ONLINE MATRIMONIAL SERVICE

NAME : VASHANTH TS

EMP ID : 12221

W3H ANALYSIS

PROJECT: ONLINE MATRIMONIAL SERVICE

What?

1.What are the modules Required?
ANS: (a). Application Admin Module
(b). Match Maker Module
(c). User Module

2.What are the functionalities required for Admin module?
ANS:
1. Verifies User Login to the application using Email Id and Password.
2. Perform CRUD functions to Users Profile and Match making module.

3.What are the functionalities required for Match Maker Module?
ANS:
1. Login to the Application using Registered id and password
2. Can view User Profile details .
3. Can Perform CRUD on users .

4.What are the Functionalities required for User Module?
ANS:
1. Login to the portal using User name/Email id and Password.
2. Select matches for thier profile ,Can view, edit their profile and Provide Ratings.
3. Can Contact with the mutually matched profiles .

5.What are the Fields required for User Module?
ANS: User Name, Email , Password , Mobile number, Address, Age, Job/Business , Salary , Property Details , Family details , Marital Status, Religion, Astrology and Horoscopic Details .

7.What are the Fields required for Match Maker Module?
ANS: Registered Id , Name ,Password , Work Timings , Address , Contact details .

8.What are the Fields required for Admin Module?
ANS: Admin id , Password , User details , Grooms ,Brides ,Matched Profiles, Match Makers details.

How?

1.Admin:

a) Verify Login
Method 1: Verify login using email id and password.
Method 2: Verify login using Social media account / username and password.

b) CRUD
Method 1: Implement the create/delete / view /
update in Application by manually selecting the User email /Registered id as it is unique.
Method 2: Implement the create/ delete / view / update for application by selecting username from listings.
Method 3: Implement the create/delete / view / update by listing all the details of Grooms and Brides as a table and providing an individual create match for users and edit/delete button for user .

2.MatchMaker module:
a) Login
Method 1: Login using Registered id and password.
Method 2: login using username and password.

b) View and Match the Profiles
Method 1: Match profiles of users according to the requests received in application and send notification about it and share the Brides/Grooms profiles as Matched , for further process.

c)Perform CRUD on Matchings
Method 1: Manually entering in the Matched Category in Application by listing profile details .
Method 2: Perform CRUD as a Entire list for every match.

3.User :

a) Login
Method 1: login using User email id and password
Method 2: login using Username and password

b) Select matching pair's profile, Can View / Edit their Profiles
Method 1: Can View and Edit profiles in application by Log in to the portal ,
Select matches from the specific section or Search by filter options.
Method 2: Can view matches in Dashboard and then log in and continue for further proceedings.

c) Connect & Interact
Method 1: Contact and interact with profiles shown as matched suggestions.
Method 2: Select the mutually matched profiles and make a connect , then interact ,Proceed further .

WHY ?

WHY NOT ??

1.Admin:

a) Verify Login

Method 1: Verify login using email id and password.

(It makes Admin to verify and authenticate the originality of the user login .)

b) CRUD

Method 1: Implement the create/delete / view /
update in Application by manually selecting the User email /Registered id as it is unique.

(It is easy for admin to enter the user id and perform
the crud operation on portal)

2. Match Maker module:

a) Login

Method 1: login using Registered Id and password.

(Login using id and password is necessary)

b) View and Match the Profiles

Method 1: Match profiles of users according to the requests received in application and send notification about it and share the Brides/Grooms profiles as Matched , for further process.

(It gives clear details to the users to choose correct match and is easy to follow up further)

c)Perform CRUD on Matchings

Method 1: Manually entering in the Matched Category in Application by listing profile details .

(It reduces the time in searching matched profiles from Grooms/Brides category from list)

3.User :

a) Login

Method 1: login using User email id and password

(Login using email id and Password is efficient to authenticate the originality of the user.)

b) Select matching pair's profile, Can View / Edit their Profiles

Method 1: Can View and Edit profiles in application by Log in to the portal ,
Select matches from the specific section or Search by filter options.

(It reduces the time and makes match easy and faster , without login user cant contact or interact and proceed)

c) Connect & Interact

Method 2: Select the mutually matched profiles and make a connect , then interact ,Proceed further

(After selecting the mutually matched profiles and contact them by sharing details will be a good choice , so that user can can know each other personally and avoid misunderstandings and can do cross verfications)

Admin:

1) To verify the originality of the user it is highly recommended to login using email id/ mobile number and Password so that username and social media account cannot be great option as it invloves payment options.
2)Crud operations using list takes much time for data retrieval .
3)If there is a table maintained for each match of user , it makes the Database clumsy and creates ambiguity issues and takes more memory space to combine entire matches .

Match Maker Module :

A) It is efficient to login using Registered Id and Password ,so that username cannot be great option.

B) Performing CRUD from the entire list takes much time .

User Module :

A) It is recommended to login using email id and Password , because several users may have similar user name so that username cannot be great option. Avoid using social media account .
Login with registrational id make admin to know this is a Bride / Groom account and then they can provide profiles according to it..

B)Viewing matched profiles without log in is not a efficient method and user can not able to contact.

C)Selecting all the profiles at a time and proceeding for connect is wastage of time and service .Updating After Every Process makes easier to manage application and service even more good

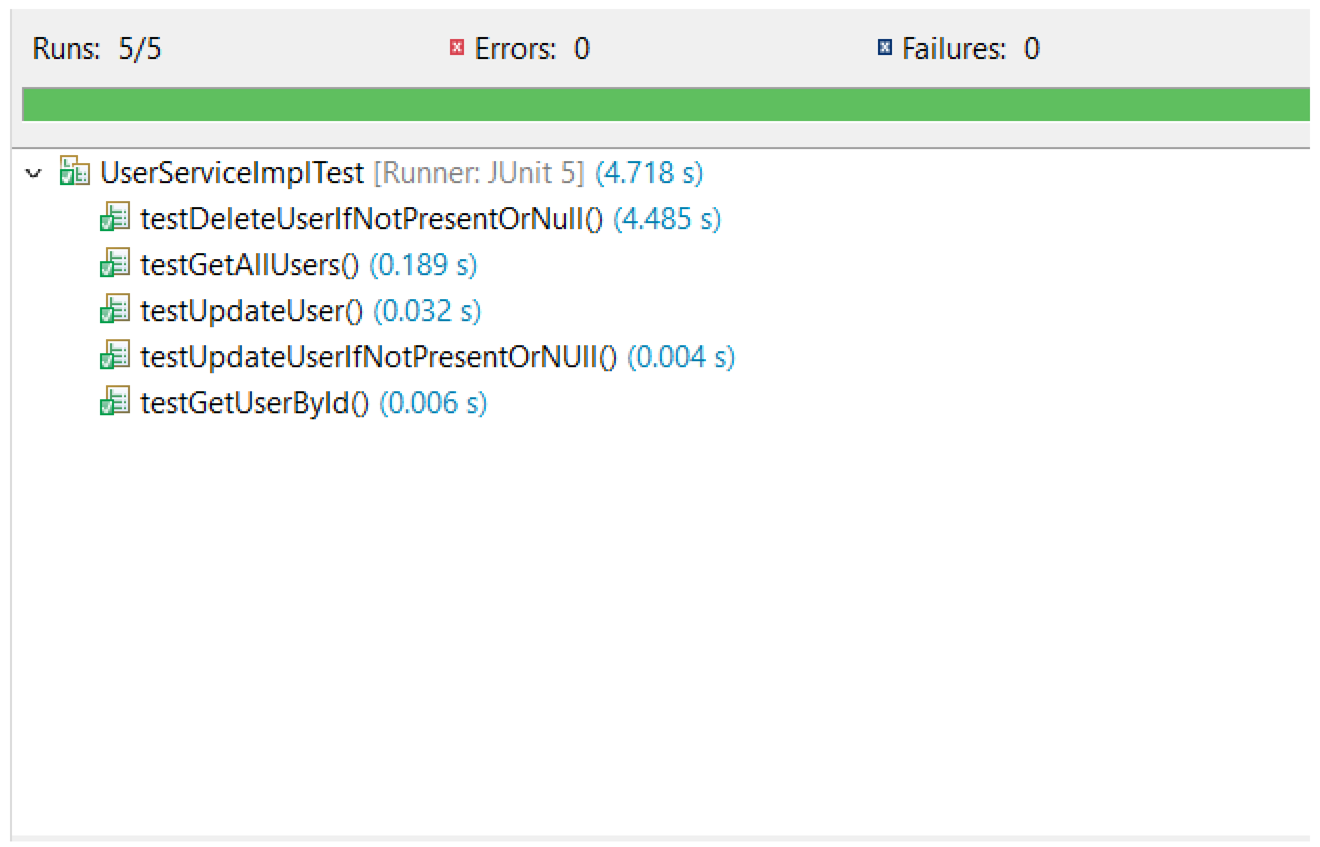
BACKEND CODE

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

import java.util.List;

import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import com.hibernate.matrimonial.model.Matchmaker;
import com.hibernate.matrimonial.model.User;
import com.hibernate.matrimonial.repository.UserRepository;
@SpringBootTest
class UserServiceImplTest {

@Autowired
UserServiceImpl userServiceImpl;

@Test
void testGetUserById() {
assertNotNull(userServiceImpl.getUserById(11));
}

@Test
@Order(1)
void testSaveUser() {
Matchmaker matchmaker = new Matchmaker(
6, "Subramaniyan", "8899001123", "Gandipuram,Coimbatore");



User user = new User(
0,
"Babu",
"Groom","No-9,BalrangaPuram,Madurrai",30,"IT","Monthly","Hindu","60000","Unmarried",matchmaker);
```

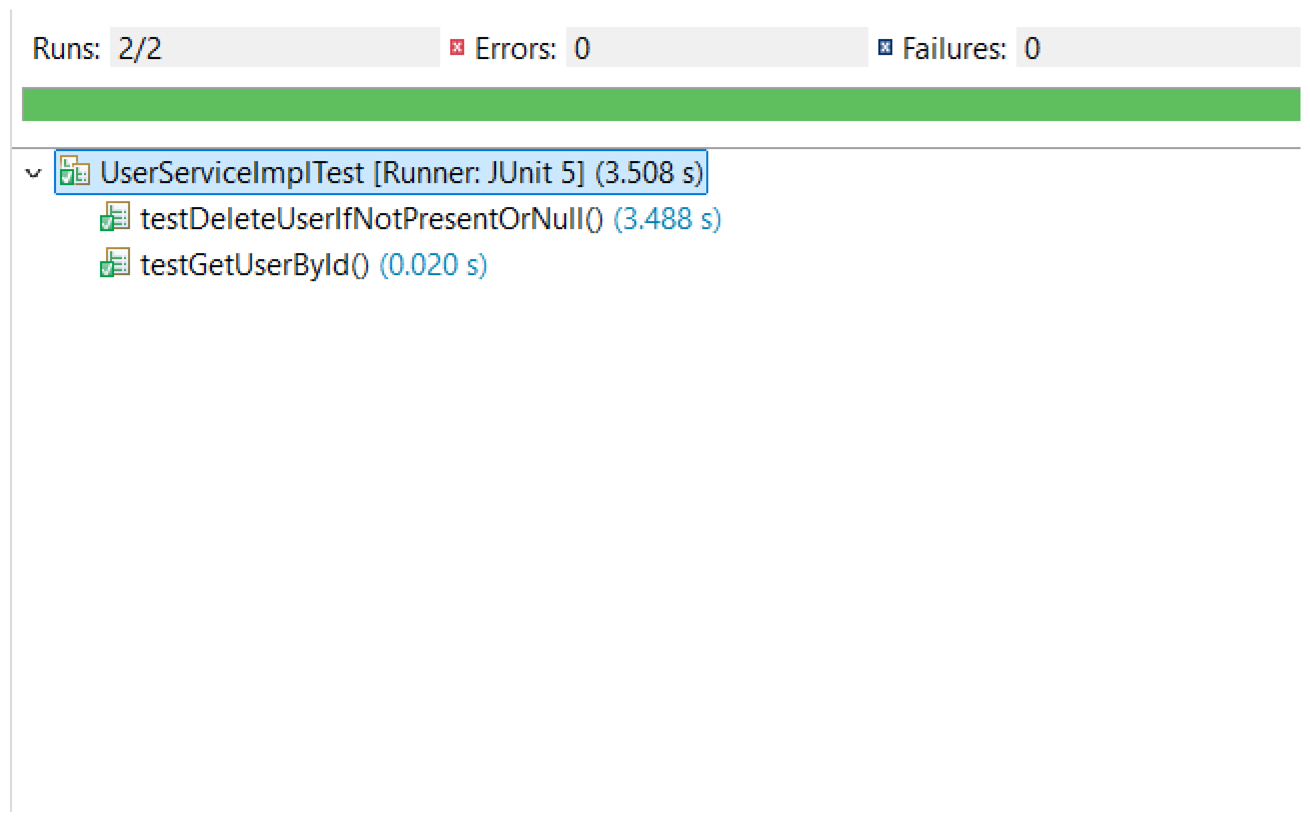
```
assertEquals("Success", userServiceImpl.saveUser(user));
}
//
@Test

void testSaveUserIFNull() {

User user = null;



assertEquals("Failure", userServiceImpl.saveUser(user));
}

@Test

void testDeleteUser() {
assertEquals("Success",userServiceImpl.deleteUser(21));
}

@Test

void testDeleteUserIfNotPresentOrNull() {
assertEquals("Failure",userServiceImpl.deleteUser(20));
}

@Test
void testUpdateUser() {
Matchmaker matchmaker = new Matchmaker(
6, "SubramaniyanKumran", "8899001123", "Gandhipuram,Coimbatore");



User user = new User(
19,
"NandhaKumaran",
"Groom","No-9,NanjuPuram,Virudhunagar",40,"IT-Tester","Monthly","Hindu","20000","Unmarried",matchmaker);
```

```
assertEquals("Success", userServiceImpl.updateUser(user));
}

@Test
void testUpdateUserIfNotPresentOrNUll() {
Matchmaker matchmaker = new Matchmaker(
6, "SubramaniyanKumaran", "8899001123", "Gandhipuram,Coimbatore");



User user = new User(
99,
"NandhuKumaran",
"Groom","No-9,NanjuPuram,Virudhunagar",40,"IT-Tester","Monthly","Hindu","20000","Unmarried",matchmaker);




assertEquals("Failed", userServiceImpl.updateUser(user));
}

@Test
@Order(2)
void testGetAllUsers() {
assertEquals(5, userServiceImpl.getAllUsers().size());
}



}
```

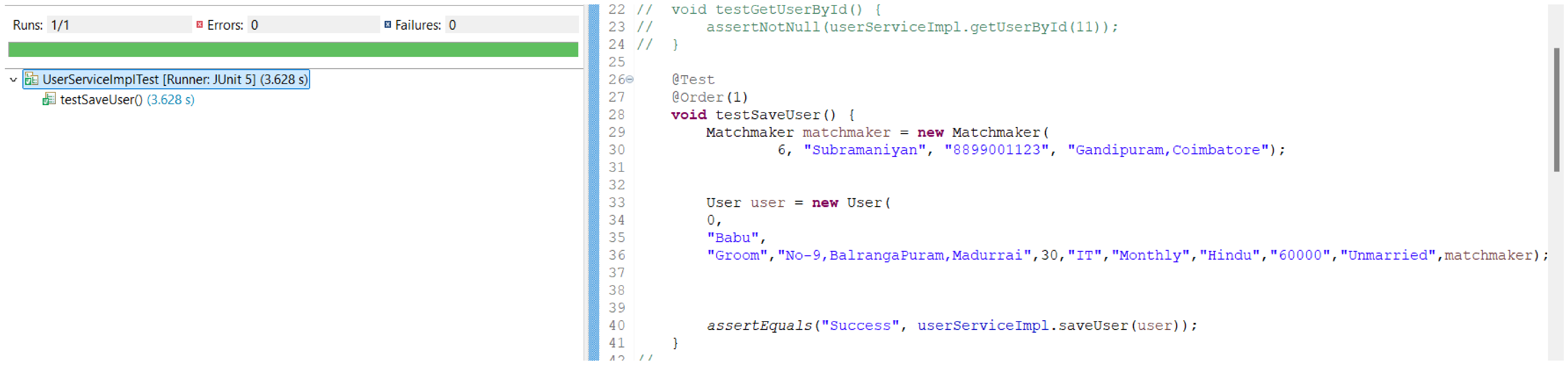
TEST SCREEN SHOTS :



```
Runs: 5/5        Errors: 0        Failures: 0

UserServiceImplTest [Runner: JUnit 5] (4.718 s)
    testDeleteUserIfNotPresentOrNull() (4.485 s)
    testGetAllUsers() (0.189 s)
    testUpdateUser() (0.032 s)
    testUpdateUserIfNotPresentOrNUll() (0.004 s)
    testGetUserById() (0.006 s)
```

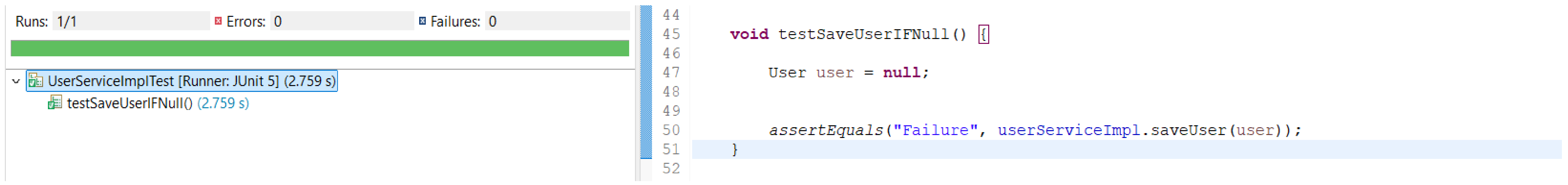


```
Runs: 2/2        Errors: 0        Failures: 0

UserServiceImplTest [Runner: JUnit 5] (3.508 s)
    testDeleteUserIfNotPresentOrNull() (3.488 s)
    testGetUserById() (0.020 s)
```



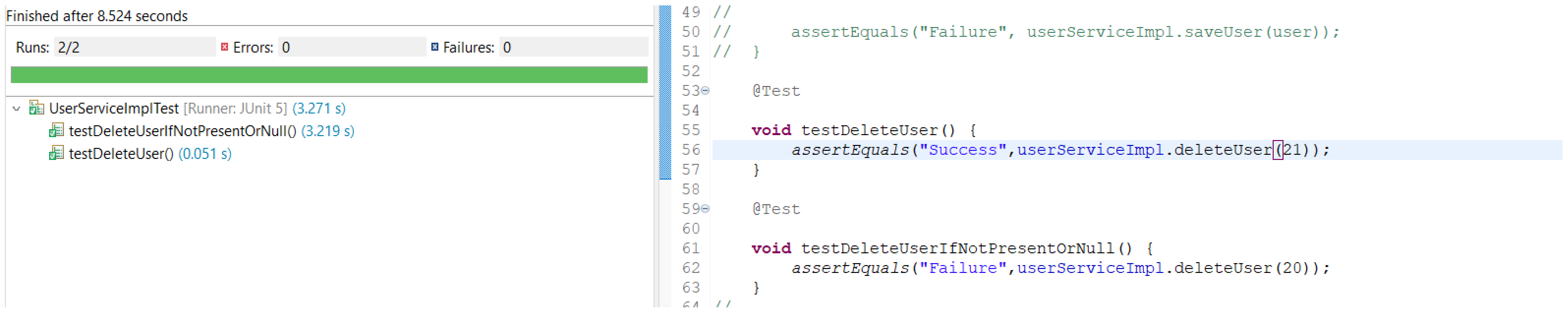
```
Runs: 1/1        Errors: 0        Failures: 0

UserServiceImplTest [Runner: JUnit 5] (3.628 s)
    testSaveUser() (3.628 s)
```

```java
//  void testGetUserById() {
//      assertNotNull(userServiceImpl.getUserById(11));
//  }

    @Test
    @Order(1)
    void testSaveUser() {
        Matchmaker matchmaker = new Matchmaker(
                6, "Subramaniyan", "8899001123", "Gandipuram,Coimbatore");


        User user = new User(
        0,
        "Babu",
        "Groom","No-9,BalrangaPuram,Madurrai",30,"IT","Monthly","Hindu","60000","Unmarried",matchmaker);



        assertEquals("Success", userServiceImpl.saveUser(user));
    }
```



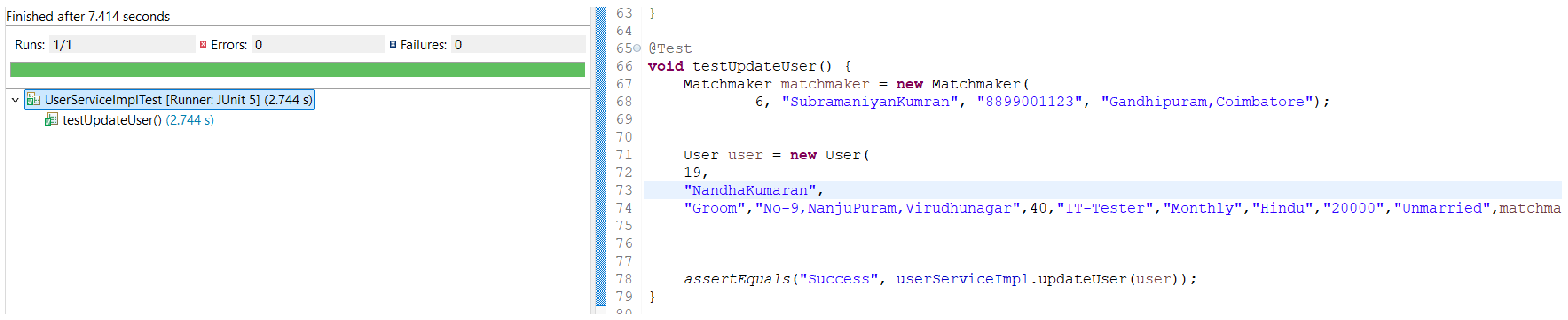
```
Runs: 1/1    Errors: 0    Failures: 0

UserServiceImplTest [Runner: JUnit 5] (2.759 s)
    testSaveUserIFNull() (2.759 s)
```

```java

    void testSaveUserIFNull() {

        User user = null;


        assertEquals("Failure", userServiceImpl.saveUser(user));
    }

```



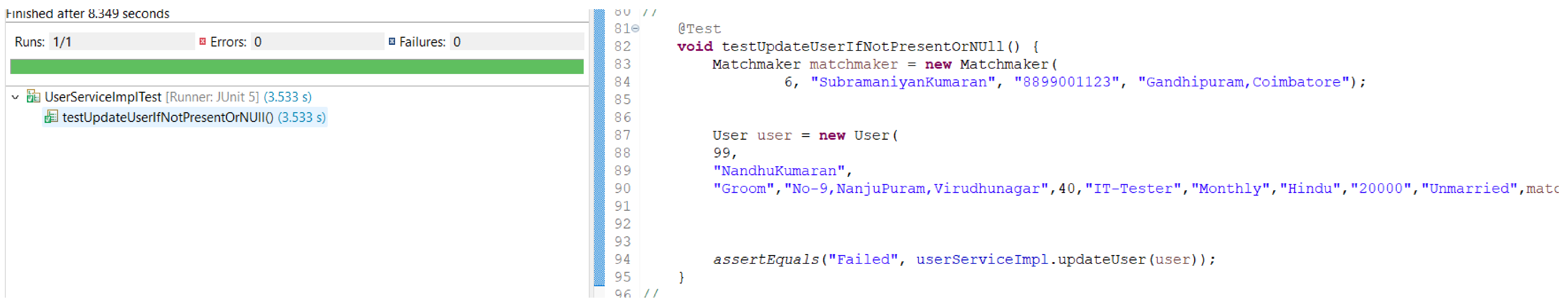
```
Finished after 8.524 seconds

Runs: 2/2    Errors: 0    Failures: 0

UserServiceImplTest [Runner: JUnit 5] (3.271 s)
    testDeleteUserIfNotPresentOrNull() (3.219 s)
    testDeleteUser() (0.051 s)
```

```java
//
//      assertEquals("Failure", userServiceImpl.saveUser(user));
//  }

    @Test

    void testDeleteUser() {
        assertEquals("Success",userServiceImpl.deleteUser(21));
    }

    @Test

    void testDeleteUserIfNotPresentOrNull() {
        assertEquals("Failure",userServiceImpl.deleteUser(20));
    }
```



```
Finished after 7.414 seconds

Runs: 1/1    Errors: 0    Failures: 0

UserServiceImplTest [Runner: JUnit 5] (2.744 s)
    testUpdateUser() (2.744 s)
```

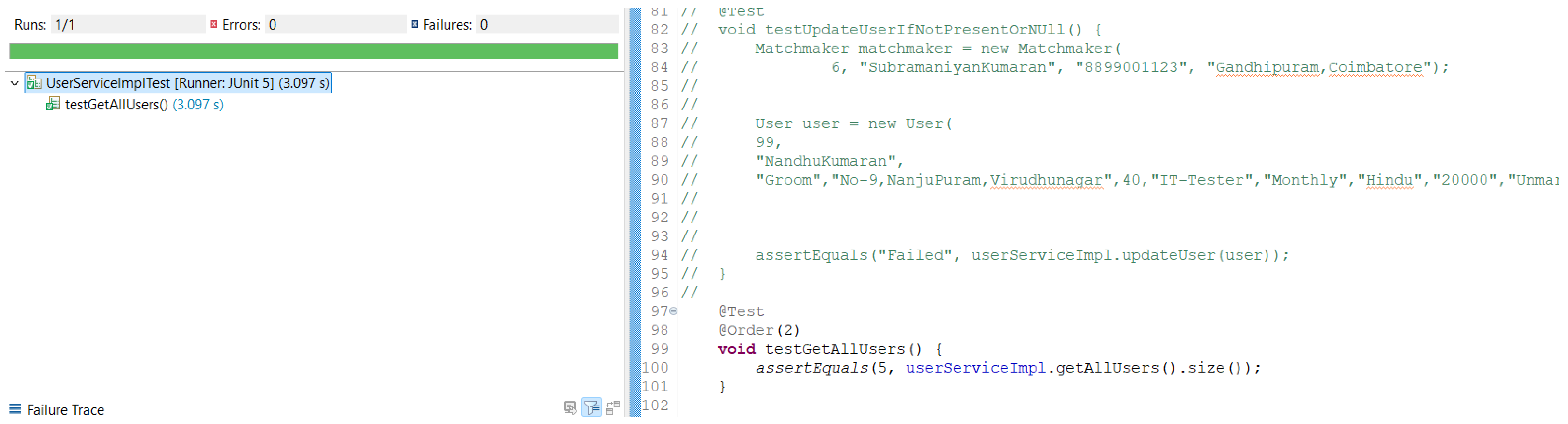
```java
}

@Test
void testUpdateUser() {
    Matchmaker matchmaker = new Matchmaker(
            6, "SubramaniyanKumran", "8899001123", "Gandhipuram,Coimbatore");


    User user = new User(
    19,
    "NandhaKumaran",
    "Groom","No-9,NanjuPuram,Virudhunagar",40,"IT-Tester","Monthly","Hindu","20000","Unmarried",matchma



    assertEquals("Success", userServiceImpl.updateUser(user));
}
```



Finished after 8.349 seconds

```
Runs: 1/1    Errors: 0    Failures: 0

UserServiceImplTest [Runner: JUnit 5] (3.533 s)
    testUpdateUserIfNotPresentOrNUll() (3.533 s)
```

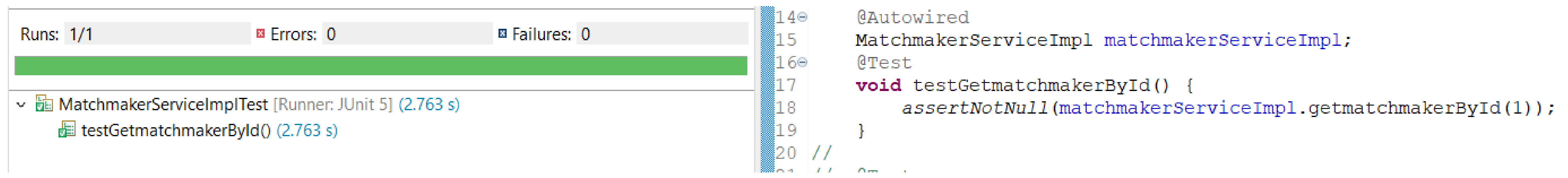
```java
//
    @Test
    void testUpdateUserIfNotPresentOrNUll() {
        Matchmaker matchmaker = new Matchmaker(
                6, "SubramaniyanKumaran", "8899001123", "Gandhipuram,Coimbatore");


        User user = new User(
        99,
        "NandhuKumaran",
        "Groom","No-9,NanjuPuram,Virudhunagar",40,"IT-Tester","Monthly","Hindu","20000","Unmarried",matc



        assertEquals("Failed", userServiceImpl.updateUser(user));
    }
//
```



```
Runs: 1/1    Errors: 0    Failures: 0

UserServiceImplTest [Runner: JUnit 5] (3.097 s)
    testGetAllUsers() (3.097 s)

Failure Trace
```

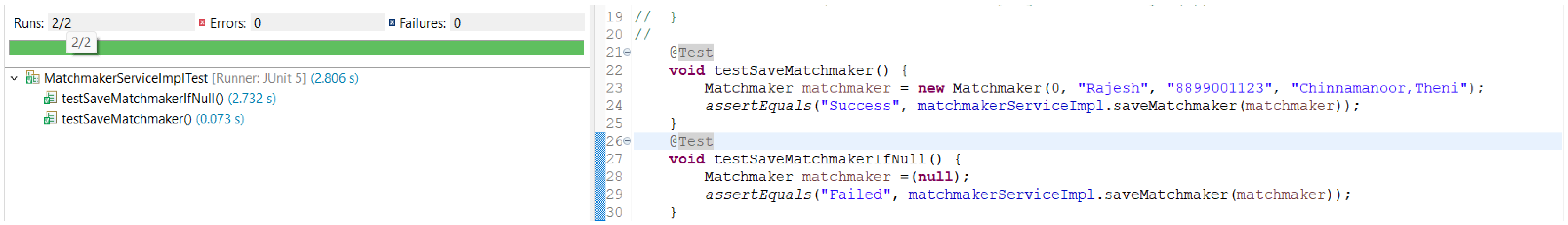
```java
//  @Test
//  void testUpdateUserIfNotPresentOrNUll() {
//      Matchmaker matchmaker = new Matchmaker(
//              6, "SubramaniyanKumaran", "8899001123", "Gandhipuram,Coimbatore");
//
//
//      User user = new User(
//      99,
//      "NandhuKumaran",
//      "Groom","No-9,NanjuPuram,Virudhunagar",40,"IT-Tester","Monthly","Hindu","20000","Unmar
//
//
//
//      assertEquals("Failed", userServiceImpl.updateUser(user));
//  }
//
    @Test
    @Order(2)
    void testGetAllUsers() {
        assertEquals(5, userServiceImpl.getAllUsers().size());
    }
    import static org.junit.jupiter.api.Assertions.*;
import static org.mockito.ArgumentMatchers.nullable;

import org.junit.jupiter.api.Order;
import org.junit.jupiter.api.Test;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.boot.test.context.SpringBootTest;

import com.hibernate.matrimonial.model.Matchmaker;
@SpringBootTest
class MatchmakerServiceImplTest {

@Autowired
MatchmakerServiceImpl matchmakerServiceImpl;
@Test
void testGetmatchmakerById() {
assertNotNull(matchmakerServiceImpl.getmatchmakerById(1));
}

@Test
void testSaveMatchmaker() {
Matchmaker matchmaker = new Matchmaker(0, "Rajesh", "8899001123", "Chinnamanoor,Theni");
assertEquals("Success", matchmakerServiceImpl.saveMatchmaker(matchmaker));
}
@Test
void testSaveMatchmakerIfNull() {
Matchmaker matchmaker =(null);
assertEquals("Failed", matchmakerServiceImpl.saveMatchmaker(matchmaker));
}

@Test
@Order(1)
void testDeleteMatchmaker() {
assertEquals("Success", matchmakerServiceImpl.deleteMatchmaker(8));
}
@Test
@Order(2)
void testDeleteMatchmakerIfNotPresent() {
assertEquals("Failure", matchmakerServiceImpl.deleteMatchmaker(8));
}

@Test
void testUpdateMatchmaker() {
Matchmaker matchmaker = new Matchmaker(6, "SubramaniyanKumaran", "8899001123", "Gandhipuram,Coimbatore");
assertEquals("Success",matchmakerServiceImpl.updateMatchmaker(matchmaker));
}

@Test
void testUpdateMatchmakerIfNullOrNotPresent() {
Matchmaker matchmaker = new Matchmaker(99,"naveen","8899","Theni");
assertEquals("Failed",matchmakerServiceImpl.updateMatchmaker(matchmaker));
}
////

@Test
void testGetAll() {
assertEquals(3, matchmakerServiceImpl.getAll().size());
}

}
```

TEST SCREENS

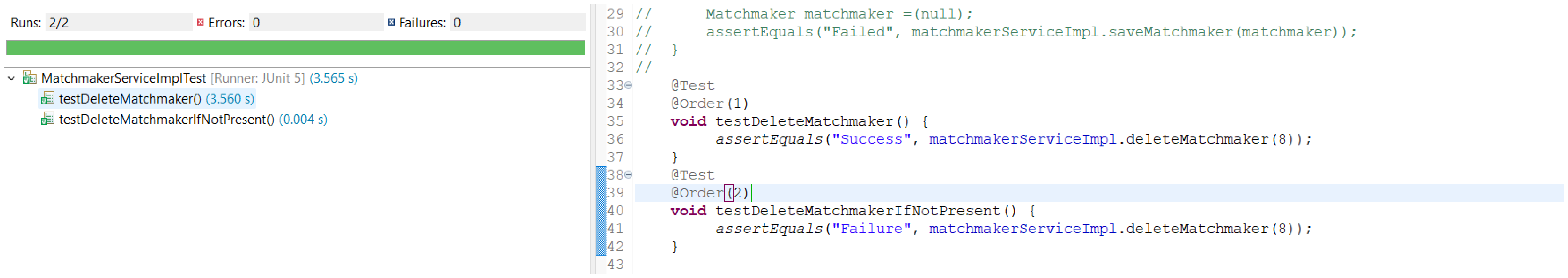


```
Runs: 1/1    Errors: 0    Failures: 0

MatchmakerServiceImplTest [Runner: JUnit 5] (2.763 s)
    testGetmatchmakerById() (2.763 s)
```

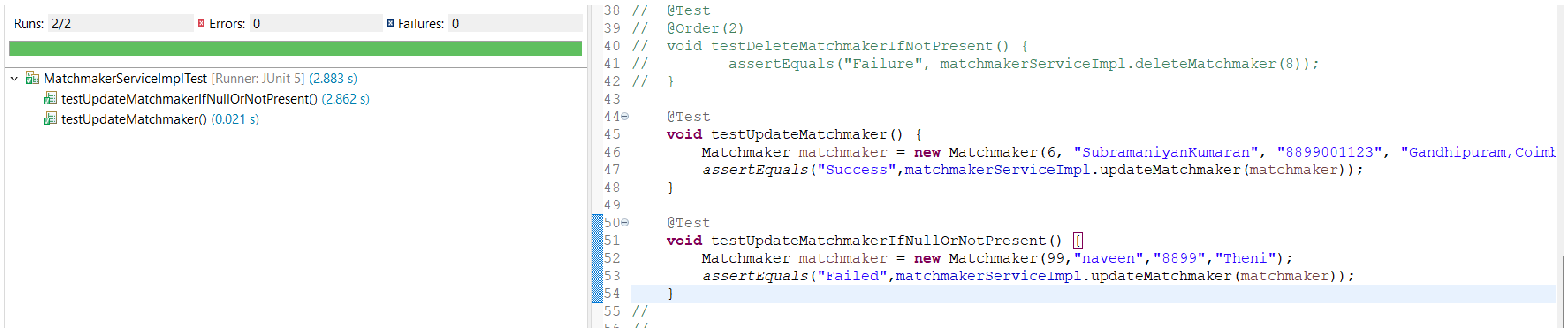
```java
    @Autowired
    MatchmakerServiceImpl matchmakerServiceImpl;
    @Test
    void testGetmatchmakerById() {
        assertNotNull(matchmakerServiceImpl.getmatchmakerById(1));
    }
//
```


```
Runs: 2/2        Errors: 0        Failures: 0

MatchmakerServiceImplTest [Runner: JUnit 5] (2.806 s)
    testSaveMatchmakerIfNull() (2.732 s)
    testSaveMatchmaker() (0.073 s)
```

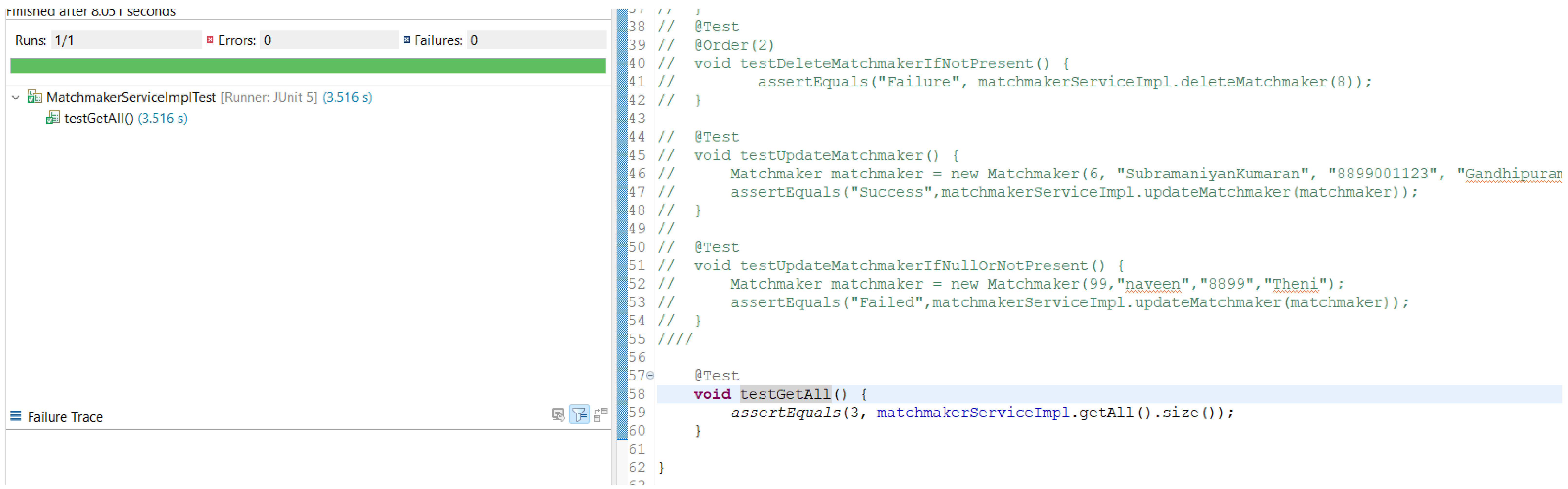
```java
//   }
//
    @Test
    void testSaveMatchmaker() {
        Matchmaker matchmaker = new Matchmaker(0, "Rajesh", "8899001123", "Chinnamanoor,Theni");
        assertEquals("Success", matchmakerServiceImpl.saveMatchmaker(matchmaker));
    }
    @Test
    void testSaveMatchmakerIfNull() {
        Matchmaker matchmaker =(null);
        assertEquals("Failed", matchmakerServiceImpl.saveMatchmaker(matchmaker));
    }
```


```
Runs: 2/2        Errors: 0        Failures: 0

MatchmakerServiceImplTest [Runner: JUnit 5] (3.565 s)
    testDeleteMatchmaker() (3.560 s)
    testDeleteMatchmakerIfNotPresent() (0.004 s)
```

```java
//      Matchmaker matchmaker =(null);
//      assertEquals("Failed", matchmakerServiceImpl.saveMatchmaker(matchmaker));
//  }
//
    @Test
    @Order(1)
    void testDeleteMatchmaker() {
         assertEquals("Success", matchmakerServiceImpl.deleteMatchmaker(8));
    }
    @Test
    @Order(2)
    void testDeleteMatchmakerIfNotPresent() {
         assertEquals("Failure", matchmakerServiceImpl.deleteMatchmaker(8));
    }

```


```
Runs: 2/2        Errors: 0        Failures: 0

MatchmakerServiceImplTest [Runner: JUnit 5] (2.883 s)
    testUpdateMatchmakerIfNullOrNotPresent() (2.862 s)
    testUpdateMatchmaker() (0.021 s)
```

```java
//  @Test
//  @Order(2)
//  void testDeleteMatchmakerIfNotPresent() {
//         assertEquals("Failure", matchmakerServiceImpl.deleteMatchmaker(8));
//  }

    @Test
    void testUpdateMatchmaker() {
        Matchmaker matchmaker = new Matchmaker(6, "SubramaniyanKumaran", "8899001123", "Gandhipuram,Coimb
        assertEquals("Success",matchmakerServiceImpl.updateMatchmaker(matchmaker));
    }

    @Test
    void testUpdateMatchmakerIfNullOrNotPresent() {
        Matchmaker matchmaker = new Matchmaker(99,"naveen","8899","Theni");
        assertEquals("Failed",matchmakerServiceImpl.updateMatchmaker(matchmaker));
    }
//
//
```



Finished after 8.051 seconds

Runs: 1/1   Errors: 0   Failures: 0

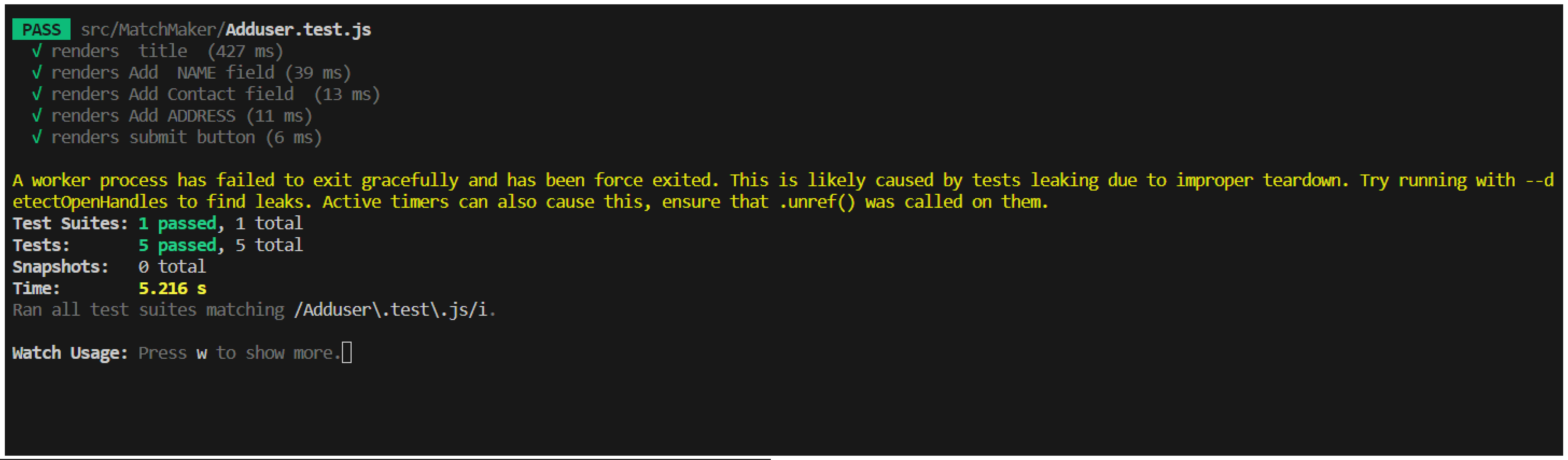
MatchmakerServiceImplTest [Runner: JUnit 5] (3.516 s)
testGetAll() (3.516 s)

Failure Trace

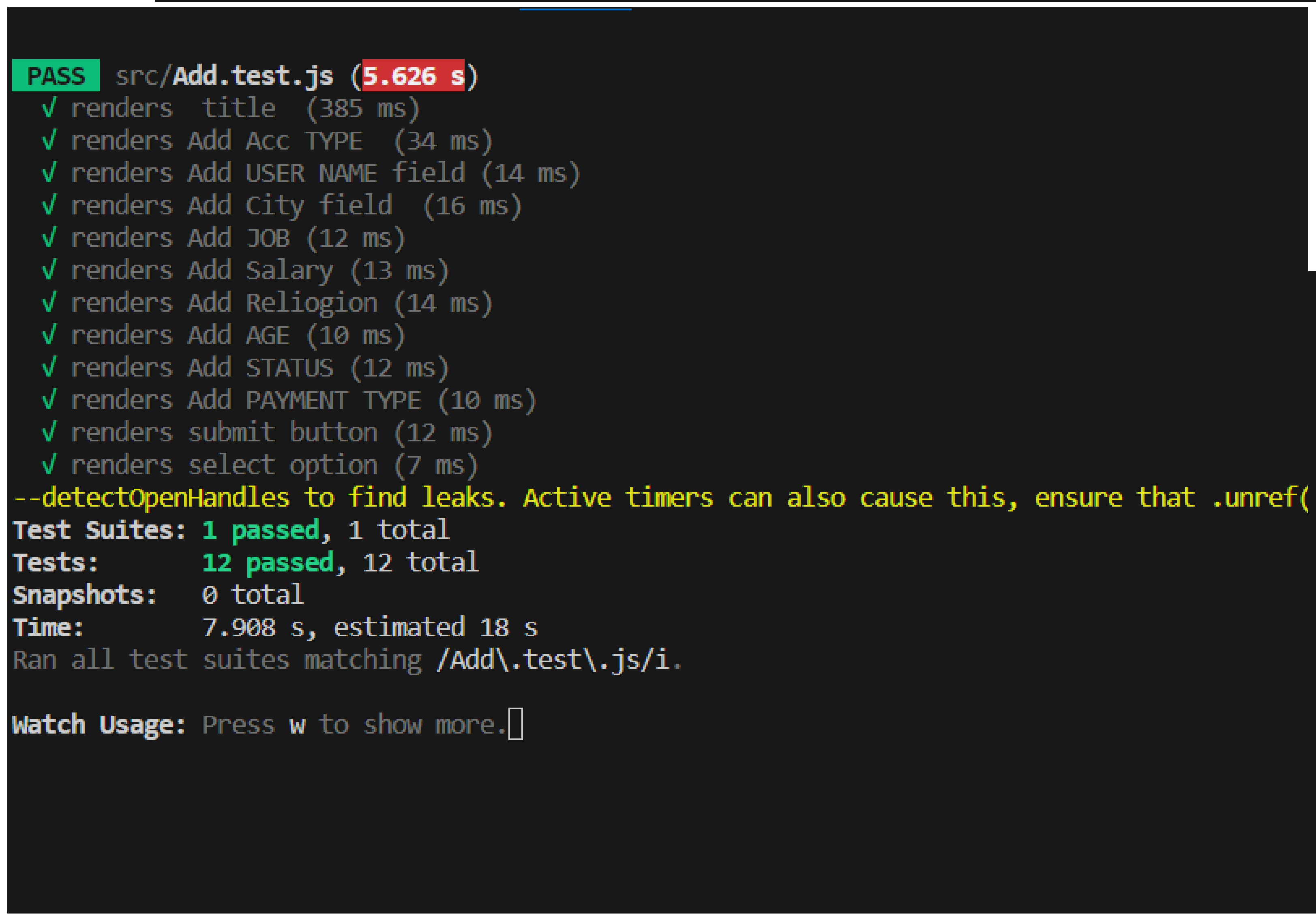
```
//  }
//  @Test
//  @Order(2)
//  void testDeleteMatchmakerIfNotPresent() {
//         assertEquals("Failure", matchmakerServiceImpl.deleteMatchmaker(8));
//  }

//  @Test
//  void testUpdateMatchmaker() {
//      Matchmaker matchmaker = new Matchmaker(6, "SubramaniyanKumaran", "8899001123", "Gandhipuram
//      assertEquals("Success",matchmakerServiceImpl.updateMatchmaker(matchmaker));
//  }
//
//  @Test
//  void testUpdateMatchmakerIfNullOrNotPresent() {
//      Matchmaker matchmaker = new Matchmaker(99,"naveen","8899","Theni");
//      assertEquals("Failed",matchmakerServiceImpl.updateMatchmaker(matchmaker));
//  }
////

    @Test
    void testGetAll() {
        assertEquals(3, matchmakerServiceImpl.getAll().size());
    }

}
```

FRONTEND JEST FILES

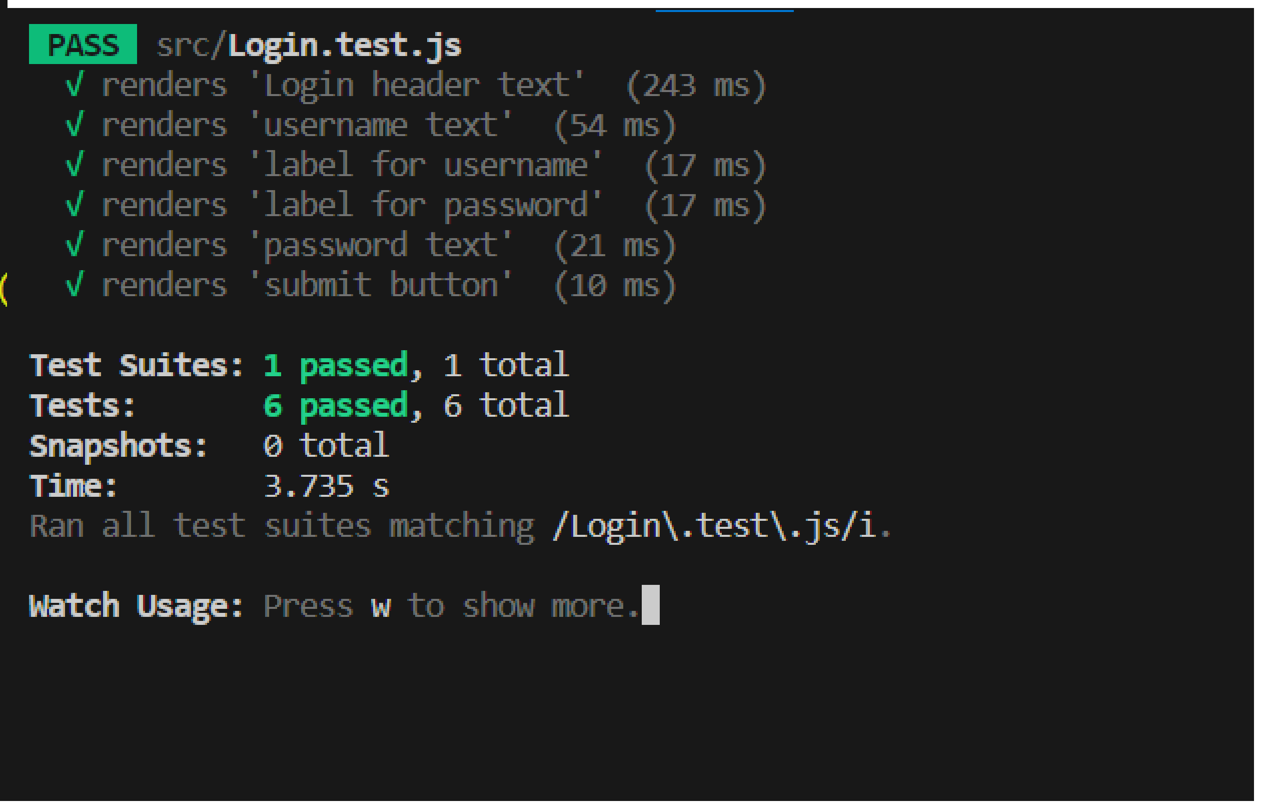
SCREENSHOTS



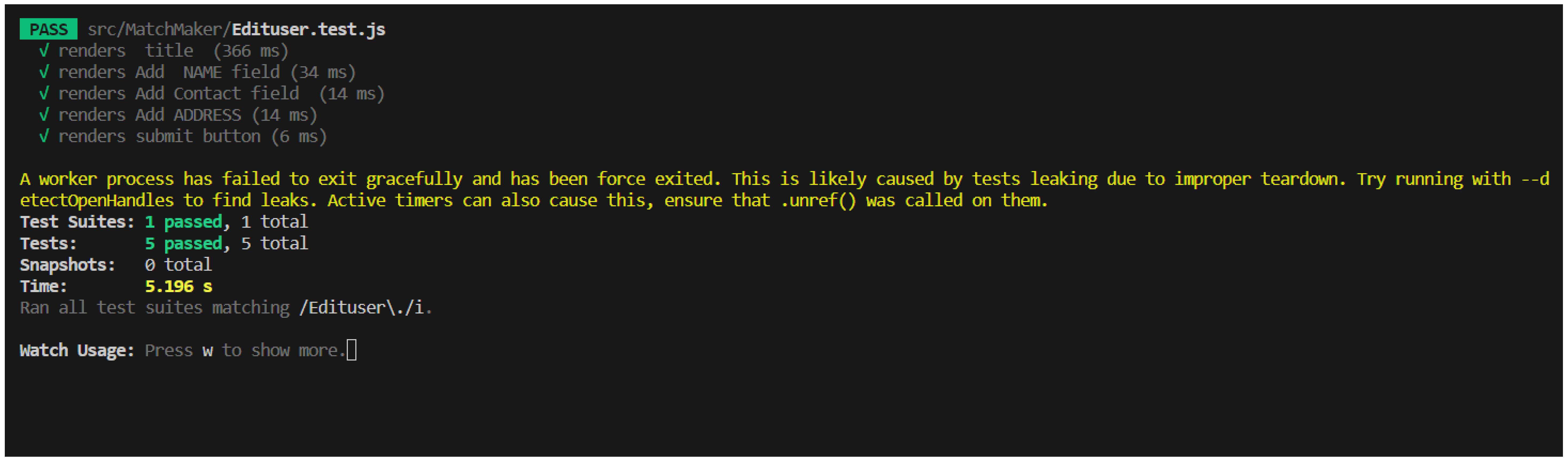
```
PASS  src/MatchMaker/Adduser.test.js
  √ renders  title  (427 ms)
  √ renders Add  NAME field (39 ms)
  √ renders Add Contact field  (13 ms)
  √ renders Add ADDRESS (11 ms)
  √ renders submit button (6 ms)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --d
etectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        5.216 s
Ran all test suites matching /Adduser\.test\.js/i.

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



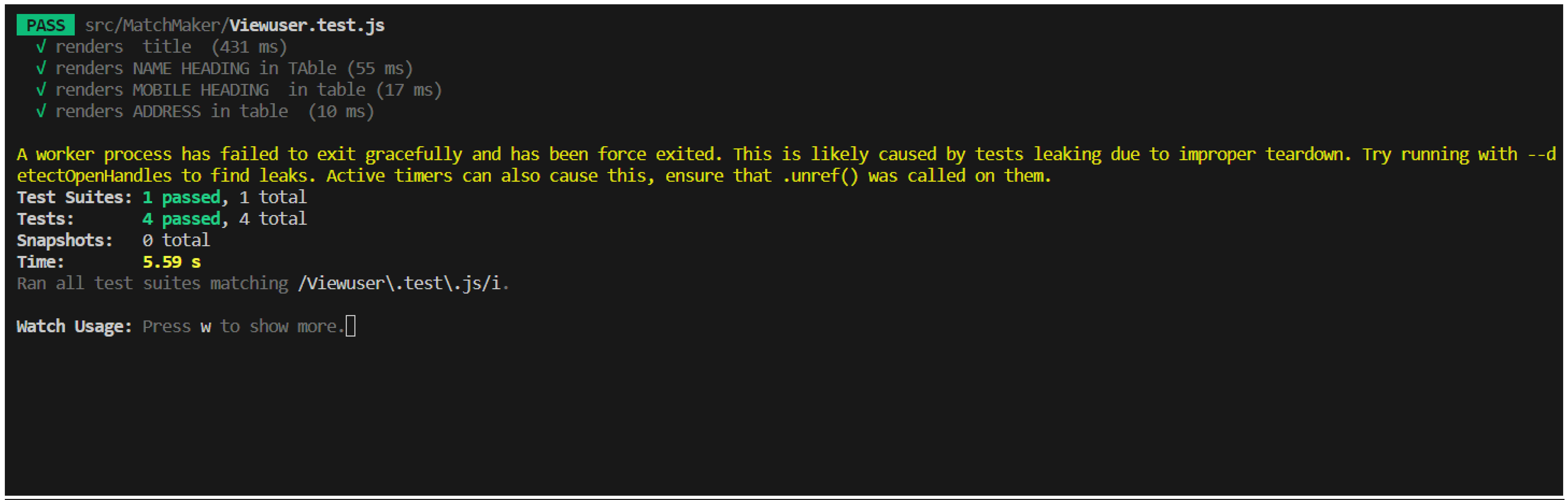
```
PASS  src/Add.test.js (5.626 s)
  √ renders  title  (385 ms)
  √ renders Add Acc TYPE  (34 ms)
  √ renders Add USER NAME field (14 ms)
  √ renders Add City field  (16 ms)
  √ renders Add JOB (12 ms)
  √ renders Add Salary (13 ms)
  √ renders Add Reliogion (14 ms)
  √ renders Add AGE (10 ms)
  √ renders Add STATUS (12 ms)
  √ renders Add PAYMENT TYPE (10 ms)
  √ renders submit button (12 ms)
  √ renders select option (7 ms)
--detectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref(
Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        7.908 s, estimated 18 s
Ran all test suites matching /Add\.test\.js/i.

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



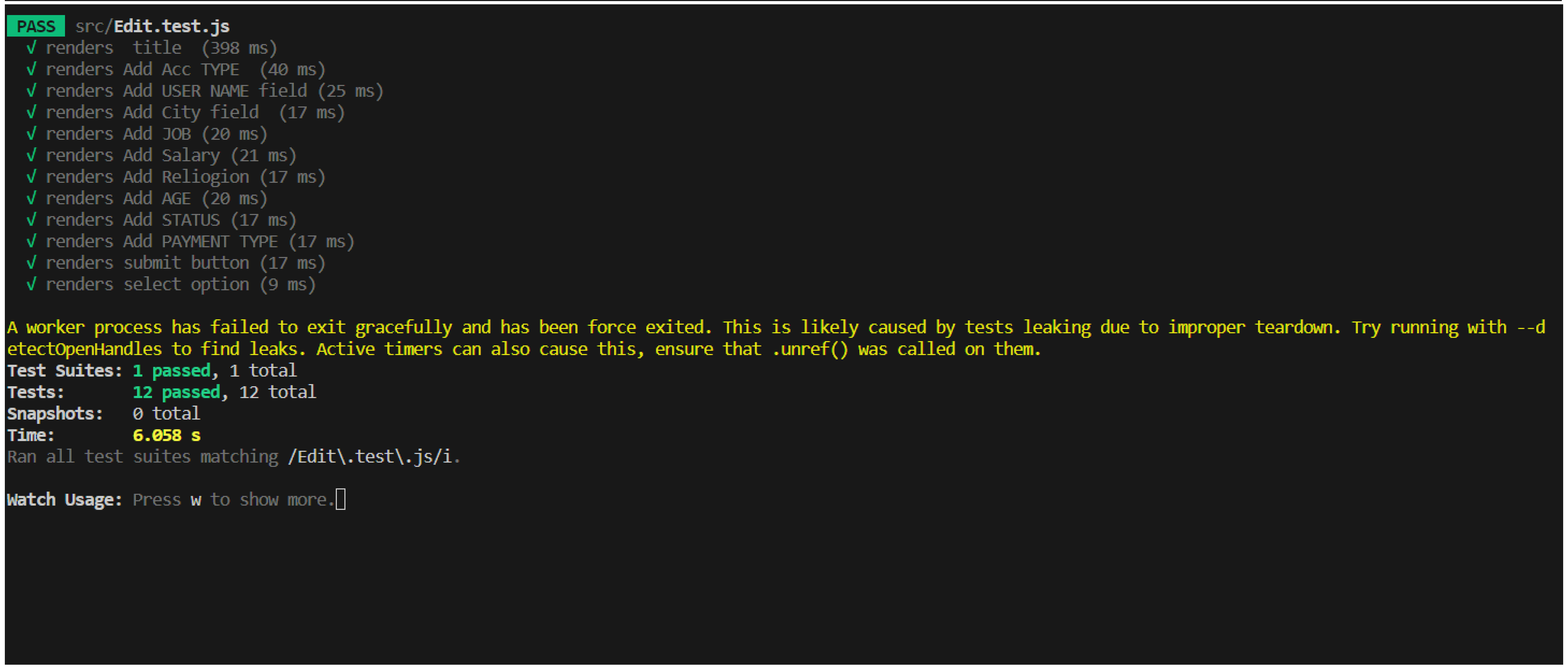
```
PASS  src/Login.test.js
  √ renders 'Login header text'  (243 ms)
  √ renders 'username text'  (54 ms)
  √ renders 'label for username'  (17 ms)
  √ renders 'label for password'  (17 ms)
  √ renders 'password text'  (21 ms)
  √ renders 'submit button'  (10 ms)

Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        3.735 s
Ran all test suites matching /Login\.test\.js/i.

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



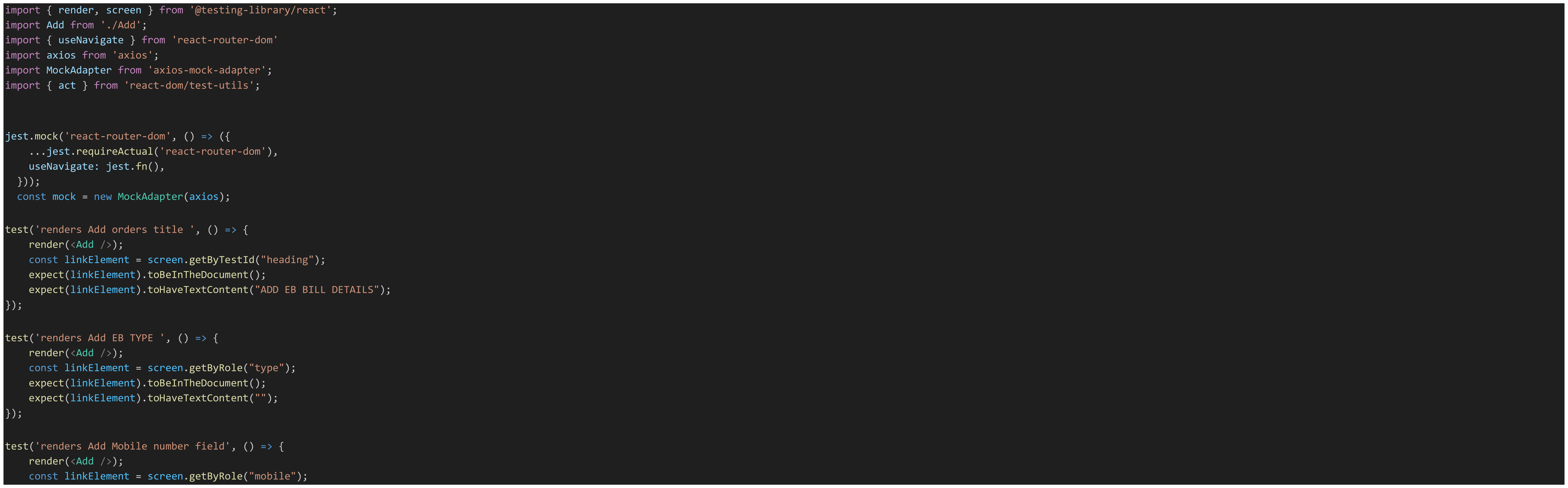
```
PASS  src/MatchMaker/Edituser.test.js
  √ renders  title  (366 ms)
  √ renders Add  NAME field (34 ms)
  √ renders Add Contact field  (14 ms)
  √ renders Add ADDRESS (14 ms)
  √ renders submit button (6 ms)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --d
etectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        5.196 s
Ran all test suites matching /Edituser\./i.

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



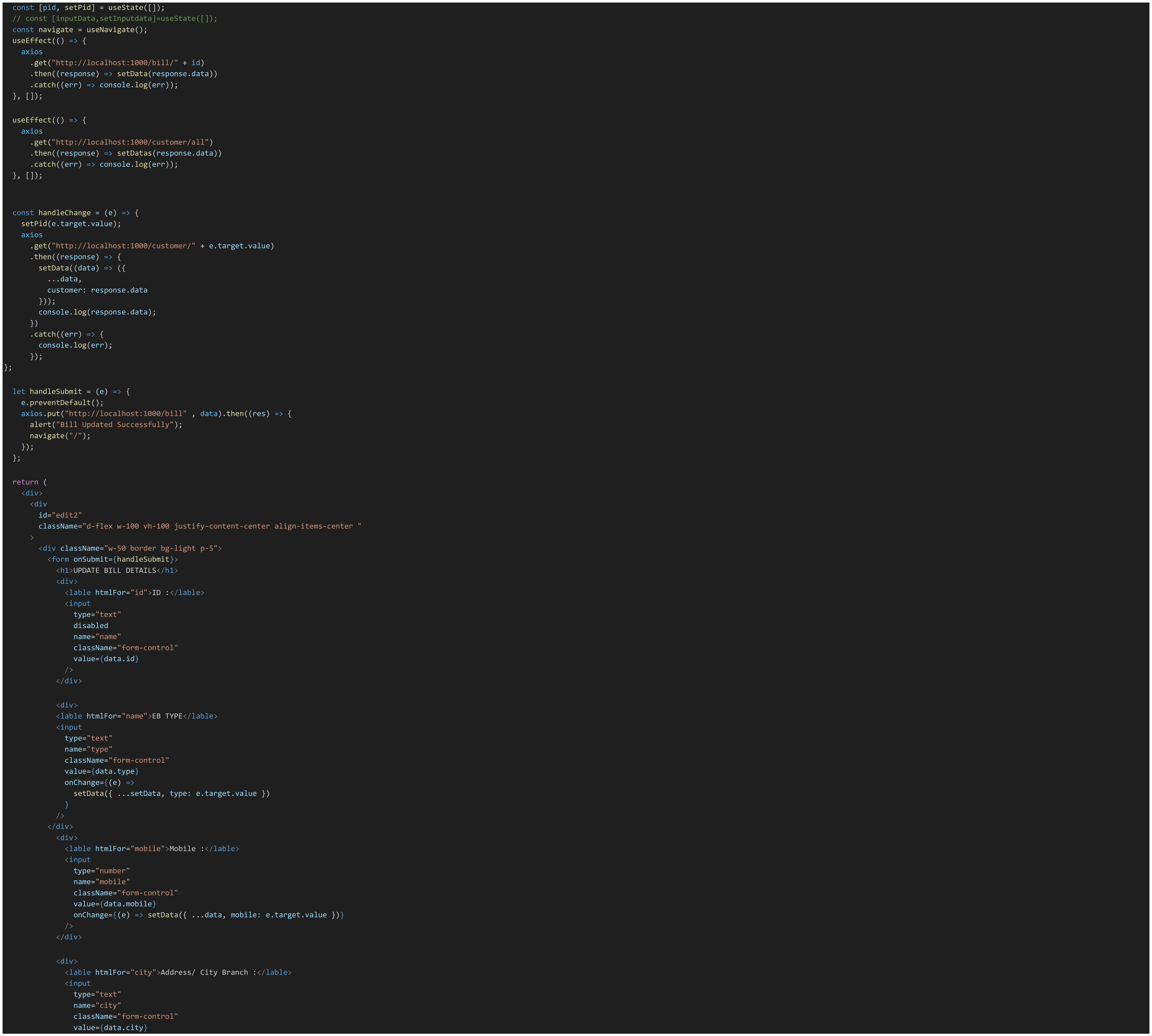
```
 PASS  src/MatchMaker/Viewuser.test.js
  √ renders  title  (431 ms)
  √ renders NAME HEADING in TAble (55 ms)
  √ renders MOBILE HEADING  in table (17 ms)
  √ renders ADDRESS in table  (10 ms)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --d
etectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
Test Suites: 1 passed, 1 total
Tests:       4 passed, 4 total
Snapshots:   0 total
Time:        5.59 s
Ran all test suites matching /Viewuser\.test\.js/i.

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



```
 PASS  src/Edit.test.js
  √ renders  title  (398 ms)
  √ renders Add Acc TYPE  (40 ms)
  √ renders Add USER NAME field (25 ms)
  √ renders Add City field  (17 ms)
  √ renders Add JOB (20 ms)
  √ renders Add Salary (21 ms)
  √ renders Add Reliogion (17 ms)
  √ renders Add AGE (20 ms)
  √ renders Add STATUS (17 ms)
  √ renders Add PAYMENT TYPE (17 ms)
  √ renders submit button (17 ms)
  √ renders select option (9 ms)

A worker process has failed to exit gracefully and has been force exited. This is likely caused by tests leaking due to improper teardown. Try running with --d
etectOpenHandles to find leaks. Active timers can also cause this, ensure that .unref() was called on them.
Test Suites: 1 passed, 1 total
Tests:       12 passed, 12 total
Snapshots:   0 total
Time:        6.058 s
Ran all test suites matching /Edit\.test\.js/i.

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



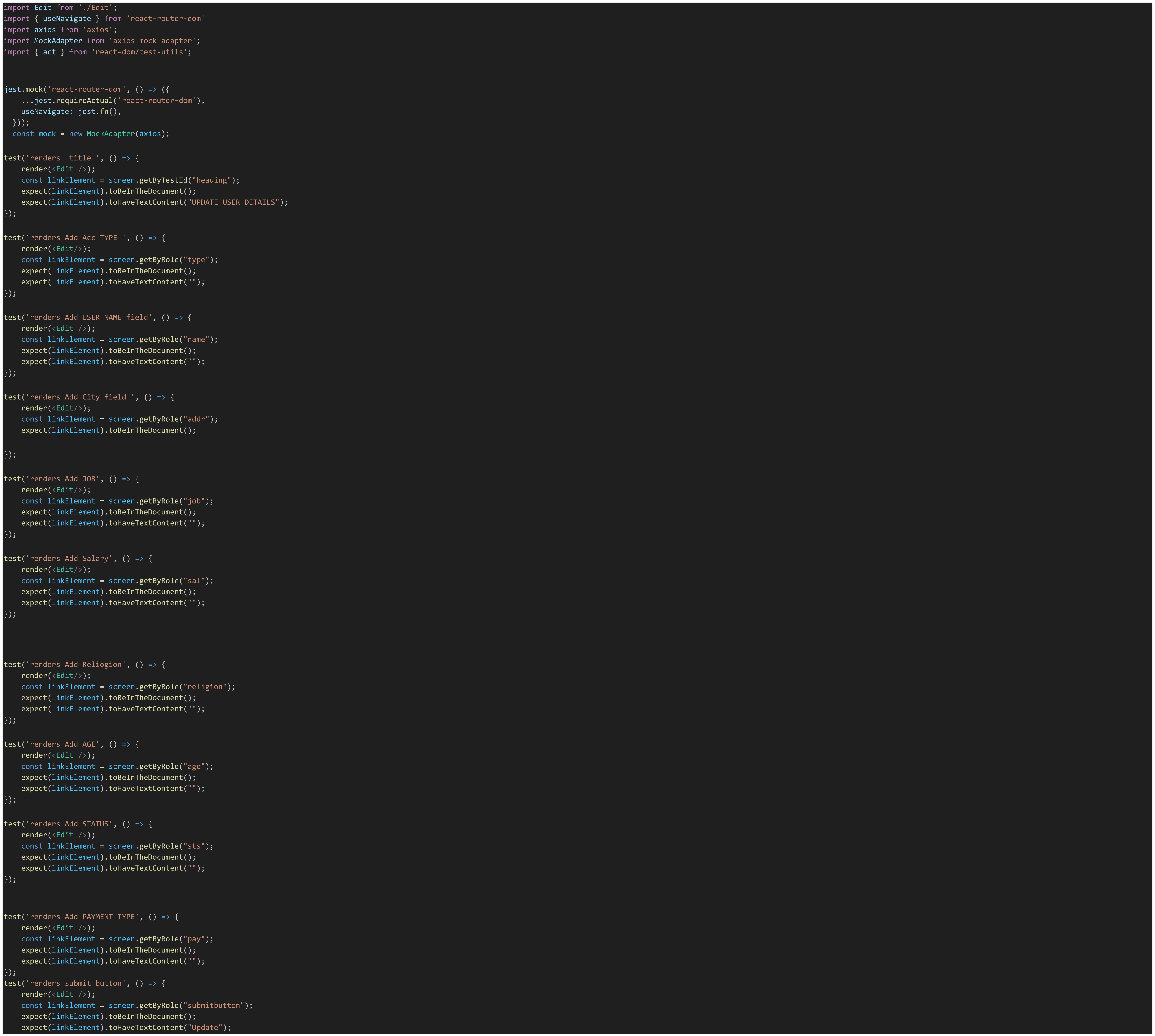
```
import { render, screen } from '@testing-library/react';
import Add from './Add';
import { useNavigate } from 'react-router-dom'
import axios from 'axios';
import MockAdapter from 'axios-mock-adapter';
import { act } from 'react-dom/test-utils';


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

test('renders Add orders title ', () => {
    render(<Add />);
    const linkElement = screen.getByTestId("heading");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("ADD EB BILL DETAILS");
});

test('renders Add EB TYPE ', () => {
    render(<Add />);
    const linkElement = screen.getByRole("type");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add Mobile number field', () => {
    render(<Add />);
    const linkElement = screen.getByRole("mobile");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add City field ', () => {
    render(<Add />);
    const linkElement = screen.getByRole("city");
    expect(linkElement).toBeInTheDocument();

});

test('renders Add Due Date Field', () => {
    render(<Add />);
    const linkElement = screen.getByRole("date");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add Price per units', () => {
    render(<Add />);
    const linkElement = screen.getByRole("price");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});


test('renders Add Bill Field', () => {
    render(<Add />);
    const linkElement = screen.getByRole("bill");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});


test('renders submit button', () => {
    render(<Add />);
    const linkElement = screen.getByRole("submitbutton");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Submit");
});


test('renders select option', () => {
    render(<Add />);

    const selectedOption = screen.getByTestId('opt');
    expect(selectedOption).toBeInTheDocument();
});

import Signin from "./Signin";
import { render, screen } from '@testing-library/react';

jest.mock('axios');
jest.mock('react-router-dom');

it("renders 'username text' ", () => {
    render(<Signin />);
    const linkElement = screen.getByRole("username");
    expect(linkElement).toBeInTheDocument();



  });

  it("renders 'label for password' ", () => {
    render(<Signin />);
    const linkElement = screen.getByRole("pwdlabel");
    expect(linkElement).toBeInTheDocument();

  });
  it("renders 'password text' ", () => {
    render(<Signin />);
    const linkElement = screen.getByRole("pwdtext");
    expect(linkElement).toBeInTheDocument();
    expect(screen.getByPlaceholderText("Enter Password")).toHaveTextContent("");

  });
  it("renders 'submit button' ", () => {
    render(<Signin/>);
    const linkElement = screen.getByTestId("login-submit");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Login");

  });



function Edit() {
  const { id } = useParams();
  const [data, setData] = useState([]);
  const [datas, setDatas] = useState([]);
```



```
  const [pid, setPid] = useState([]);
  // const [inputData,setInputdata]=useState([]);
  const navigate = useNavigate();
  useEffect(() => {
    axios
      .get("http://localhost:1000/bill/" + id)
      .then((response) => setData(response.data))
      .catch((err) => console.log(err));
  }, []);

  useEffect(() => {
    axios
      .get("http://localhost:1000/customer/all")
      .then((response) => setDatas(response.data))
      .catch((err) => console.log(err));
  }, []);


  const handleChange = (e) => {
    setPid(e.target.value);
    axios
      .get("http://localhost:1000/customer/" + e.target.value)
      .then((response) => {
        setData((data) => ({
          ...data,
          customer: response.data
        }));
        console.log(response.data);
      })
      .catch((err) => {
        console.log(err);
      });
};

  let handleSubmit = (e) => {
    e.preventDefault();
    axios.put("http://localhost:1000/bill" , data).then((res) => {
      alert("Bill Updated Successfully");
      navigate("/");
    });
  };

  return (
    <div>
      <div
        id="edit2"
        className="d-flex w-100 vh-100 justify-content-center align-items-center "
      >
        <div className="w-50 border bg-light p-5">
          <form onSubmit={handleSubmit}>
            <h1>UPDATE BILL DETAILS</h1>
            <div>
              <lable htmlFor="id">ID :</lable>
              <input
                type="text"
                disabled
                name="name"
                className="form-control"
                value={data.id}
              />
            </div>

            <div>
            <lable htmlFor="name">EB TYPE</lable>
            <input
              type="text"
              name="type"
              className="form-control"
              value={data.type}
              onChange={(e) =>
                setData({ ...setData, type: e.target.value })
              }
            />
          </div>
            <div>
              <lable htmlFor="mobile">Mobile :</lable>
              <input
                type="number"
                name="mobile"
                className="form-control"
                value={data.mobile}
                onChange={(e) => setData({ ...data, mobile: e.target.value })}
              />
            </div>

            <div>
              <lable htmlFor="city">Address/ City Branch :</lable>
              <input
                type="text"
                name="city"
                className="form-control"
                value={data.city}
                onChange={(e) => setData({ ...data, city: e.target.value })}
              />
            </div>


            <div>
            <lable htmlFor="desg">Bill Due Date :</lable>
            <input
              type="text"
              name="desg"
              className="form-control"
              value={data.date}
              onChange={(e) =>
                setData({ ...data, date: e.target.value })
              }
            />
          </div>


          <div>
            <lable htmlFor="nos">Consumed Units :</lable>
            <input
              type="number"
              name="nos"
              className="form-control"
              value={data.nos}
              onChange={(e) =>
                setData({ ...data, nos: e.target.value })
              }
            />
          </div>

          <div>
            <lable htmlFor="price">Price Per Unit :</lable>
            <input
              type="number"
              name="price"
              className="form-control"
              value={data.price}
              onChange={(e) =>
                setData({ ...data, price: e.target.value })
              }
            />
          </div>
          {/* <div>
            <label htmlFor="price" >Bill </label>
            <input
             type="number"
             name=""
          </div> */}
          <div>
            <lable htmlFor="totalBill">Bill Amount :</lable>
            <input
              type="number"
              name="totalBill"
              className="form-control"
              value={data.totalBill}
              onBlur={(e) =>
                setData({ ...data, totalBill:  parseInt(data.price)*parseInt(data.nos)})
              }
            />
          </div>

            <br />

            <button className="btn btn-info ">Update</button>
          </form>
        </div>
      </div>
    </div>
  );
}

export default Edit;




jest.mock('axios');
jest.mock('react-router-dom');


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



});

it("renders 'label for password' ", () => {
  render(<Login />);
  const linkElement = screen.getByRole("pwdlabel");
  expect(linkElement).toBeInTheDocument();

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

});




it("renders 'submit button' ", () => {
    render(<Login/>);
    const linkElement = screen.getByTestId("login-submit");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Login");

  });
```

```
import { render, screen } from '@testing-library/react';
import Edituser from './Edituser';
import { useNavigate } from 'react-router-dom'
import axios from 'axios';
import MockAdapter from 'axios-mock-adapter';
import { act } from 'react-dom/test-utils';


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

test('renders  title ', () => {
    render(<Edituser/>);
    const linkElement = screen.getByTestId("heading");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("UPDATE MATCH MAKER DATA'S");
});




test('renders Add  NAME field', () => {
    render(<Edituser />);
    const linkElement = screen.getByRole("name");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add Contact field ', () => {
    render(<Edituser />);
    const linkElement = screen.getByRole("mobile");
    expect(linkElement).toBeInTheDocument();

});




test('renders Add ADDRESS', () => {
    render(<Edituser />);
    const linkElement = screen.getByRole("addr");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});


test('renders submit button', () => {
    render(<Edituser />);
    const linkElement = screen.getByTestId("submitbutton");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Update");
});
```

```
mport { render, screen } from '@testing-library/react';
```



```
import Edit from './Edit';
import { useNavigate } from 'react-router-dom'
import axios from 'axios';
import MockAdapter from 'axios-mock-adapter';
import { act } from 'react-dom/test-utils';


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

test('renders  title ', () => {
    render(<Edit />);
    const linkElement = screen.getByTestId("heading");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("UPDATE USER DETAILS");
});

test('renders Add Acc TYPE ', () => {
    render(<Edit/>);
    const linkElement = screen.getByRole("type");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add USER NAME field', () => {
    render(<Edit />);
    const linkElement = screen.getByRole("name");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add City field ', () => {
    render(<Edit/>);
    const linkElement = screen.getByRole("addr");
    expect(linkElement).toBeInTheDocument();

});

test('renders Add JOB', () => {
    render(<Edit/>);
    const linkElement = screen.getByRole("job");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add Salary', () => {
    render(<Edit/>);
    const linkElement = screen.getByRole("sal");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});



test('renders Add Reliogion', () => {
    render(<Edit/>);
    const linkElement = screen.getByRole("religion");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add AGE', () => {
    render(<Edit />);
    const linkElement = screen.getByRole("age");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});

test('renders Add STATUS', () => {
    render(<Edit />);
    const linkElement = screen.getByRole("sts");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});


test('renders Add PAYMENT TYPE', () => {
    render(<Edit />);
    const linkElement = screen.getByRole("pay");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("");
});
test('renders submit button', () => {
    render(<Edit />);
    const linkElement = screen.getByRole("submitbutton");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent("Update");
});


test('renders select option', () => {
    render(<Edit />);

    const selectedOption = screen.getByTestId('opt');
    expect(selectedOption).toBeInTheDocument();
});
```