void testGetUserById() {

        User retrievedUser = userService.getUserById(7);
        assertNotNull(retrievedUser);
        assertEquals("ponraj", retrievedUser.getName());
        assertEquals("Dress", retrievedUser.getDonation());
        assertEquals(123456789, retrievedUser.getPhoneNumber());
    }
```

```
  @Test

  void testGetUserById_NotFound() {

        User retrievedUser = userService.getUserById(99);

    assertNull(retrievedUser);

  }


  @Test

  void testDeleteUser() {


    assertEquals("success", userService.deleteIdUser(10));

  }


  @Test

  void testDeleteUserNotFound() {


    assertEquals("failure", userService.deleteIdUser(100));

  }

}
```

Jest test cases:

```
import { render, screen, fireEvent, waitFor } from '@testing-library/react';
import { useNavigate } from 'react-router-dom'
import axios from 'axios';
import MockAdapter from 'axios-mock-adapter';
import AddUser from './pages/AddUser';
```

```
jest.mock('react-router-dom', () => ({
    ...jest.requireActual('react-router-dom'),
    useNavigate: jest.fn(),
  }));

  const mock = new MockAdapter(axios);

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

test('renders Add User title ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("heading");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("DONATE");
});

test('renders Username Label ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("unamelabel");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Name");
});

test('renders Username  ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("uname");
    expect(linkElement).toBeInTheDocument();
});

test('renders Email Id Label ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("emaillabel");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Email ID");
});

test('renders Email Id  ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("email");
    expect(linkElement).toBeInTheDocument();
});

test('renders Address Label ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("addresslabel");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Address");
});

test('renders Address  ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("address");
    expect(linkElement).toBeInTheDocument();
});

test('renders Phone Number Label ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("phnlabel");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Phone Number");
});

test('renders Phone Number  ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("phn");
    expect(linkElement).toBeInTheDocument();
});

test('renders Donation Label ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("donationlabel");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Donation");
});

test('renders Donation  ', () => {
    render(<AddUser />);
    const linkElement = screen.getByRole("donation");
    expect(linkElement).toBeInTheDocument();
});
```

```
it("renders 'Donate' ", () => {
    render(<AddUser />);
    const loginButton = screen.getByRole('button', { name: "Submit" });
    expect(loginButton).toBeInTheDocument();
  });
```

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import { useNavigate, useParams } from 'react-router-dom';
import axios from 'axios';
import Edit from './pages/Edit';
import MockAdapter from 'axios-mock-adapter';



jest.mock('axios');

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

//   const mock = new MockAdapter(axios);

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

describe('Edit', () => {

//   beforeEach(() => {
//     render(<Edit />);
//   });



  test('Request Edit labels',  () => {
    render(<Edit />);
    expect(screen.getByRole('heading')).toBeInTheDocument();
    expect(screen.getByRole('requestlabel')).toBeInTheDocument();
    expect(screen.getByRole('useridlabel')).toBeInTheDocument();
    expect(screen.getByRole('namelabel')).toBeInTheDocument();
    expect(screen.getByRole('emaillabel')).toBeInTheDocument();
    expect(screen.getByRole('addresslabel')).toBeInTheDocument();
    expect(screen.getByRole('phnlabel')).toBeInTheDocument();
    expect(screen.getByRole('donationlabel')).toBeInTheDocument();
    expect(screen.getByRole('statuslabel')).toBeInTheDocument();

  });

  test('renders Edit Input Fields', () => {
    render(<Edit />);
    expect(screen.getByRole('request')).toBeInTheDocument();
    expect(screen.getByRole('userid')).toBeInTheDocument();
    expect(screen.getByRole('name')).toBeInTheDocument();
    expect(screen.getByRole('email')).toBeInTheDocument();
    expect(screen.getByRole('address')).toBeInTheDocument();
    expect(screen.getByRole('phn')).toBeInTheDocument();
    expect(screen.getByRole('donation')).toBeInTheDocument();
    expect(screen.getByRole('status')).toBeInTheDocument();
  });

  test('Update', () => {
    render(<Edit />);
    expect(screen.getByRole('button', {name: "Update
Status"})).toBeInTheDocument();
  });



});
```

```

import React from 'react';
import { render, screen } from '@testing-library/react';
import axios from 'axios';
import { useNavigate, useParams } from 'react-router-dom';
import MockAdapter from 'axios-mock-adapter';
import EditUser from './pages/EditUser';

jest.mock('axios');

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

//   const mock = new MockAdapter(axios);

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

describe('EditUser', () => {
//   beforeEach(() => {
//     render(<EditUser />);
//   });



  test('renders EditUser form',  () => {
    render(<EditUser />);
    expect(screen.getByRole('heading', { name: "UPDATE USER
DATA" })).toBeInTheDocument();
    expect(screen.getByRole('userid')).toBeDisabled();
    expect(screen.getByRole('name')).toBeInTheDocument();
    expect(screen.getByRole('email')).toBeInTheDocument();
    expect(screen.getByRole('address')).toBeInTheDocument();
    expect(screen.getByRole('phn')).toBeInTheDocument();
    expect(screen.getByRole('donation')).toBeInTheDocument();
  });

  test('updates user data on form submission', async() => {
    render(<EditUser />);
    expect(screen.getByRole('name')).toBeInTheDocument();
    expect(screen.getByRole('email')).toBeInTheDocument();
    expect(screen.getByRole('address')).toBeInTheDocument();
    expect(screen.getByRole('phn')).toBeInTheDocument();
    expect(screen.getByRole('donation')).toBeInTheDocument();
  });

  test('Update User', () => {
    render(<EditUser />);
    expect(screen.getByRole('button', {name: "Update
User"})).toBeInTheDocument();
  });
});
```

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import Hero from './components/Hero';

describe('Hero Component', () => {

    beforeEach(() => {
        render(<Hero />);
      });

  test('should render Hero component', () => {

    expect(screen.getByRole('heading')).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Log in"})).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Sign Up"})).toBeInTheDocument();
    expect(screen.getByRole('textbox')).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Get
Started"})).toBeInTheDocument();

  });



});
```

```
import { render, screen, fireEvent, waitFor } from '@testing-library/react';
import { useNavigate } from 'react-router-dom'
import axios from 'axios';
import MockAdapter from 'axios-mock-adapter';
import Login from './components/Login';



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

const mock = new MockAdapter(axios);

const mockNavigate = jest.fn();
useNavigate.mockReturnValue(mockNavigate);


it("renders 'Login Name text' " , () =>{
  render(<Login />);
const linkElement = screen.getByRole("userlabel");
expect(linkElement).toBeInTheDocument();
})

it("renders 'User Type text' " , () =>{
  render(<Login />);
const linkElement = screen.getByRole("usertypelabel");
expect(linkElement).toBeInTheDocument();
})

it("renders 'Password text' " , () =>{
    render(<Login />);
  const linkElement = screen.getByRole("pwddlabel");
  expect(linkElement).toBeInTheDocument();
  })

it("renders 'submit button' ", () => {
  render(<Login />);
  const loginButton = screen.getByRole('button', { name: "Login" });
  expect(loginButton).toBeInTheDocument();
});

it("renders 'continue with Google button' ", () => {
  render(<Login />);
  const loginButton = screen.getByRole('button', { name: "Continue with
Google" });
  expect(loginButton).toBeInTheDocument();
});

it("renders 'Navigate Back' ", () => {
  render(<Login />);
  const loginButton = screen.getByRole('button', { name: "Go Back" });
  expect(loginButton).toBeInTheDocument();
});
```

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import Navbar1 from './components/Navbar1';
import { useNavigate } from 'react-router-dom'


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

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);


describe('Navbar1 Component', () => {
  test('should render Navbar1 component', () => {
    render(<Navbar1 />);

    expect(screen.getByRole('navigation')).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Donate"})).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Donator
Details"})).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "View
notifications"})).toBeInTheDocument();
  });
});
```

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import Navbar2 from './components/Navbar2';
import { useNavigate } from 'react-router-dom'
```

```
jest.mock('react-router-dom', () => ({
    ...jest.requireActual('react-router-dom'),
    useNavigate: jest.fn(),
  }));

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);


describe('Navbar2 Component', () => {
  test('should render Navbar2 component', () => {
    render(<Navbar2 />);

    expect(screen.getByRole('navigation')).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "Handle
Requests"})).toBeInTheDocument();
    expect(screen.getByRole('button', {name: "View
notifications"})).toBeInTheDocument();
  });
});
```

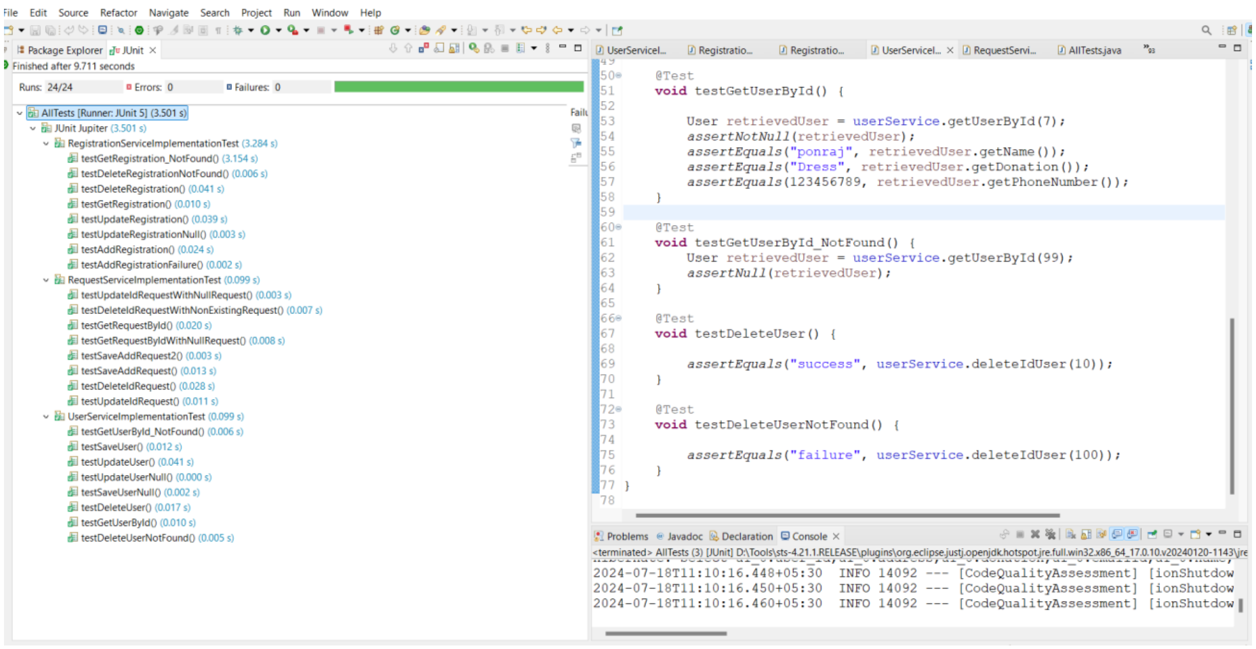
```
import React from 'react';
import axios from "axios";
import { render, screen, fireEvent } from '@testing-library/react';
import Signup from './components/Signup';
import { useNavigate } from 'react-router-dom'
import MockAdapter from 'axios-mock-adapter';

jest.mock('axios');

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

//   const mock = new MockAdapter(axios);
```

```
  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

describe('Register Component', () => {
  beforeEach(() => {
    render(<Signup />);
  });

  test('should render Register component', () => {
    expect(screen.getByRole('heading')).toBeInTheDocument();
    expect(screen.getByRole('namelabel')).toBeInTheDocument();
    expect(screen.getByRole('usertypelabel')).toBeInTheDocument();
    expect(screen.getByRole('emaillabel')).toBeInTheDocument();
    expect(screen.getByRole('pwdlabel')).toBeInTheDocument();
  });

  test('should update state on input change', () => {
    const nameInput = screen.getByRole('name');
    expect(nameInput).toBeInTheDocument();

    const typeInput = screen.getByRole('usertype');
    expect(typeInput).toBeInTheDocument();


    const emailInput = screen.getByRole('email');
    expect(emailInput).toBeInTheDocument();


    const passwordInput = screen.getByRole('pwd');
    expect(passwordInput).toBeInTheDocument();

  });

  it("renders 'submit button' ", () => {
    const signupButton = screen.getByRole('button', { name: "Sign Up" });
    expect(signupButton).toBeInTheDocument();
  });

  it("renders 'continue with Google button' ", () => {
    const signupButton = screen.getByRole('button', { name: "Continue with
Google" });
```

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

  it("renders 'Navigate Back' ", () => {
    const signupButton = screen.getByRole('button', { name: "Go Back" });
    expect(signupButton).toBeInTheDocument();
  });



});
import React from 'react';
import { render, screen, fireEvent, waitFor } from '@testing-library/react';
import ViewRequest from './pages/ViewRequest';
import { useNavigate, useParams } from 'react-router-dom';

jest.mock('axios');

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

//   const mock = new MockAdapter(axios);

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

describe('ViewRequest', () => {

    test('should render Register component', () => {
    render(<ViewRequest />);

      expect(screen.getByRole("Requests")).toBeInTheDocument();
      expect(screen.getByRole('request')).toBeInTheDocument();
      expect(screen.getByRole('userid')).toBeInTheDocument();
      expect(screen.getByRole('name')).toBeInTheDocument();
      expect(screen.getByRole('email')).toBeInTheDocument();
      expect(screen.getByRole('address')).toBeInTheDocument();
      expect(screen.getByRole('phn')).toBeInTheDocument();
      expect(screen.getByRole('donation')).toBeInTheDocument();
      expect(screen.getByRole('status')).toBeInTheDocument();
      expect(screen.getByRole("action")).toBeInTheDocument();
    });

    test('buttons in the Request Table', () => {
        render(<ViewRequest />);
        expect(screen.getByRole("button", {name: "Update"})).toBeInTheDocument();
        expect(screen.getByRole("button", {name: "Delete"})).toBeInTheDocument();
    });


  });
```

```
import React from 'react';
import { render, screen } from '@testing-library/react';
import axios from 'axios';
import { useNavigate, useParams } from 'react-router-dom';
import MockAdapter from 'axios-mock-adapter';
import ViewUser from './pages/ViewUser';

jest.mock('axios');

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

//   const mock = new MockAdapter(axios);

  const mockNavigate = jest.fn();
  useNavigate.mockReturnValue(mockNavigate);

describe('ViewUser tests', () => {
  test('renders ViewUser table without input data', () => {
    render(<ViewUser />);
    expect(screen.getByRole('userid')).toBeInTheDocument();
    expect(screen.getByRole('name')).toBeInTheDocument();
    expect(screen.getByRole('email')).toBeInTheDocument();
    expect(screen.getByRole('address')).toBeInTheDocument();
    expect(screen.getByRole('phn')).toBeInTheDocument();
    expect(screen.getByRole('donation')).toBeInTheDocument();
    expect(screen.getByRole('status')).toBeInTheDocument();
    expect(screen.getByRole("action")).toBeInTheDocument();
  });

  test("renders 'Add Button' ", () => {
    render(<ViewUser />);
    expect(screen.getByRole("button", {name: "Add +"})).toBeInTheDocument();
  });
});
```

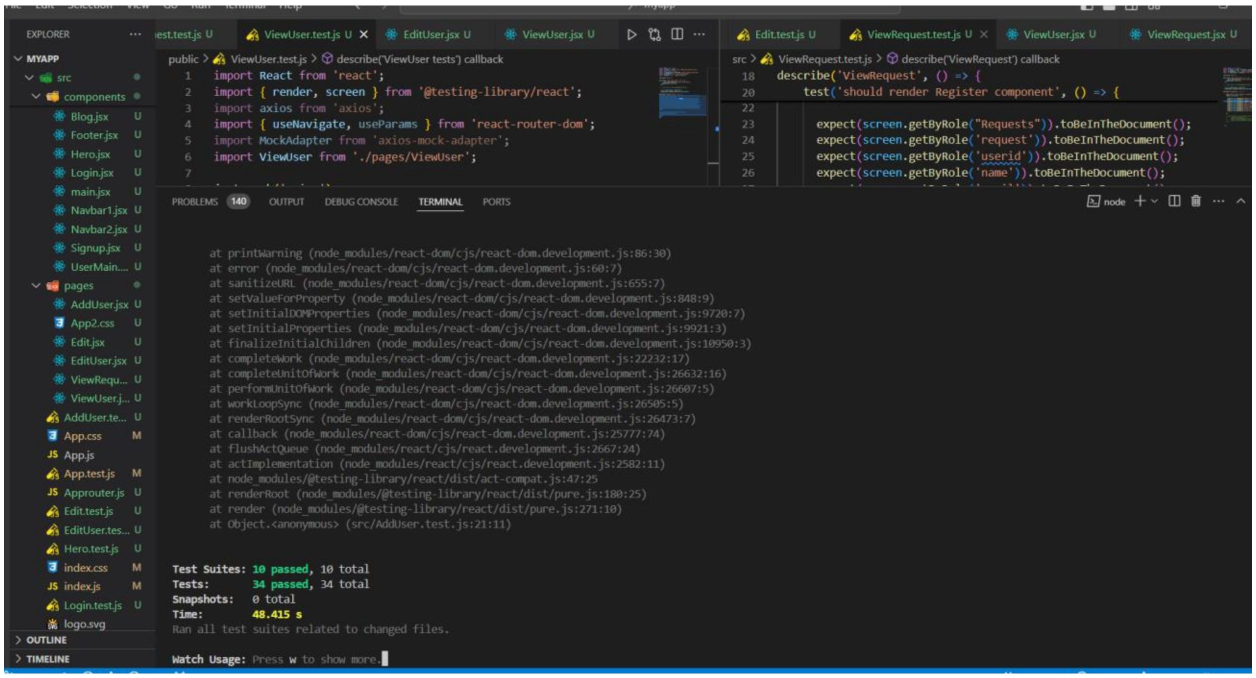
ScreenShots:



```
Finished after 9.711 seconds
Runs: 24/24    Errors: 0    Failures: 0

AllTests [Runner: JUnit 5] (3.501 s)
  JUnit Jupiter (3.501 s)
    RegistrationServiceImplementationTest (3.284 s)
      testGetRegistration_NotFound() (3.154 s)
      testDeleteRegistrationNotFound() (0.006 s)
      testDeleteRegistration() (0.041 s)
      testGetRegistration() (0.010 s)
      testUpdateRegistration() (0.039 s)
      testUpdateRegistrationNull() (0.003 s)
      testAddRegistration() (0.024 s)
      testAddRegistrationFailure() (0.002 s)
    RequestServiceImplementationTest (0.099 s)
      testUpdateIdRequestWithNullRequest() (0.003 s)
      testDeleteIdRequestWithNonExistingRequest() (0.007 s)
      testGetRequestById() (0.020 s)
      testGetRequestByIdWithNullRequest() (0.008 s)
      testSaveAddRequest2() (0.003 s)
      testSaveAddRequest() (0.013 s)
      testDeleteIdRequest() (0.028 s)
      testUpdateIdRequest() (0.011 s)
    UserServiceImplementationTest (0.099 s)
      testGetUserById_NotFound() (0.006 s)
      testSaveUser() (0.012 s)
      testUpdateUser() (0.041 s)
      testUpdateUserNull() (0.000 s)
      testSaveUserNull() (0.002 s)
      testDeleteUser() (0.017 s)
      testGetUserById() (0.010 s)
      testDeleteUserNotFound() (0.005 s)
```

```java
    @Test
    void testGetUserById() {

        User retrievedUser = userService.getUserById(7);
        assertNotNull(retrievedUser);
        assertEquals("ponraj", retrievedUser.getName());
        assertEquals("Dress", retrievedUser.getDonation());
        assertEquals(123456789, retrievedUser.getPhoneNumber());
    }

    @Test
    void testGetUserById_NotFound() {
        User retrievedUser = userService.getUserById(99);
        assertNull(retrievedUser);
    }

    @Test
    void testDeleteUser() {

        assertEquals("success", userService.deleteIdUser(10));
    }

    @Test
    void testDeleteUserNotFound() {

        assertEquals("failure", userService.deleteIdUser(100));
    }
}
```

```
<terminated> AllTests (3) [JUnit] D:\Tools\sts-4.21.1.RELEASE\plugins\org.eclipse.justj.openjdk.hotspot.jre.full.win32.x86_64_17.0.10.v20240120-1143\jre
2024-07-18T11:10:16.448+05:30  INFO 14092 --- [CodeQualityAssessment] [ionShutdow
2024-07-18T11:10:16.450+05:30  INFO 14092 --- [CodeQualityAssessment] [ionShutdow
2024-07-18T11:10:16.460+05:30  INFO 14092 --- [CodeQualityAssessment] [ionShutdow
```



```
import React from 'react';
import { render, screen } from '@testing-library/react';
import axios from 'axios';
import { useNavigate, useParams } from 'react-router-dom';
import MockAdapter from 'axios-mock-adapter';
import ViewUser from './pages/ViewUser';
```

```
describe('ViewRequest', () => {
    test('should render Register component', () => {

        expect(screen.getByRole("Requests")).toBeInTheDocument();
        expect(screen.getByRole('request')).toBeInTheDocument();
        expect(screen.getByRole('userid')).toBeInTheDocument();
        expect(screen.getByRole('name')).toBeInTheDocument();
```

```
      at printWarning (node_modules/react-dom/cjs/react-dom.development.js:86:30)
      at error (node_modules/react-dom/cjs/react-dom.development.js:60:7)
      at sanitizeURL (node_modules/react-dom/cjs/react-dom.development.js:655:7)
      at setValueForProperty (node_modules/react-dom/cjs/react-dom.development.js:848:9)
      at setInitialDOMProperties (node_modules/react-dom/cjs/react-dom.development.js:9720:7)
      at setInitialProperties (node_modules/react-dom/cjs/react-dom.development.js:9921:3)
      at finalizeInitialChildren (node_modules/react-dom/cjs/react-dom.development.js:10950:3)
      at completeWork (node_modules/react-dom/cjs/react-dom.development.js:22232:17)
      at completeUnitOfWork (node_modules/react-dom/cjs/react-dom.development.js:26632:16)
      at performUnitOfWork (node_modules/react-dom/cjs/react-dom.development.js:26607:5)
      at workLoopSync (node_modules/react-dom/cjs/react-dom.development.js:26505:5)
      at renderRootSync (node_modules/react-dom/cjs/react-dom.development.js:26473:7)
      at callback (node_modules/react-dom/cjs/react-dom.development.js:25777:74)
      at flushActQueue (node_modules/react/cjs/react.development.js:2667:24)
      at actImplementation (node_modules/react/cjs/react.development.js:2582:11)
      at node_modules/@testing-library/react/dist/act-compat.js:47:25
      at renderRoot (node_modules/@testing-library/react/dist/pure.js:180:25)
      at render (node_modules/@testing-library/react/dist/pure.js:271:10)
      at Object.<anonymous> (src/AddUser.test.js:21:11)

Test Suites: 10 passed, 10 total
Tests:       34 passed, 34 total
Snapshots:   0 total
Time:        48.415 s
Ran all test suites related to changed files.

Watch Usage: Press w to show more.
```