Junit and Jest Assessment

Online Taxi booking System

W3H Analysis for front-end unit testing:

WHAT	HOW
1. **What are the features to be tested?** **CAB COMPONENT:** • The application's title has to be tested. • Pickup location search bar has to be tested. • Drop location search bar has to be tested. • Booking page navigation button has to be tested. • Customer page navigation button has to be tested. • View button has to be tested. • Book button has to be tested. • Back button has to be tested. **CUSTOMER COMPONENT:** • Customer Id search bar has to be tested. • Booking page navigation button has to be tested. • Cab page navigation button has to be tested. • View button has to be tested. • Back button has to be tested. **BOOKING COMPONENT:** • Booking Id search bar has to be tested. • Customer page navigation button has to be tested. • Cab page navigation button has to be tested. • View button has to be tested. • Back button has to be tested. 2. **What is the tool used for unit testing?** **Jest** is the tool which is used for backend unit testing.	1. **How are the features to be tested?** **CAB COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** **CUSTOMER COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** **BOOKING COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** 2. **How is the tool used for testing?** **Method 1: Jest is the JavaScript testing framework which is used for front-end unit testing.**

WHY	WHY NOT
Why are these features used for testing? **CAB COMPONENT:** • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** Because, based on the requirement of testing, choosing the method is fine and useful. **CUSTOMER COMPONENT:** • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** Because, based on the requirement of testing, choosing the method is fine and useful. **BOOOKING COMPONENT:** • **Method 3: By using all the methods as per requirement and comfort we can execute the test cases.** Because, based on the requirement of testing, choosing the method is fine and useful.	**Why are these features not used for testing?** **CAB COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. **Because, using some of the selected methods for execution of test will not be helpful.** **CUSTOMER COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. **Because, using some of the selected methods for execution of test will not be helpful.** **BOOKING COMPONENT:** • Method 1: By using get by title method we can execute the test cases. • Method 2: By using get by title, get by role, get by id we can execute the test cases. **Because, using some of the selected methods for execution of test will not not be helpful.**

W3H Analysis for back-end unit testing:

WHAT	HOW
1. What are the features to be tested? **CAB:** • Getting all details from the database i.e. findAllcab method has to be tested. • Searching for pickup location has to be tested. • Searching for drop location has to be tested. • Searching for cabs i.e. FindCabById method has to be tested. **CUSTOMER:** • Getting all details from the database i.e. findAllcustomer method has to be tested. • Searching for customers based on id i.e. findCustomerById method has to be tested. **BOOKING:** • Getting all details from the database i.e. findAllBooking method has to be tested. • Searching for booking based on id i.e. findBookingById method has to be tested. • Booking of cab i.e. addBooking method has to be tested. **2. What is the tool used for unit testing?** **Ans: Junit** is the tool which is used for backend unit testing.	**3. How are the features to be tested?** **CAB:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • **Method 2: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** • Method 3: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. **CUSTOMER:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • Method 2: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. • **Method 3: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** **BOOKING:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • **Method 2: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** • Method 3: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. **4. How is the tool useful for testing?** **Method 2: Junit is used for backend unit testing.**

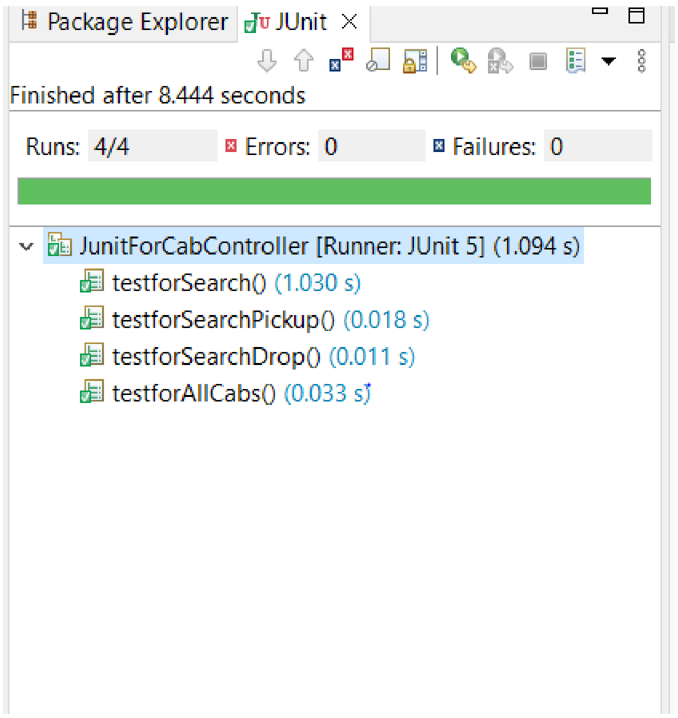
WHY	WHY NOT

Why are these features used for testing? **CAB:** • **Method 2: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** Because as per user stories, get all details of cab, find a cab by its id, search by pickup location and search cab by drop location must be tested. assertNotNull method which tests the reference variable of a controller has a value or not. For that, this method is more than enough to test. **CUSTOMER:** • **Method 3: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** Because as per user stories, get all details of customer and find a customer by their id must be tested. assertNotNull method which tests the reference variable of a controller has a value or not. For that, this method is more than enough to test. **BOOKING:** • **Method 2: Inject the controller class and using the reference of controller, call the according method and using assertNotNull method and execute the test case.** Because as per user stories, get all details of booking, add booking method and find a booking by id must be tested. assertNotNull method which tests the reference variable of a controller has a value or not. For that, this method is more than enough to test. **2. Why is this tool used for testing?** **Method 2: Junit is good for unit testing.** **It is best for backend unit testing and test report is easy to access.**	**Why are these features not used for testing?** **CAB:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • Method 3: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. **Because assertNull method is used to check whether the tested method is null. Also, assertEquals method is used to compare the values.** **So, it will not be appropriate for testing these features.** **CUSTOMER:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • Method 2: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. **Because assertNull method is used to check whether the tested method is null. Also, assertEquals method is used to compare the values.** **So, it will not be appropriate for testing these features.** **BOOKING:** • Method 1: Inject the controller class and using the reference of controller, call the according method and using assertNull method and execute the test case. • Method 3: Inject the controller class and using the reference of controller, call the according method and using assertEquals method and execute the test case. **Because assertNull method is used to check whether the tested method is null. Also, assertEquals method is used to compare the values.** **So, it will not be appropriate for testing these features.**

TEST DOCUMENT FOR JUNIT:

For cab controller:

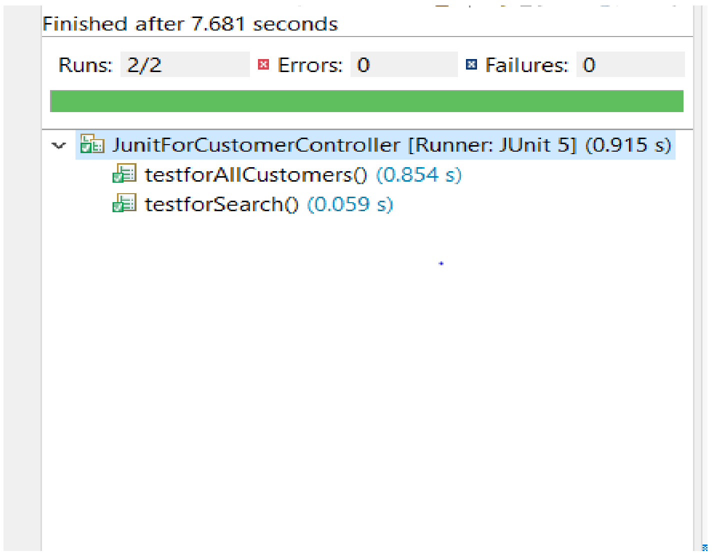
```
package com.taxi;

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.taxi.controller.CabController;
import com.taxi.model.Cab;

@SpringBootTest
class JunitForCabController {

@Autowired
private CabController cabController;

@Test
void testforSearch() {
Cab cab=new Cab();
cab=cabController.findByCabId(2);
assertNotNull(cab);

}

@Test
void testforSearchPickup() {

Cab cab=new Cab();

cab=cabController.findCabPickup("Neyveli");

assertNotNull(cab);



}


@Test

void testforSearchDrop() {

Cab cab=new Cab();

cab=cabController.findCabDrop("Chennai");

assertNotNull(cab);


}

@Test

void testforAllCabs() {

List<Cab>cab=cabController.getAllCab();

assertNotNull(cab);

}



}
```



```
Finished after 8.444 seconds

Runs: 4/4    Errors: 0    Failures: 0

JunitForCabController [Runner: JUnit 5] (1.094 s)
    testforSearch() (1.030 s)
    testforSearchPickup() (0.018 s)
    testforSearchDrop() (0.011 s)
    testforAllCabs() (0.033 s)
```

For Customer controller:

```java
package com.taxi;

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

import java.util.List;

import org.junit.jupiter.api.Test;

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

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

import com.taxi.controller.CustomerController;

import com.taxi.model.Customer;

@SpringBootTest

class JunitForCustomerController {


@Autowired

private CustomerController customerController;

@Test

void testforSearch() {

Customer customer=new Customer();

customer=customerController.findCustomerId(3);

assertEquals(customer.getCustomerName(), "Srinivasan");;

}


@Test
void testforAllCustomers() {

List<Customer> customer=customerController.getAllcustomers();

assertNotNull(customer);

}

}
```



```
Finished after 7.681 seconds
Runs: 2/2   Errors: 0   Failures: 0

JunitForCustomerController [Runner: JUnit 5] (0.915 s)
    testforAllCustomers() (0.854 s)
    testforSearch() (0.059 s)
```

For booking cntroller:

package com.taxi;

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

import java.util.List;

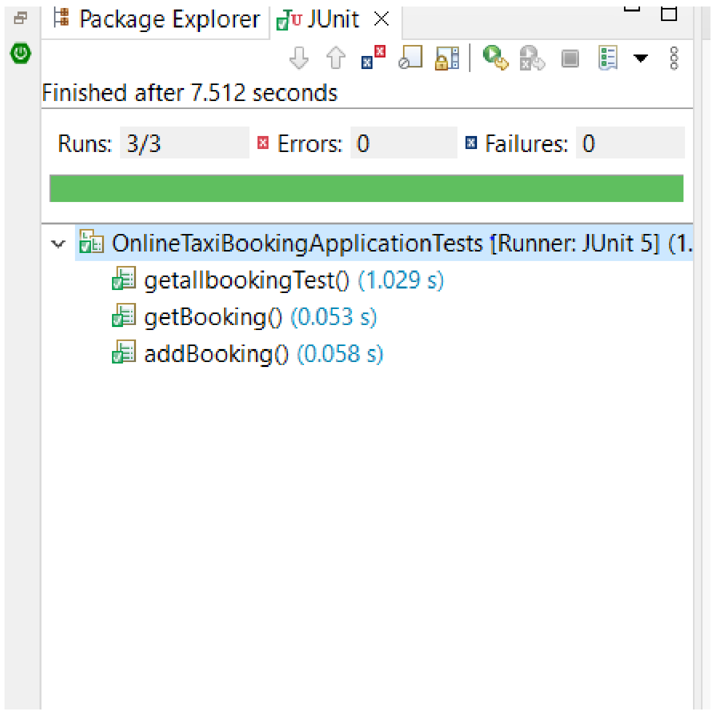
import org.junit.jupiter.api.Order;

import org.junit.jupiter.api.Test;

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

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

```
import com.taxi.controller.BookingController;

import com.taxi.model.Booking;

import com.taxi.model.Cab;

import com.taxi.model.Customer;


@SpringBootTest

class OnlineTaxiBookingApplicationTests {


@Autowired

private BookingController bookingController;


// for booking controller

@Test

void getallbookingTest() {

List<Booking> book = bookingController.findAllBooking();

System.out.println(book);

assertNotNull(book);

}


// for adding


@Test

@Order(2)

void addBooking() {

Booking book = new Booking();
Customer customer = new Customer();
Cab cab = new Cab();

book.setBookingStatus("booked");
customer.setCustomerId(3);
cab.setCabId(1);
book.setCustomer(customer);
book.setCab(cab);

Booking result = bookingController.addBooking(book);
assertNotNull(result);
}

@Test
@Order(2)
void getBooking() {
Booking book = new Booking();
book = bookingController.findBookingId(1);

assertNotNull(book);

}

}
```



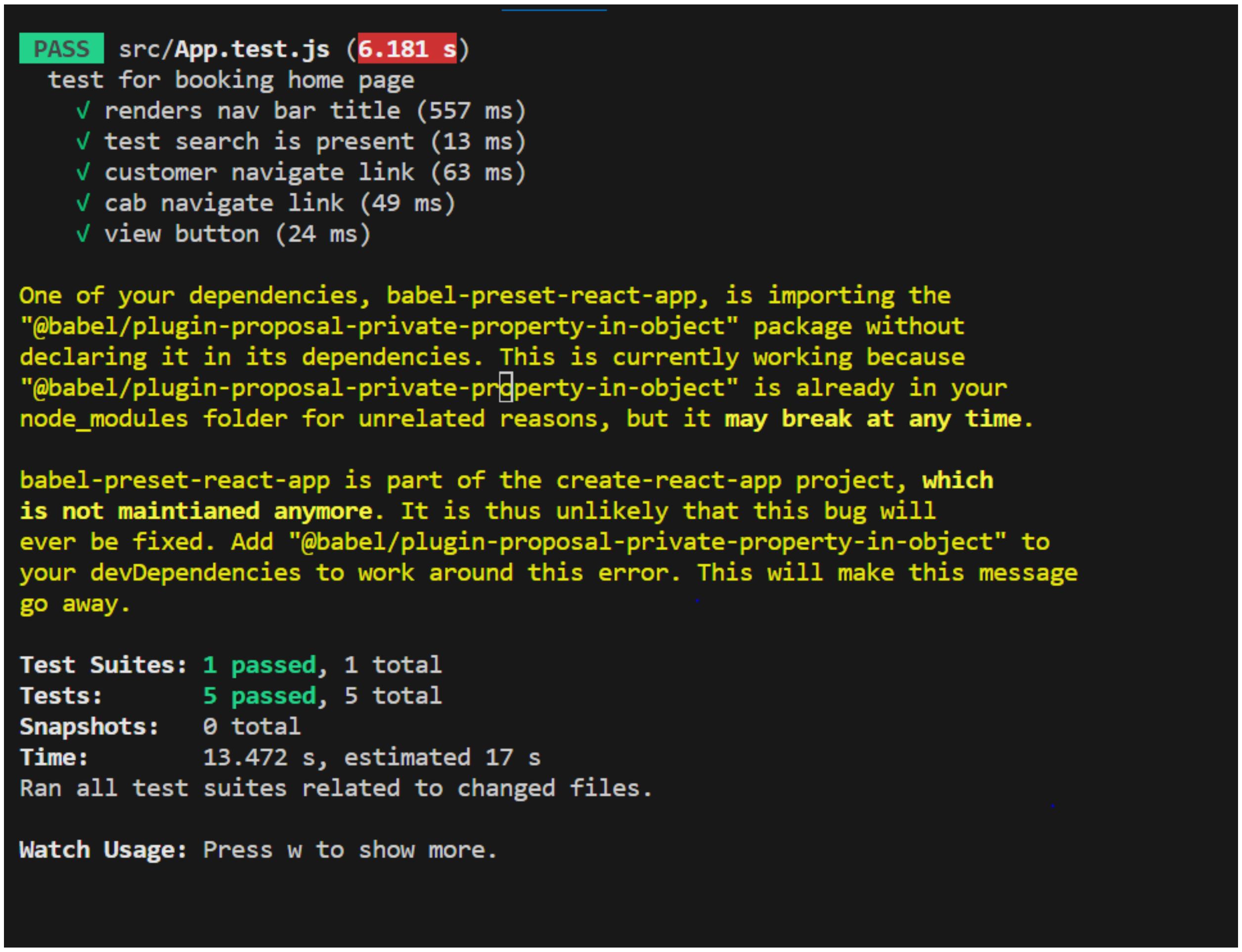
```
Finished after 7.512 seconds
Runs: 3/3   Errors: 0   Failures: 0
OnlineTaxiBookingApplicationTests [Runner: JUnit 5] (1.
    getallbookingTest() (1.029 s)
    getBooking() (0.053 s)
    addBooking() (0.058 s)
```

JEST Tests:

For Booking page:

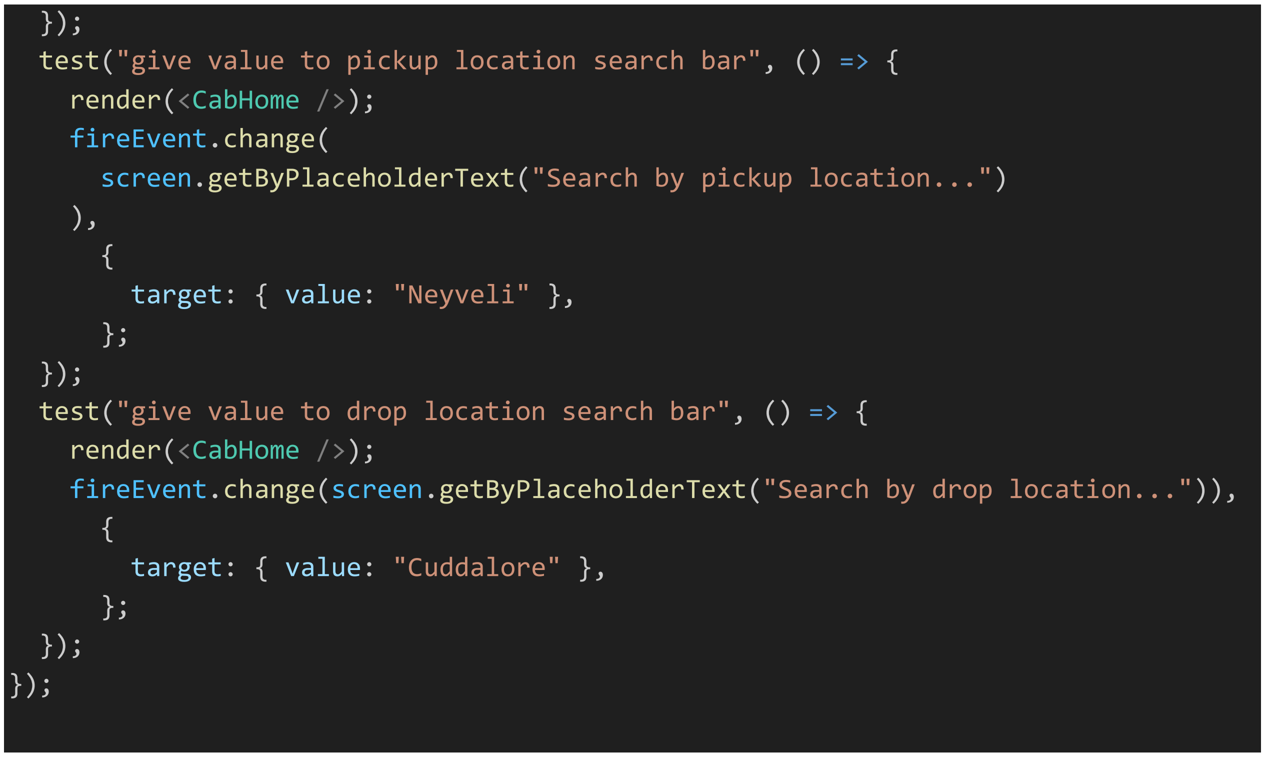
```
import { fireEvent, render, screen } from "@testing-library/react";
import BookingHome from "./Pages/BookingHome";
import CabHome from "./Pages/CabHome";
import "@testing-library/jest-dom";
import AddBooking from "./Pages/AddBookng";
jest.mock("react-router-dom");
```

```
describe("test for booking home page", () => {
  test("renders nav bar title", () => {
    render(<BookingHome />);
    const linkElement = screen.getByTestId("title");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent(
      "Online Taxi Booking Management System"
    );
  });

  test("test search is present", () => {
    render(<BookingHome />);
    const linkElement = screen.getByTitle("search");
    expect(linkElement).toBeInTheDocument();
  });

  test("customer navigate link", () => {
    render(<BookingHome />);
    const customerElement = screen.getByRole("customer");
    expect(customerElement).toBeInTheDocument();
  });

  test("cab navigate link", async () => {
    render(<BookingHome />);
    const cabElement = await screen.findByRole("cab");
    console.log(cabElement);
    expect(cabElement).toBeInTheDocument();
  });

  test("view button", async () => {
    render(<BookingHome />);
    const viewElement = screen.getByText(/View/i);
    expect(viewElement).toBeInTheDocument();
  });
});
```

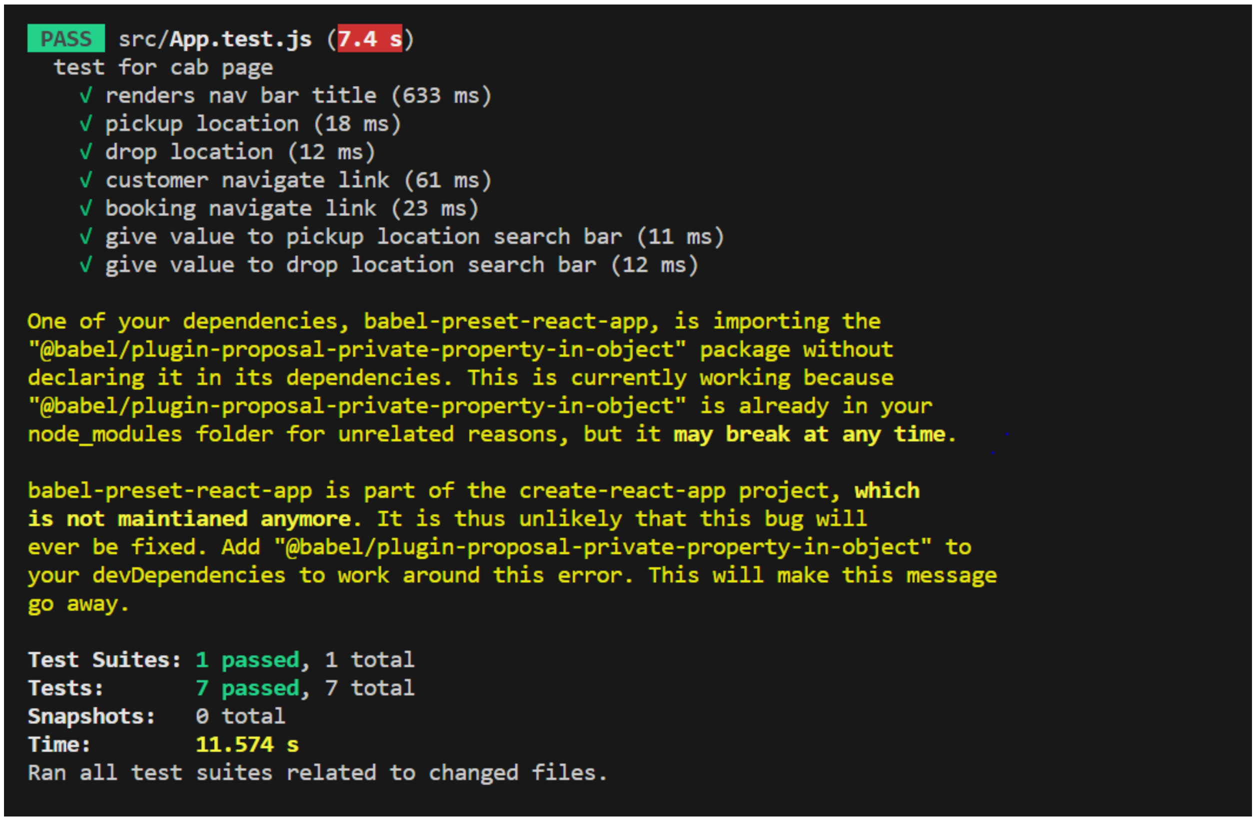

```
 PASS  src/App.test.js (6.181 s)
  test for booking home page
    √ renders nav bar title (557 ms)
    √ test search is present (13 ms)
    √ customer navigate link (63 ms)
    √ cab navigate link (49 ms)
    √ view button (24 ms)

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Test Suites: 1 passed, 1 total
Tests:       5 passed, 5 total
Snapshots:   0 total
Time:        13.472 s, estimated 17 s
Ran all test suites related to changed files.

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

For Cab Page:

```
import { fireEvent, render, screen } from "@testing-library/react";
import BookingHome from "./Pages/BookingHome";
import CabHome from "./Pages/CabHome";
import "@testing-library/jest-dom";
import AddBooking from "./Pages/AddBookng";
jest.mock("react-router-dom");




describe("test for cab page", () => {

  test("renders nav bar title", () => {
        render(<CabHome />);
        const linkElement = screen.getByTestId("title");
        expect(linkElement).toBeInTheDocument();
        expect(linkElement).toHaveTextContent(
          "Online Taxi Booking Management System"
        );
      });
  test("pickup location", () => {
    render(<CabHome />);
    const searchpickup = screen.getByPlaceholderText(
      "Search by pickup location..."
    );
    expect(searchpickup).toBeInTheDocument();
  });

  test("drop location", () => {
    render(<CabHome />);
    const searchdrop = screen.getByPlaceholderText(
      "Search by drop location..."
    );
    expect(searchdrop).toBeInTheDocument();
  });

  test("customer navigate link", () => {
    render(<CabHome />);
    const customerElement = screen.getByRole("customer");
    expect(customerElement).toBeInTheDocument();
  });
  test("booking navigate link", () => {
    render(<CabHome />);
    const customerElement = screen.getByRole("booking");
    expect(customerElement).toBeInTheDocument();
    // expect(linkElement).toHaveTextContent("Go to Customer Page");
```



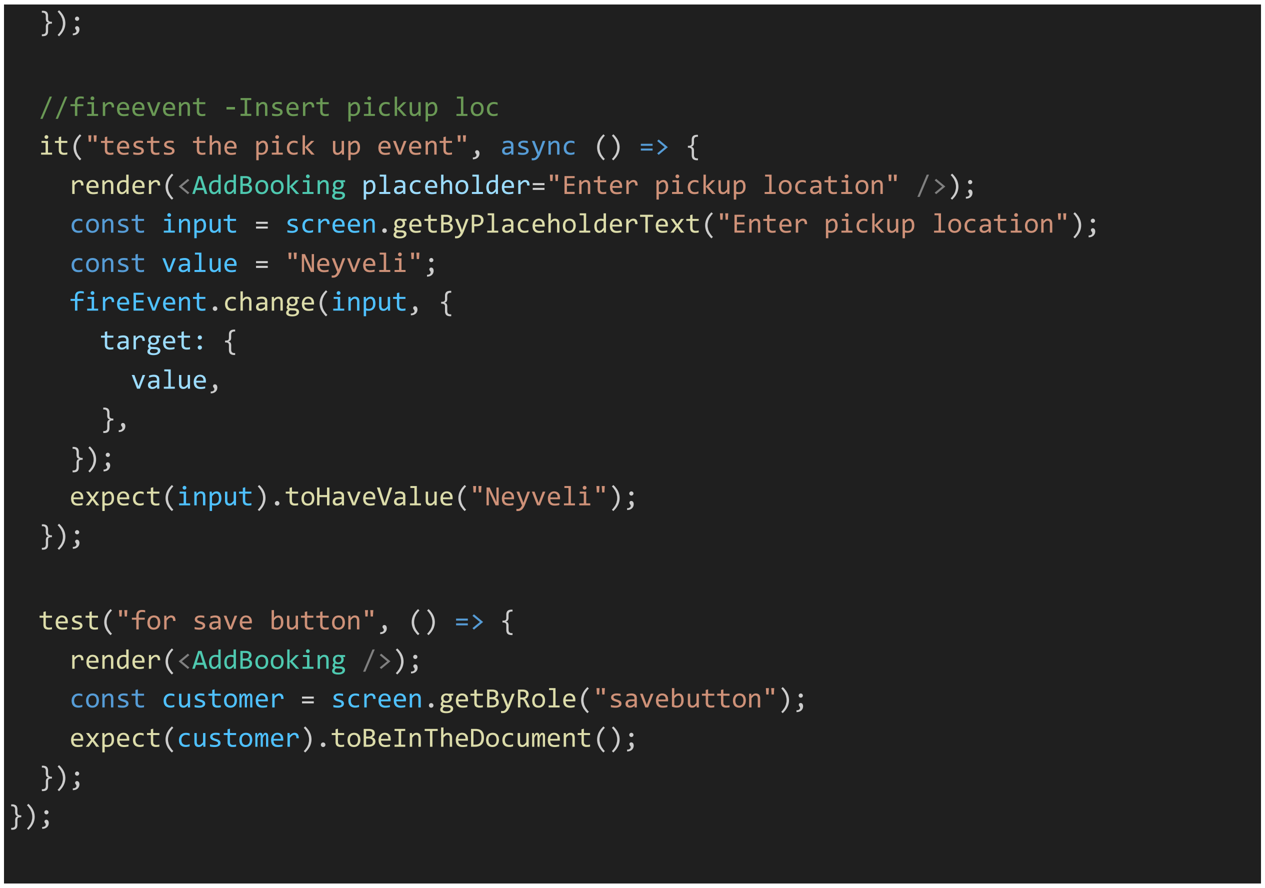
```
  });
  test("give value to pickup location search bar", () => {
    render(<CabHome />);
    fireEvent.change(
      screen.getByPlaceholderText("Search by pickup location...")
    ),
      {
        target: { value: "Neyveli" },
      };
  });
  test("give value to drop location search bar", () => {
    render(<CabHome />);
    fireEvent.change(screen.getByPlaceholderText("Search by drop location...")),
      {
        target: { value: "Cuddalore" },
      };
  });
});
```



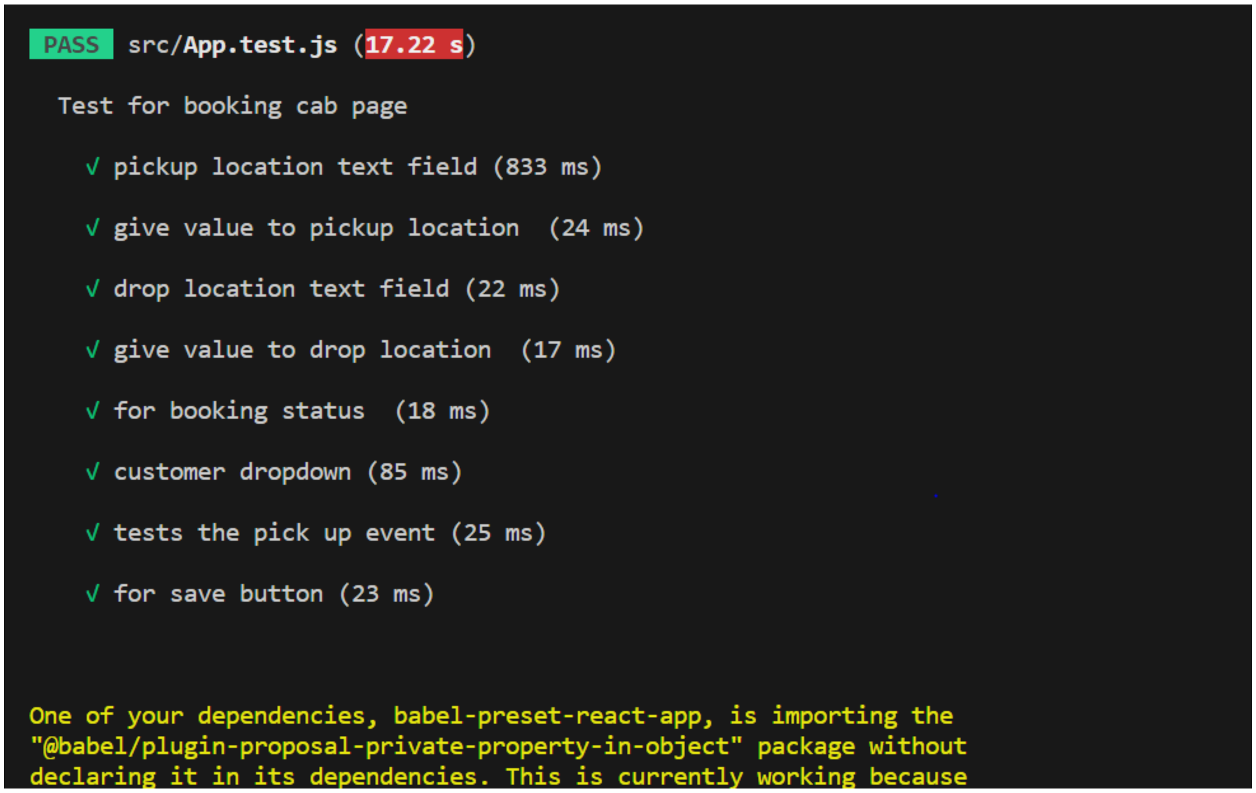
```
 PASS  src/App.test.js (7.4 s)
  test for cab page
    √ renders nav bar title (633 ms)
    √ pickup location (18 ms)
    √ drop location (12 ms)
    √ customer navigate link (61 ms)
    √ booking navigate link (23 ms)
    √ give value to pickup location search bar (11 ms)
    √ give value to drop location search bar (12 ms)

One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Test Suites: 1 passed, 1 total
Tests:       7 passed, 7 total
Snapshots:   0 total
Time:        11.574 s
Ran all test suites related to changed files.
```

For add booking page:

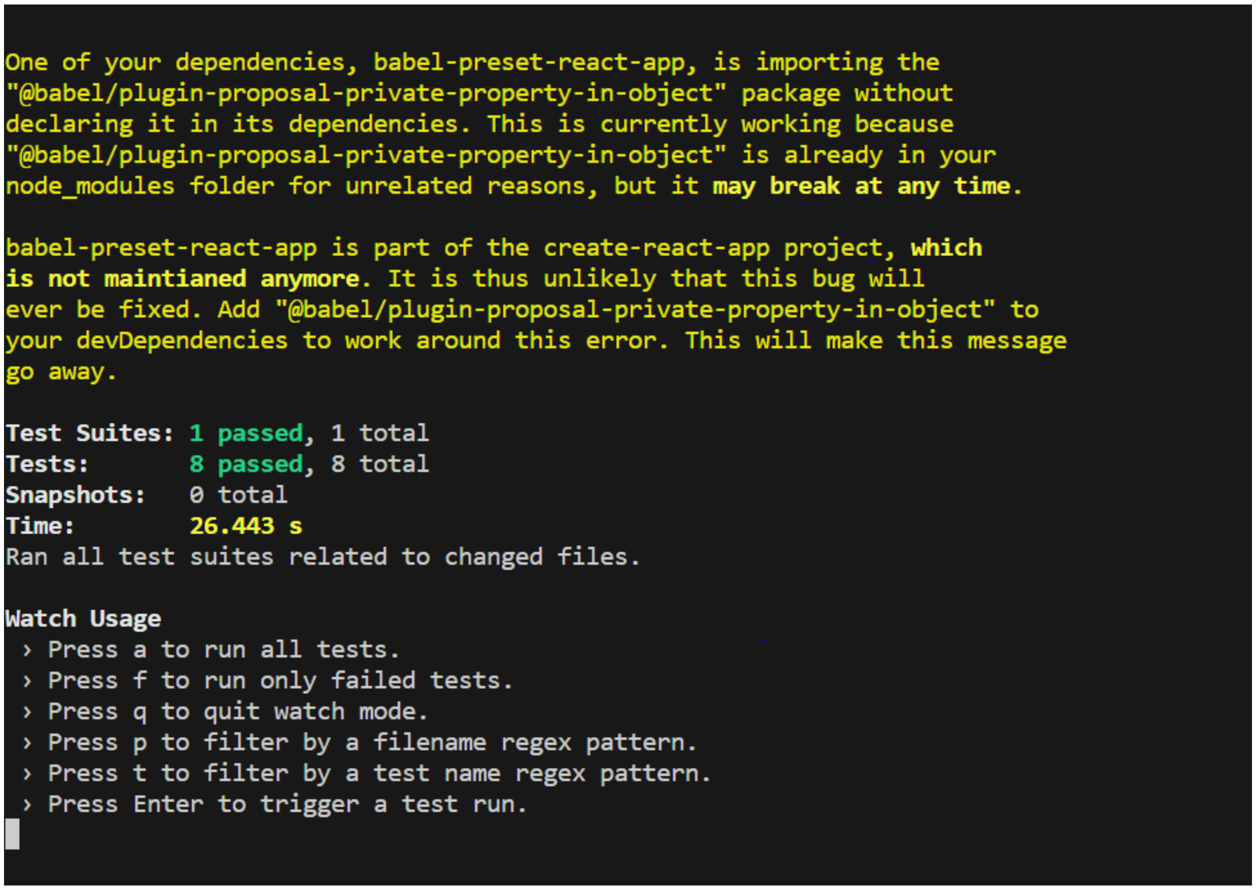
```
describe("Test for booking cab page", () => {
  test("pickup location text field", () => {
    render(<AddBooking />);
    const pickup = screen.getByPlaceholderText("Enter pickup location");
    expect(pickup).toBeInTheDocument();
  });

  test("give value to pickup location ", () => {
    render(<AddBooking />);
    fireEvent.change(screen.getByPlaceholderText("Enter pickup location")),
      {
        target: { value: "Neyveli" },
      };
  });

  test("drop location text field", () => {
    render(<AddBooking />);
    const drop = screen.getByPlaceholderText("Enter drop location");
    expect(drop).toBeInTheDocument();
  });

  test("give value to drop location ", () => {
    render(<AddBooking />);
    fireEvent.change(screen.getByPlaceholderText("Enter drop location")),
      {
        target: { value: "Cuddalore" },
      };
  });

  test("for booking status ", () => {
    render(<AddBooking />);
    fireEvent.change(screen.getByPlaceholderText("Enter booking status")),
      {
        target: { value: "booked" },
      };
  });

  test("customer dropdown", () => {
    render(<AddBooking />);
    const customer = screen.getByRole("customerdropdown");
    expect(customer).toBeInTheDocument();
```



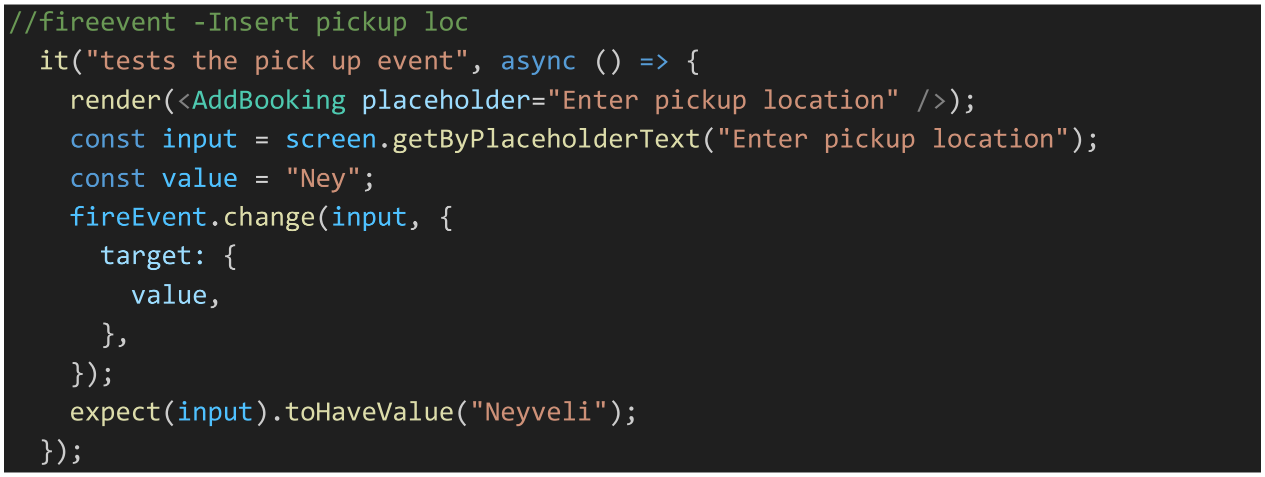
```
  });

  //fireevent -Insert pickup loc
  it("tests the pick up event", async () => {
    render(<AddBooking placeholder="Enter pickup location" />);
    const input = screen.getByPlaceholderText("Enter pickup location");
    const value = "Neyveli";
    fireEvent.change(input, {
      target: {
        value,
      },
    });
    expect(input).toHaveValue("Neyveli");
  });

  test("for save button", () => {
    render(<AddBooking />);
    const customer = screen.getByRole("savebutton");
    expect(customer).toBeInTheDocument();
  });
});
```



```
 PASS  src/App.test.js (17.22 s)

   Test for booking cab page

     √ pickup location text field (833 ms)

     √ give value to pickup location  (24 ms)

     √ drop location text field (22 ms)

     √ give value to drop location  (17 ms)

     √ for booking status  (18 ms)

     √ customer dropdown (85 ms)

     √ tests the pick up event (25 ms)

     √ for save button (23 ms)


 One of your dependencies, babel-preset-react-app, is importing the
 "@babel/plugin-proposal-private-property-in-object" package without
 declaring it in its dependencies. This is currently working because
```

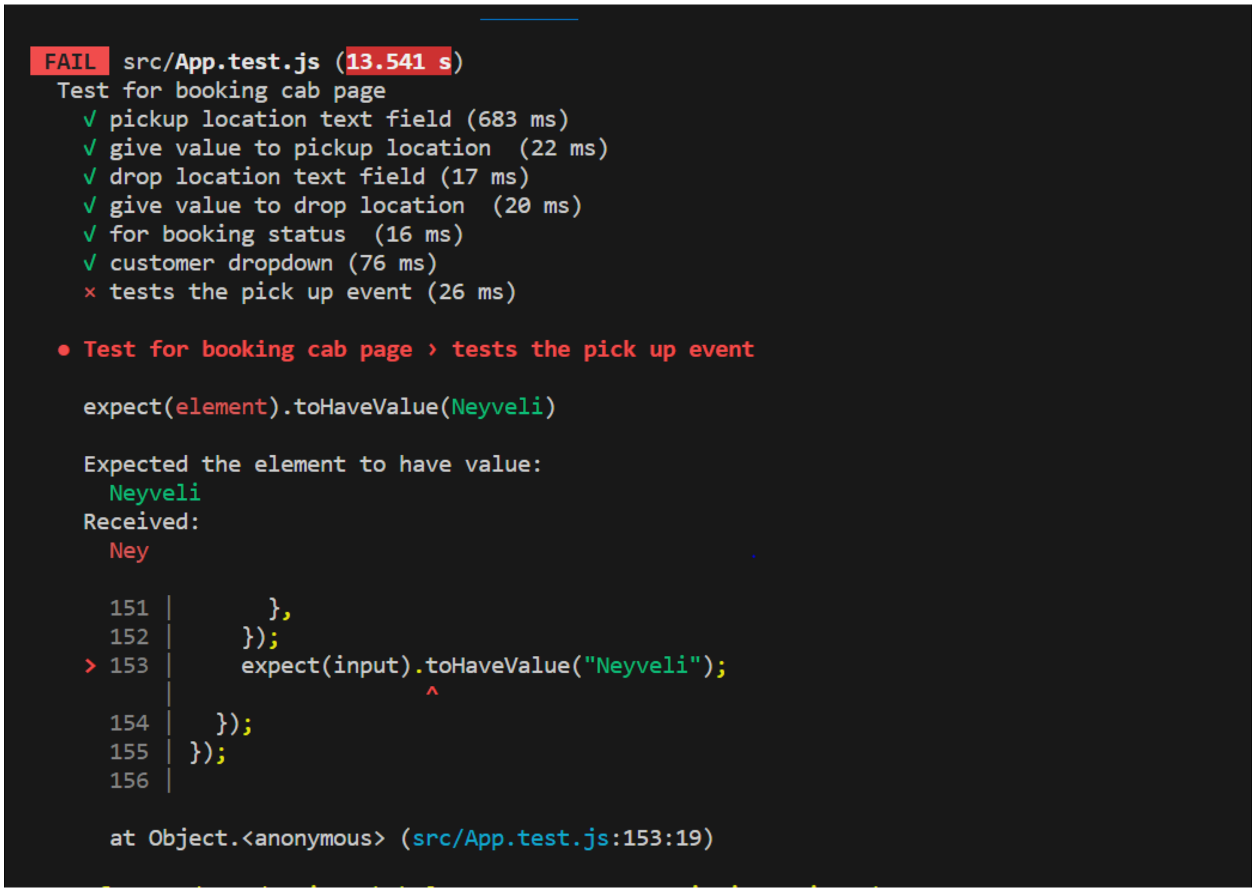


```
One of your dependencies, babel-preset-react-app, is importing the
"@babel/plugin-proposal-private-property-in-object" package without
declaring it in its dependencies. This is currently working because
"@babel/plugin-proposal-private-property-in-object" is already in your
node_modules folder for unrelated reasons, but it may break at any time.

babel-preset-react-app is part of the create-react-app project, which
is not maintianed anymore. It is thus unlikely that this bug will
ever be fixed. Add "@babel/plugin-proposal-private-property-in-object" to
your devDependencies to work around this error. This will make this message
go away.

Test Suites: 1 passed, 1 total
Tests:       8 passed, 8 total
Snapshots:   0 total
Time:        26.443 s
Ran all test suites related to changed files.

Watch Usage
 › Press a to run all tests.
 › Press f to run only failed tests.
 › Press q to quit watch mode.
 › Press p to filter by a filename regex pattern.
 › Press t to filter by a test name regex pattern.
 › Press Enter to trigger a test run.
```



```
//fireevent -Insert pickup loc
  it("tests the pick up event", async () => {
    render(<AddBooking placeholder="Enter pickup location" />);
    const input = screen.getByPlaceholderText("Enter pickup location");
    const value = "Ney";
    fireEvent.change(input, {
      target: {
        value,
      },
    });
    expect(input).toHaveValue("Neyveli");
  });
```

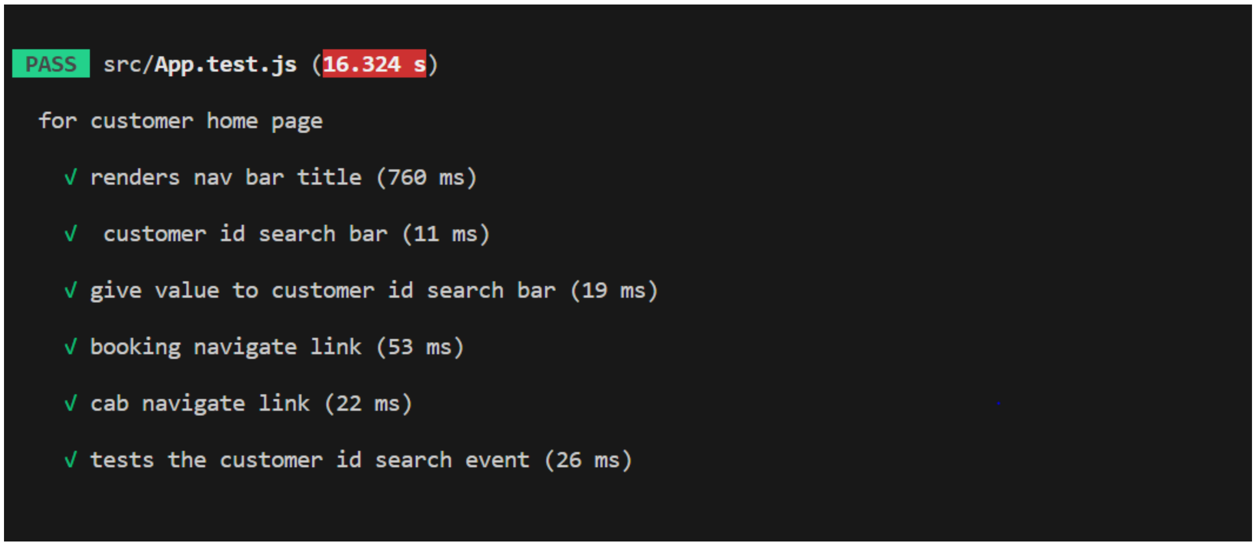
If we give unmatched data means, testcase will fail.



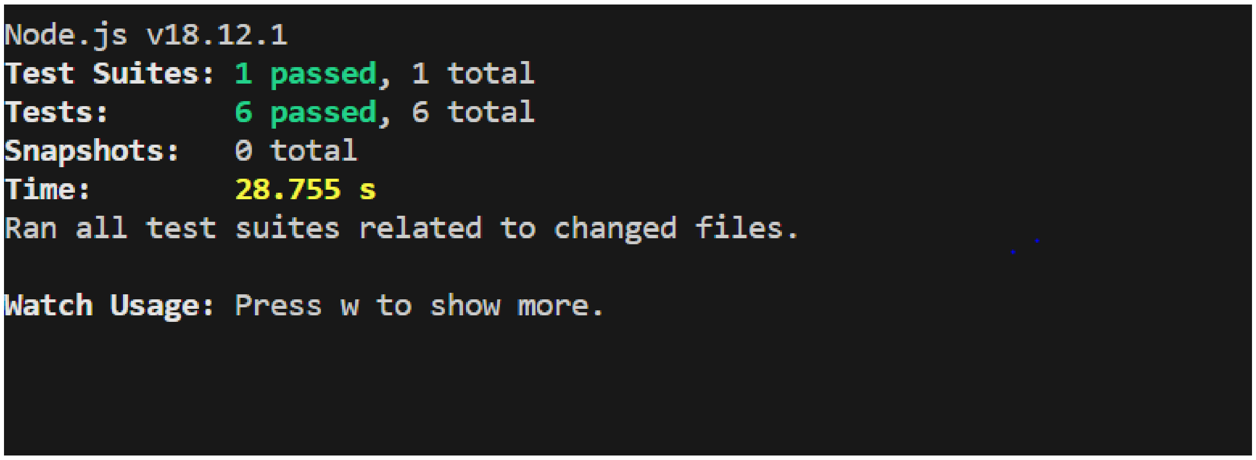
```
 FAIL  src/App.test.js (13.541 s)
  Test for booking cab page
    √ pickup location text field (683 ms)
    √ give value to pickup location  (22 ms)
    √ drop location text field (17 ms)
    √ give value to drop location  (20 ms)
    √ for booking status  (16 ms)
    √ customer dropdown (76 ms)
    × tests the pick up event (26 ms)

  ● Test for booking cab page › tests the pick up event

    expect(element).toHaveValue(Neyveli)

    Expected the element to have value:
      Neyveli
    Received:
      Ney

      151 |         },
      152 |       });
    > 153 |       expect(input).toHaveValue("Neyveli");
          |                     ^
      154 |     });
      155 |   });
      156 |

      at Object.<anonymous> (src/App.test.js:153:19)
```

For customer page:

```
describe("for customer home page", () => {
  test("renders nav bar title", () => {
    render(<CustomerHome />);
    const linkElement = screen.getByTestId("title");
    expect(linkElement).toBeInTheDocument();
    expect(linkElement).toHaveTextContent(
      "Online Taxi Booking Management System"
    );
  });

  test(" customer id search bar", () => {
    render(<CustomerHome />);
    const searchcust = screen.getByPlaceholderText("Search by id...");
    expect(searchcust).toBeInTheDocument();
  });

  test("give value to customer id search bar", () => {
    render(<CustomerHome />);
    fireEvent.change(screen.getByPlaceholderText("Search by id...")),
      {
        target: { value: "1" },
      };
  });

  test("booking navigate link", () => {
    render(<CustomerHome />);
    const customerElement = screen.getByRole("booking");
    expect(customerElement).toBeInTheDocument();
  });
  test("cab navigate link", () => {
    render(<CustomerHome />);
    const customerElement = screen.getByRole("cab");
    expect(customerElement).toBeInTheDocument();
  });

  //fireevent - search customer id
  it("tests the customer id search event", async () => {
    render(<CustomerHome placeholder="Search by id..." />);
    const input = screen.getByPlaceholderText("Search by id...");
    const value = "1";
    fireEvent.change(input, {
      target: {
        value,
      },
    });
    expect(input).toHaveValue("1");
  });
});
```

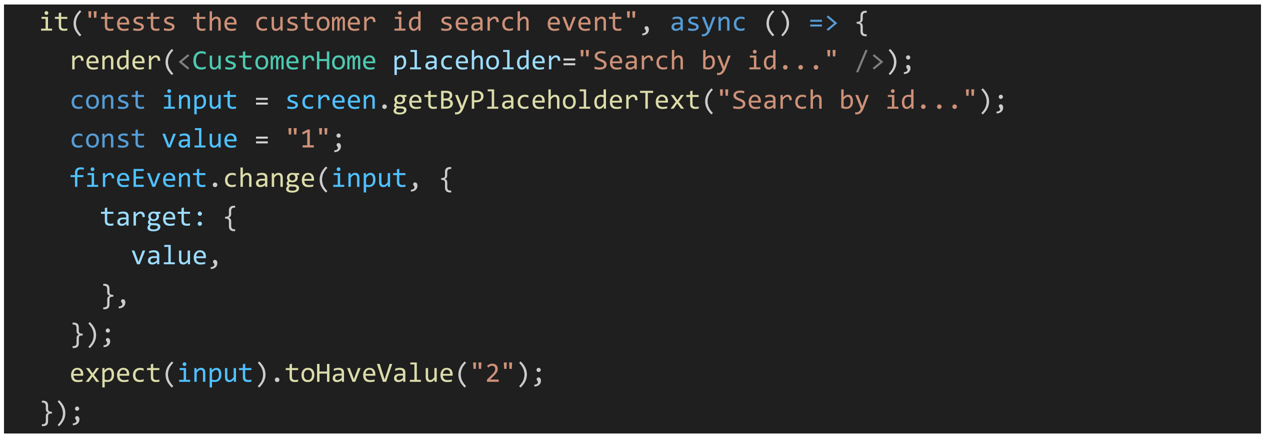


```
PASS  src/App.test.js (16.324 s)
  for customer home page
    √ renders nav bar title (760 ms)
    √  customer id search bar (11 ms)
    √ give value to customer id search bar (19 ms)
    √ booking navigate link (53 ms)
    √ cab navigate link (22 ms)
    √ tests the customer id search event (26 ms)
```



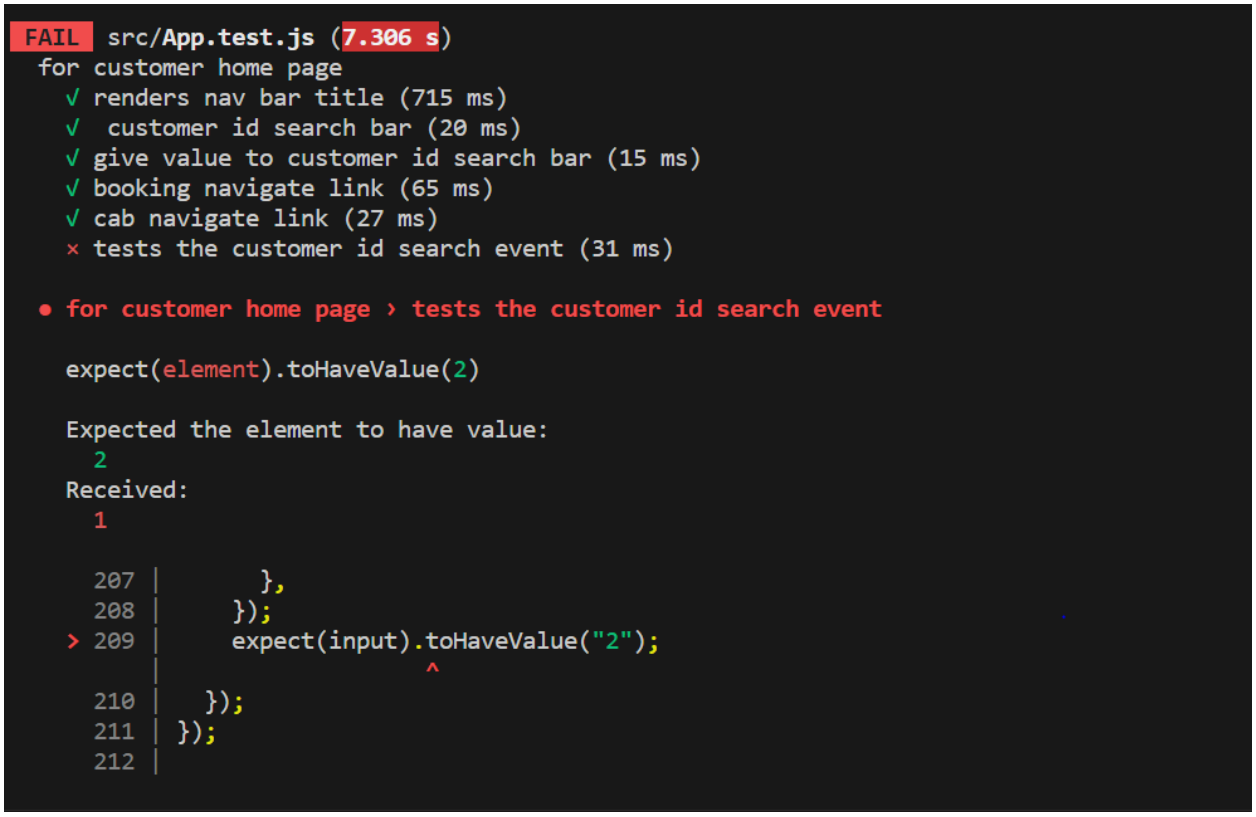
```
Node.js v18.12.1
Test Suites: 1 passed, 1 total
Tests:       6 passed, 6 total
Snapshots:   0 total
Time:        28.755 s
Ran all test suites related to changed files.

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

```
//fireevent -search customer id
```



```
it("tests the customer id search event", async () => {
  render(<CustomerHome placeholder="Search by id..." />);
  const input = screen.getByPlaceholderText("Search by id...");
  const value = "1";
  fireEvent.change(input, {
    target: {
      value,
    },
  });
  expect(input).toHaveValue("2");
});
```

If we give unmatched data means, testcase will fail.



```
FAIL  src/App.test.js (7.306 s)
  for customer home page
    √ renders nav bar title (715 ms)
    √  customer id search bar (20 ms)
    √ give value to customer id search bar (15 ms)
    √ booking navigate link (65 ms)
    √ cab navigate link (27 ms)
    × tests the customer id search event (31 ms)

  ● for customer home page › tests the customer id search event

    expect(element).toHaveValue(2)

    Expected the element to have value:
      2
    Received:
      1

                 },
               });
    >          expect(input).toHaveValue("2");
                            ^
             });
           });
```